Multivariate Time Series Forecasting of Patient's Health using LSTM

By: Shardul Ghuge

Supervisor: Prof. Michael Guerzhoy

April 2023

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

This thesis investigates the robustness and performance of Long Short-Term Memory (LSTM) networks for multivariate time series forecasting in Intensive Care Unit (ICU) settings using the MIMIC-IV dataset. Our approach addresses the challenges posed by high-dimensional, noisy, and sparse Electronic Health Record (EHR) data while capturing complex temporal dependencies and non-linear relationships. We evaluate the LSTM model's performance under various conditions, such as the presence of random noise and varying dataset sizes, and demonstrate its ability to effectively capture correlations and trends in the data. The model achieves a Root Mean Squared Error (RMSE) of approximately **0.05** for multivariate time series forecasting of all 83 health variables and demonstrates promising results in predicting patient mortality with varying mortality rates.

In our analysis, we also assess the model's robustness in the presence of noise and its performance across different dataset sizes. Results demonstrate that the model's performance remains relatively stable across different dataset sizes, with only a slight increase in R-squared (R^2) score as the dataset size increases, implying the LSTM's ability to capture underlying relationships even with smaller training data. Our findings also reveal that the LSTM model can effectively capture underlying relationships between features and forecast future time steps in the presence of significant noise, with the breaking point between 68 and 75 random features out of 83. This research highlights the potential of LSTM networks for multivariate time series forecasting in healthcare settings and contributes to the development of more effective patient care and monitoring systems in ICUs and beyond.

Shardul Ghuge ESC499Y1

Acknowledgements

I take this opportunity to express my deepest sense of gratitude and sincere thanks to everyone who helped me to complete this work successfully. I would like to place on record my sincere gratitude to my supervisor Prof. Michael Guerzhoy for the guidance and mentorship throughout the course. Finally, I thank my family, and friends who contributed to the successful fulfillment of this thesis.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, the application of machine learning in medical research has expanded significantly. A key area of focus involves predicting patients' future conditions based on their past medical history. Examples include predicting patient mortality in the ICU, the probability of heart failure given certain diagnoses, and more [1]. Consequently, discriminative modeling has dominated this field, leaving generative modeling relatively underexplored. This thesis aims to address this research gap by utilizing Electronic Health Records (EHR) to develop generative models that create realistic synthetic patient data.

Continuous health monitoring is a critical component of patient care, especially in the ICU. Failure to rescue (FTR) is a leading cause of mortality in hospitals, accounting for 13% to 22% of cases [2]. Traditional methods, such as linear regression and autoregressive integrated moving average (ARIMA), have limitations in handling (a) complex temporal dependencies, (b) non-linear relationships, (c) large-scale/high-dimensional data, and (d) missing data. This thesis aspires to develop a predictive health monitoring method that addresses these issues.

The primary motivation behind this research is the potential benefits of generative modeling for predictive health monitoring. Current work in the field is dominated by discriminative modeling, focusing on tasks such as mortality and heart failure prediction. However, generative modeling, commonly used for image and text generation, can be highly valuable in predictive health monitoring. The widespread implementation of Electronic Health Records (EHR) now makes it possible to explore this area.

The main goals of this thesis are to: (a) evaluate the use of LSTM as a viable choice for performing multivariate time series forecasting; (b) identify the underlying cause-and-effect relationships between the latent variables of the model; and (c) uncover the limitations of the model. The LSTM model will be trained on the MIMIC-IV dataset, which contains de-identified EHR data for over 40,000 ICU patients at the Beth Israel Deaconess Medical Center [3]. This dataset provides valuable temporal data, including diagnoses, lab events, procedures, and prescriptions, which is essential for learning the progression of latent variables.

The LSTM model will generate synthetic data, which will then be validated by comparing the properties of the synthetic data to the real data. The primary objective of this thesis is to use hypothesis testing (statistical or machine learning-based) on the synthetic dataset to uncover the latent variables, thereby performing causal inference on the model. This will provide insights into the cause-and-effect relationships between latent variables and improve our understanding of patients' health progression in the ICU.

In addition to addressing the research gap in generative modeling, this thesis will also explore the practical applications of synthetic EHR data. Generating realistic synthetic data can help overcome the challenges associated with accessing real EHR data, which often contains sensitive or regulated medical information about patients [4]. By developing a method for generating synthetic data, this thesis aims to prevent the healthcare sector from lagging behind in utilizing machine learning-based tools.

Therefore, this thesis will first review the existing literature on time series forecasting models and RNN medical modeling. Next, it will describe the sources of data used and their preparation techniques. Subsequently, two RNN models will be explained and used as building blocks for the time series forecasting models. Lastly, the methods for evaluating the models and their results will be discussed, along with the exploration of causal inference to gain a better understanding of the cause-and-effect relationships between the latent variables.

2 Literature Review

This section will consist of four subsections. The first will provide a general overview of data relevant to this research and some of the data preparation techniques. The second will discuss existing Time series forecasting techniques to predict patients' health conditions. The third will look at trends in machine learning-based medical modelling, with a focus on RNN-based techniques. Lastly, literature will be examined for Causal Inference techniques on trained ML models (RNN's).

2.1 Datasets and Preparation

This subsection will provide a general overview of the structure of MIMIC-IV EHR, as well as a brief overview of preprocessing steps that are typically necessary for these datasets to get them ready for training.

2.1.1 MIMIC-IV Data and Preprocessing

The widespread implementation of electronic health record (EHR) systems has become more prevalent over recent years, with a reported 84% of hospitals and 86% of office-based physicians adopting some form of EHR by the year 2017 [5] [6]. This widespread adoption of EHRs has resulted in the creation of large amounts of patient medical history data. This data is composed of both static features, such as patient demographics, and dynamic data including vital signs and sequences of non-quantitative events, such as diagnoses and physician notes [7].

When looking at EHR data, the sequential event data is often separated into distinct hospital admissions, as these admissions tend to be short-term events. As a result, it is common to clean and filter the data based on the number of admissions. In order to make predictions on a short-term basis, some authors choose to retain only the first admission per patient, as demonstrated by Suresh et al. [8]. However, for longer-term predictions, only patients with a minimum number of admissions, typically 2, are considered [9], [10]. Additionally, individual ICU stays may also have restrictions in terms of their length. For instance, Kaji et al. [11] removed stays that were shorter than 2 days, and truncated stays that exceeded 14 days. Similarly, Suresh et al. considered only ICU stays lasting between 12 and 240 hours.

As is the case with many real-world datasets, including MIMIC, electronic health record (EHR) datasets often have missing and incorrect values. To address these issues, various approaches have been taken. Pham et al. [9] eliminate patients with missing demographic information in categorical form. In terms of handling missing quantitative features, models may rely heavily on imputing the missing values. Kaji et al. opt to exclude features that have missing values in more than 25% of cases and use the median of each variable to fill in the remaining missing values. They also address incorrect values recorded in MIMIC by replacing values that exceed a 95th percentile threshold with the median for that variable. Lipton et al. [12] fill in missing values through both forward-filling and back-filling, though they caution that back-filling is inappropriate for forecasting tasks . If a variable is not measured for a patient, clinically normal values are used as physicians typically do not order lab tests for variables they believe to be within the normal range. In addition, when dealing with extremely rare clinical events with limited associated data, it is common to eliminate these events. For instance, Farhan et al. [13] exclude any clinical events that occur in less than 1% of all sequences.

2.2 Time series forecasting techniques to predict patient's health

Time series forecasting is a statistical technique used to predict future events or outcomes based on historical data collected over time. In healthcare, time series forecasting is commonly used to predict a patient's health condition. This can include predicting the risk of a particular disease, hospital readmission, or the likelihood of developing a certain medical condition. The forecasting models are often trained on electronic health records (EHR) data, which includes demographic information, clinical observations, and laboratory results. Common time series forecasting techniques include statistical models (e.g. ARIMA, exponential smoothing), machine learning algorithms (e.g. random forests, gradient boosting), and deep learning models (e.g. LSTMs, CNNs). The accuracy of these methods varies depending on the complexity of the forecasting problem and the quality of the data used to train the models.

Several studies have used the MIMIC EHR dataset to forecast patient health conditions using time series analysis techniques. For example, a study by Johnson et al. [14] used a combination of traditional statistical models and machine learning algorithms to forecast the risk of death in intensive care unit (ICU) patients. The authors found that a gradient boosting machine (GBM) model outperformed the other methods, achieving an area under the receiver operating characteristic curve (AUROC) of 0.920.

These studies demonstrate the potential of using the MIMIC EHR dataset and time series analysis techniques to forecast patient health conditions. However, there is still room for improvement in terms of the accuracy and generalizability of these methods. In the present study, we aim to further advance the state of the art by focusing on the multivariate forecasting case and considering causality relationship between the various latent variables.

2.2.1 Univariate Time Series Forecasting

Univariate time series forecasting is a type of forecasting method that uses only one variable, or feature, to make predictions about future values of the same variable. In the context of predicting patient health conditions, this could involve using a single time-series feature, such as the patient's body temperature, heart rate, or blood pressure, to make predictions about future changes in the patient's health status.

One advantage of univariate time series forecasting is that it is relatively simple to implement and can be done with basic statistical techniques, such as moving averages, exponential smoothing, or ARIMA models. Additionally, univariate time series forecasting can be useful when there is a strong relationship between the variable being used for forecasting and the target variable of interest (i.e. the patient's health condition).

There have been numerous studies in recent years that have used univariate time series forecasting techniques to predict patient health conditions. These studies have primarily employed statistical and machine learning methods, such as ARIMA, SARIMA, SVM, LSTM, etc.

One example is the study by Zhou et al. [15], who used an ARIMA model to forecast the inpatient length of stay (LOS) for critically ill patients. They found that the ARIMA model outperformed other traditional time series models and was able to accurately predict patient LOS with a mean absolute error of 1.26 days.

Another study by Wang et al. [16] developed a machine learning model to accurately predict sepsis in intensive care unit (ICU) patients. The model was based on univariate forecasting, using a range of variables such as age, gender, past medical history, and physiological parameters. The model was tested on a dataset of ICU patients and achieved an accuracy of 0.87.

LSTM, a type of recurrent neural network, has also been widely used in the prediction of patient health conditions. For example, Golas et al. [17] applied LSTM to predict the risk of 30-day readmission for heart failure patients. Their findings indicated that the LSTM model was able to accurately predict readmission risk with an AUC of 0.81.

In conclusion, univariate time series forecasting techniques have shown promise in predicting various patient health conditions, with ARIMA, SARIMA, and LSTM being among the most commonly used methods. These techniques have been effective in accurately forecasting patient LOS, sepsis severity, and readmission risk, among other health conditions [18].

2.2.2 Multivariate Time Series Forecasting

In contrast, multivariate time series forecasting involves modelling and predicting multiple related variables over time. In this type of forecasting, the goal is to model the interdependencies and relationships between multiple time series in order to make predictions about their future values. This type of forecasting is useful when there are multiple variables that have an impact on the outcome of interest, and it can be more accurate than univariate forecasting because it takes into account the complex relationships between the variables. In the context of predicting patient health conditions, this could involve using multiple time-series features, such as the patient's body temperature, heart rate, and blood pressure, to make more accurate predictions about the patient's health status.

One advantage of multivariate time series forecasting is that it can capture complex relationships between variables that may not be captured by a univariate approach. For example, changes in a patient's heart rate may be closely related to changes in their body temperature, and incorporating both variables into the forecasting model may result in more accurate predictions.

Examples of studies that have used multivariate time series forecasting for predicting patient health conditions include [8], [19], [20]. These studies have found that using multiple features can lead to improved accuracy in predicting patient health outcomes, such as clinical interventions, ICU stays, and mortality.

Suresh et al.'s paper [8] explores the use of a multivariate time series model for predicting clinical interventions within ICU. The authors use various patient features, such as demographic information and previous hospitalization records, as well as time-series data on patients' physiological variables, such as heart rate and blood pressure, to build their model. The results show that the multivariate time series model outperforms univariate models in predicting clinical interventions.

Chen et al.'s paper [19] focuses on using multivariate time series analysis to predict patients' length of stay in the ICU. The authors use data on patients' vital signs, laboratory results, and demographic information to build their model. They compare their results with those of univariate models and find that the multivariate model provides a more accurate prediction of ICU stays.

Kim et al.'s paper [20] investigates the use of multivariate time series analysis for predicting patient mortality in the ICU. The authors use data on patients' vital signs, laboratory results, and demographic information to build their model and compare their results with those of univariate models. They find that the multivariate model provides a more accurate prediction of patient mortality in the ICU. These three papers provide evidence for the benefits of using multivariate time series models over univariate models for predicting patient health outcomes in healthcare settings.

2.3 RNN Based Time Series Forecasting

From the previous section, we have determined that the Multivariate Time Series Forecasting is the way to go for better predictions. Then, several ways to perform multivariate time series forecasting include: Vector Autoregression (VAR), Vector Autoregression Moving Average (VARMA), Vector Autoregression Integrated Moving Average (VARIMA), Multivariate Linear Regression, Multivariate ARIMA, Multivariate State Space Models (MS-ARIMA), Convolutional Neural Networks (CNNs), Transfer Learning and Recurrent Neural Networks (RNN's) [21]. Each of these methods has its own strengths and limitations depending on the specific characteristics of the data being analyzed.

Out of these, for the purpose of our use case, RNN (Recurrent Neural Networks) models are commonly used for modeling and forecasting medical and clinical time series data due to their ability to capture patterns in sequential data. There are several variants of RNN models, including vanilla RNN, LSTM (Long Short-Term Memory), and GRU (Gated Recurrent Units). These models are built on the concept of passing information through hidden states, allowing them to effectively retain information over long sequences.

Furthermore, Recurrent Neural Network (RNN) based techniques are preferred in certain situations where the data has certain characteristics [22] [23] [24] such as:

1. Complex Temporal Dependencies: RNNs are designed to handle sequential data, making them a good choice for multivariate time series data where the variables are dependent on each other over time.

2. Non-Linear Relationships: RNNs can capture non-linear relationships between variables in a time series, making them well suited for complex datasets with non-linear patterns.

3. Large-Scale Data: RNNs can handle large amounts of data, making them a good choice for datasets with large number of variables or long time series data.

4. Missing Data: RNNs can handle missing data in a time series, as they have the ability to incorporate time dependencies in the missing data points.

In summary, RNN-based techniques are preferred when the time series data has complex temporal dependencies, non-linear relationships, is large-scale or has missing data. However, it is important to note that RNNs can be computationally expensive and may require a lot of data to train effectively.

2.3.1 Deep Dive into Mathematics of RNN's

Previously, we described the certain situations (section 2.3) where RNN's are preferred for time series forecasting. In this section, the underlying mathematics of the various types of RNN's is evaluated which allow it to model and forecast medical and clinical time series data very well. Particularly, RNN models are well-suited for modeling and forecasting medical and clinical time series data due to their ability to handle sequential and time-dependent data. The mathematics behind different types of RNN models are briefly explained below:

1. Vanilla RNNs use a simple feedforward structure that includes a hidden state. At each time step, the hidden state receives information from the previous time step, as well as from the current input. The hidden state is updated using the following equation:

$$h_t = \tanh(W_{xh} \cdot x_t + W_{hh} \cdot h_{t-1} + b_h)$$

   where x_t is the input at time t, h_t is the hidden state at time t, W_{xh} and W_{hh} are weight matrices, and b_h is a bias vector.

2. GRUs are another type of RNN that use gating mechanisms to control the flow of information. GRUs have two gates: the update gate and the reset gate. The update equations for the hidden state in a GRU are as follows:

$$\begin{aligned}
z_t &= \sigma(W_{xz} \cdot x_t + W_{hz} \cdot h_{t-1} + b_z) \\
r_t &= \sigma(W_{xr} \cdot x_t + W_{hr} \cdot h_{t-1} + b_r) \\
g_t &= \tanh(W_{xg} \cdot x_t + W_{hg} \cdot (r_t \cdot h_{t-1}) + b_g) \\
h_t &= (1 - z_t) \cdot h_{t-1} + z_t \cdot g_t
\end{aligned}$$

where z_t and r_t are the update and reset gates respectively, and g_t is the candidate hidden state.

3. LSTMs are a type of RNN that address the vanishing gradient problem by using gating mechanisms to control the flow of information through the hidden state. LSTMs have three gates: the input gate, forget gate, and output gate. The update equations for the hidden state in an LSTM are as follows:

$$
\begin{aligned}
i_t &= \sigma(W_{xi} \cdot x_t + W_{hi} \cdot h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} \cdot x_t + W_{hf} \cdot h_{t-1} + b_f) \\
o_t &= \sigma(W_{xo} \cdot x_t + W_{ho} \cdot h_{t-1} + b_o) \\
g_t &= \tanh(W_{xg} \cdot x_t + W_{hg} \cdot h_{t-1} + b_g) \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot g_t \\
h_t &= o_t \cdot \tanh(c_t)
\end{aligned}
$$

where i_t, f_t, and o_t are the input, forget, and output gates respectively, g_t is the cell state, and c_t is the cell state at time t.

In conclusion, the mathematics behind RNNs, especially LSTMs and GRUs, allows them to effectively handle sequential and time-dependent data, making them well-suited for modeling and forecasting medical and clinical time series data. Particularly, Long Short-Term Memory (LSTM) networks build upon the structure of Vanilla Recurrent Neural Networks (RNNs) and Gated Recurrent Units (GRUs) in several ways that make them a good choice for multivariate time series forecasting for medical and clinical time series data. LSTMs have a memory cell that can store information for a longer duration, allowing them to capture longer-term dependencies in the data. This makes LSTMs better suited for data with long-term dependencies, such as medical and clinical time series data. Also, LSTMs have three gates (input, forget, and output gates) that control the flow of information into and out of the memory cell. This allows LSTMs to selectively forget or retain information as needed, which can improve the model's ability to capture complex patterns in the data. Lastly, LSTMs have a structure that allows for better gradient flow during training, which can lead to faster convergence and better performance. Hence, the different gating mechanisms and memory cell allow LSTMs to control the flow of information, which helps to mitigate the vanishing gradient problem and improve model performance.

2.3.2 Feature Engineering and Embedding

Feature engineering and embedding are critical components of the RNN (LSTM) model building process for multivariate time series forecasting of medical and clinical time series data. Feature engineering involves transforming raw input data into a set of features that can be used as input to the RNN model. These features capture the important patterns and relationships in the data that can be used to make accurate predictions.

Embedding refers to the process of representing categorical variables as dense vectors. Categorical variables, such as patient demographics, are often present in medical and clinical time series data. By embedding these variables, the RNN model can learn more complex relationships between the categorical variables and the target variable.

In the context of multivariate time series forecasting, feature engineering can involve the extraction of lags, trends, and seasonality in the data, as well as the creation of interaction features between different variables. These features can provide the RNN model with additional information about the underlying patterns in the data, helping to improve its accuracy.

Embedding can be implemented in several ways, including one-hot encoding and the use of pre-trained embeddings. One-hot encoding involves creating a binary representation of each categorical variable, where each category is represented by a different binary column. Pre-trained embeddings are learned from a large corpus of text data and can be used to represent categorical variables in a dense, low-dimensional space.

In summary, feature engineering and embedding are critical components of the RNN (LSTM) model building process for multivariate time series forecasting of medical and clinical time series data. Feature engineering helps to capture important patterns and relationships in the data, while embedding allows the model to learn complex relationships between categorical variables and the target variable.

2.3.3 Handling Variable Time Intervals Between Events

Handling variable time intervals between events is a common challenge in time series forecasting using Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. In medical and clinical time series data, events often occur at irregular intervals and the time difference between two events can vary widely. This makes it difficult for traditional RNNs and LSTMs to capture the correct temporal dependencies between events.

To address this challenge, several approaches have been proposed. One common approach is to use a timestamp encoding technique to convert the variable time intervals into a numerical representation that can be used as an input to the RNN. For example, one could represent the time difference between two events as the number of time steps that have elapsed since the previous event. This encoding can be done using an exponential decay function to give more weight to recent events and less weight to events that occurred a long time ago.

Another approach is to use a custom LSTM architecture that is designed specifically to handle variable time intervals. One example of this is the Interval LSTM, which uses an additional neural network layer to process the time intervals and integrate them into the LSTM model. The Interval LSTM can then effectively capture the temporal dependencies between events, even when the time intervals between events are variable.

Finally, transfer learning techniques can be applied to pre-train an LSTM network on a related task with similar time series data, and then fine-tune the network for the specific task at hand. This can allow the LSTM to better handle variable time intervals between events, as the pre-training can help the LSTM to learn general temporal dependencies that are common to many time series tasks.

In conclusion, handling variable time intervals between events is an important challenge in medical and clinical time series forecasting using RNNs and LSTMs. There are several approaches that have been proposed to address this challenge, including timestamp encoding techniques, custom LSTM architectures, and transfer learning. It is important to carefully consider the specific characteristics of the data and select the appropriate approach for the specific forecasting task at hand.

2.3.4 RNN Model Architectures

The architecture of RNNs can be modified in several ways to better model medical and clinical time series data. These modifications can include [25] [26] [27] [28]:

1. Stacking multiple layers of RNN cells: This can improve the model's ability to learn

complex temporal relationships between variables in the time series.

2. Using bi-directional RNNs: These RNNs process the input sequence in both forward and backward directions, providing the model with information about both past and future events in the time series.

3. Incorporating attention mechanisms: Attention mechanisms can be used to dynamically weight the importance of different input features, allowing the model to focus on the most important variables in the time series data.

4. Adding fully connected layers after the RNN: This can be used to capture complex relationships between the input variables and the output, providing the model with additional modeling power.

5. Using a specialized RNN architecture: There are several RNN architectures designed specifically for time series forecasting, including variants of LSTMs and GRUs that have been modified to better handle medical and clinical data.

2.4 Causal Inference on Trained Models

Once the various models have been tuned for desirable performance, causal inference can be performed to gain a better understanding of cause-and-effect relationship between the latent variables. The goal is to understand how changes in the input variables will affect the output time series and make predictions accordingly. To perform causal inference when the desired LSTM model is trained, one approach is to use the trained model to simulate the effect of a change in the input variables on the output time series. This can be done by inputting the time series data with the manipulated input variables into the LSTM model and observing the resulting changes in the output time series. The difference between the original and the manipulated output time series can then be used to infer causality between the input variables and the output time series.

In the context of more sophisticated methods, these are some of the other available options:

1. Granger causality [29]: This method tests the relationship between two time series, where one time series is said to cause the other if past values of the first time series are useful

in predicting future values of the second time series. In the context of LSTM time series forecast, this can be achieved by fitting an LSTM model with one time series as input and the other as output, and evaluating the model's performance using metrics such as R-squared.

2. Structural VAR models [30]: These models explicitly model the causal relationships between multiple time series, and can be used to identify the causality between a set of predictors and a target time series. A structural VAR model can be implemented by fitting an LSTM model to multiple time series inputs and a single time series output, and using the model's parameters to infer the causal relationships between the inputs and the target.

3. Causal impact analysis [31]: This method measures the effect of a specific event or intervention on a time series by comparing the forecasted values before and after the event. In the context of LSTM time series forecast, this can be implemented by fitting separate LSTM models before and after the event, and comparing the forecasted values from the two models to assess the causal impact of the event.

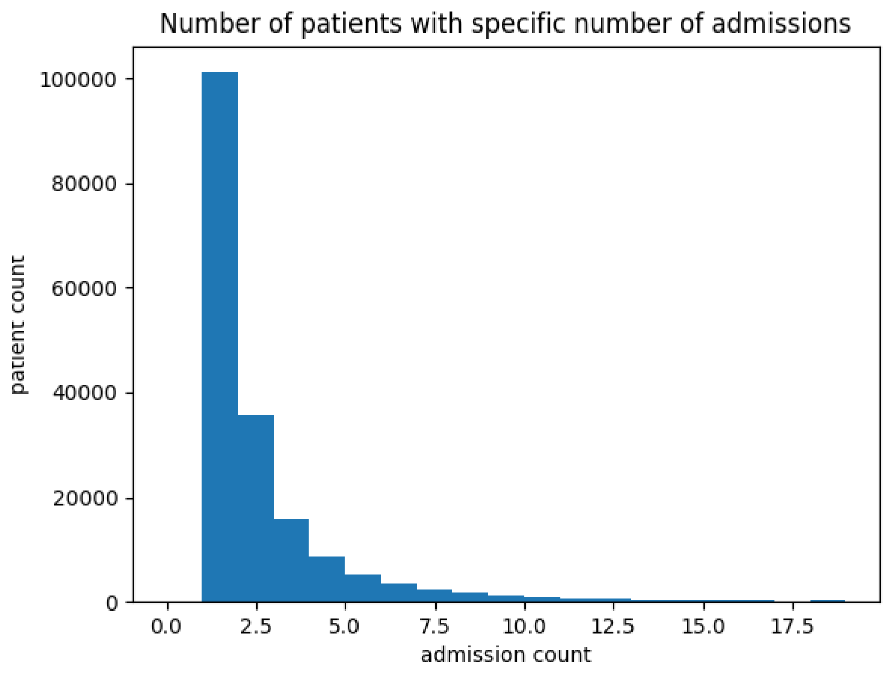
4. Dragon Net [32]: This method combines causal inference and deep learning techniques to estimate causal effects from observational data. Dragon Net consists of two components: the representation learning component, which uses a deep neural network to learn a suitable representation of the input data, and the causal inference component, which estimates treatment effects using the learned representation. In the context of LSTM time series forecast, Dragon Net can be applied by training a deep neural network, such as an LSTM, to learn a representation of the input time series data, and then using the causal inference component to estimate the causal effects of changes in the input variables on the output time series.

3 Datasets and Preprocessing

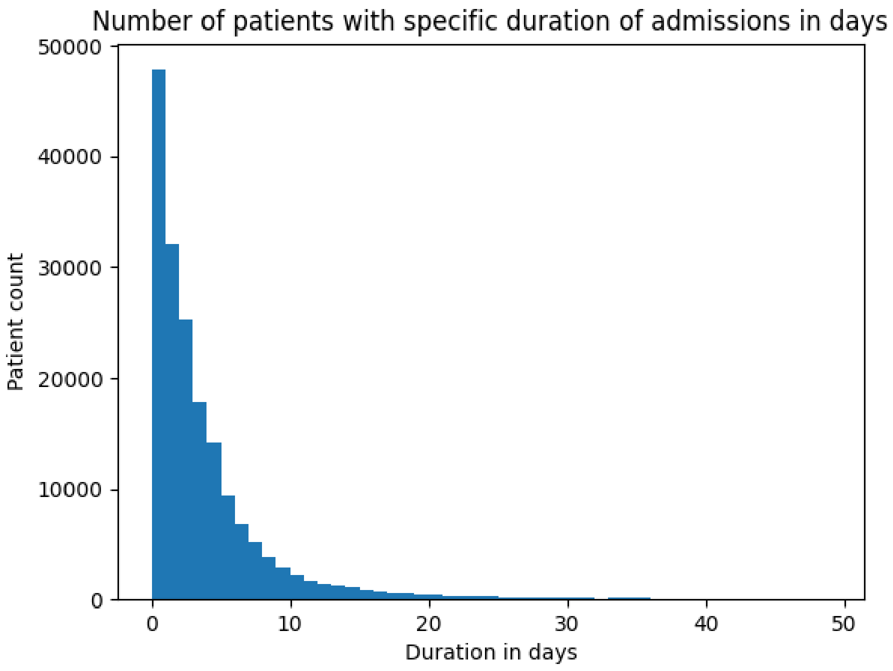
The MIMIC-IV [3] EHR data has been used to train the RNN for the task of multivariate time series forecasting. This de-identified dataset has diagnoses, lab events, input/output events, procedures, and prescriptions for each patient tracked for each visit over a period of time. MIMIC-IV provides this type of data for 40,000+ patients admitted to intensive care units at the Beth Israel Deaconess Medical Center. From this multivariate data, the goal is develop an appropriate time series forecast of the key latent variables. The following subsections will discuss the properties of the datasets and preprocessing required to get the desired outcomes.

3.1 Admissions

The admissions file within the MIMIC-IV dataset had information regarding 431,088 patients with the following distribution of admission counts as illustrated in 1a. From this, it can be seen that most patients (about 100,000) only have a single admission recorded into the ICU and very few have more than 5 admissions. As such, to keep the data consistent, only the first admission of each patient was preserved for training purposes. Furthermore, it can be observed from Figure 1b that majority of patients spend less than 48 hours in the ICU. As such, only the data for patients whose duration of stay is ≤ 48 hours is preserved. After doing this, the patient count in the dataframe is 82,592. This step is in alignment with the work of De Brouwer et al. [33] who holds the current #1 global rank for both the MSE and NegLL metric in terms of Multivariate Time Series Forecasting on the MIMIC-III dataset. Another reasoning for this is to limit the size of the overall preprocessed data to allow for faster training on the limited GPU environment. However, if it is deemed necessary, the elapsed time window will be increased to be ≥ 48 hours in the future.



(a) Admission Count Histogram



(b) Elapsed Time Historgram

Figure 1: Histograms Summarizing the Patients within the Admission File

3.2 Input/Output, Lab, Prescription Events

Having preprocessed the desired admission for each patient, a variety of features can be obtained across the various hospital/icu events. Namely, these are the input events, output events, lab events and prescription events. For each of these events, only the *top-k* most popular events have been chosen carefully to be the latent variables that categorize the current health condition of a patient. This technique of choosing the *top-k* most popular events is also derived from De Brouwer et al.'s [33] work on Multivariate Time Series Forecasting of EHR.

For this study, 82 variables were selected which are listed in Table 1. To standardize the units, any uncertain occurrences and outliers were removed that were outside of a 5 standard deviation interval. For models that needed to divide the time series into bins, measurements were divided into 60-minute intervals, resulting in 49 bins for a 48-hour period. If two observations fell in the same bin, they were either averaged or summed, depending on the nature of the observation. Lab measurements were averaged, while inputs, outputs, and prescription events were summed. This resulted in a total of 3,082,224 unique measurements across all patients, with an average of 145 measurements per patient over 48 hours. Lastly, it can be noted that patients with less than 50 measurements over the 48 hour period are also excluded from the dataset to allow for adequate data points for the model to learn from.

Lab measurements	**Input Events**	**Output Events**	**Prescriptions**
Hematocrit	Albumin 5%	Foley	Sodium Chloride 0.9% Flush
Hemoglobin	Dextrose 5%	Void	Acetaminophen
Platelet Count	Lorazepam (Ativan)	OR Urine	Docusate Sodium
White Blood Cells	Calcium Gluconate	Chest Tube #1	Heparin
RDW	Midazolam (Versed)	Pre-Admission	Senna
Red Blood Cells	Phenylephrine	Oral Gastric	Ondansetron
MCV	Furosemide (Lasix)	OR EBL	0.9% Sodium Chloride
MCHC	Hydralazine	Emesis	Bisacodyl
MCH	Norepinephrine	Jackson Pratt #1	Potassium Chloride
Glucose	Magnesium Sulfate	TF Residual	Magnesium Sulfate
Creatinine	Nitroglycerin	Nasogastric	
Urea Nitrogen	Insulin - Glargine	PACU Urine	
Potassium	Insulin - Humalog	Stool	
Sodium	Insulin - Regular	Condom Cath	
Chloride	Heparin Sodium	Straight Cath	
Bicarbonate	Morphine Sulfate		
Anion Gap	Potassium Chloride		
Magnesium	Packed Red Blood Cells		
Calcium, Total	Gastric Meds		
Phosphate	D5 1/2NS		
INR(PT)	LR		
PT	K Phos		
PTT	Solution		
pH	Sterile Water		
Estimated GFR (MDRD equation)	Metoprolol		
	Piggyback		
	OR Crystalloid Intake		
	OR Cell Saver Intake		
	PO Intake		
	GT Flush		
	KCL (Bolus)		
	Magnesium Sulfate (Bolus)		

Table 1: Retained longitudinal features for each event type

3.3 Data Merging to Create the 3D Tensor

In the previous step, the 82 features illustrated in Table 1 are extracted for each patient at each time step. This results in the creation of a 3D tensor of shape: ($num\_patients$, $num\_timesteps$, $num\_features$) where $num\_timesteps$ is fixed at 49 and $num\_features$ is fixed at 82 for all $N = num\_patients$ rows of the data tensor.

However, it is critical to note that not all of the 82 variables will be populated at each time step. In fact, on average only 2-3 variables have been populated at each time step for our dataset. This leads to the creation of a very sparse dataset which needs to be accounted for when deciding the RNN model architecture and training procedure. This will be discussed in depth in section 4.2. For now, the sparse dataset has the shape mentioned above with all the `NAN` entries listed as 0s.

After the merge is complete, *MinMaxScaler* is used to normalize the data before training an LSTM because the LSTM is a type of neural network that is sensitive to the scale of the input data. Normalizing the data helps to ensure that the model is not dominated by one variable with a much larger scale than the others, which could result in poor performance [34]. The MinMaxScaler transforms the data to a specific range, typically between 0 and 1, by subtracting the minimum value and dividing by the range. This helps the model to learn and converge faster and to produce better results, as the activation functions in the LSTM can operate more effectively in a small range of values. Lastly, the data is split into train, validation and test sets with a ratio of 70/20/10.

4 RNN Architectures and Code Structure

Recurrent Neural Networks (RNNs) are a class of neural networks that have shown remarkable success in modeling sequential data. There are three main types of RNNs: Vanilla RNN, Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM). Each type of RNN has its unique architecture and properties, which makes them suitable for different tasks.

- **Vanilla RNN:** This is the simplest form of an RNN, consisting of a feedforward structure with a hidden state. The hidden state is updated at each time step using the input and the previous hidden state. Due to their simple architecture, Vanilla RNNs are easy to implement and computationally efficient. However, they suffer from the vanishing and exploding gradient problems, making it challenging to learn long-term dependencies in the data.

- **GRU:** The Gated Recurrent Unit (GRU) is a more advanced type of RNN that aims to address the limitations of Vanilla RNNs. GRUs have two gates: the update gate and the reset gate. The update gate controls the amount of information from the previous hidden state that should be carried forward, while the reset gate controls the amount of information from the previous hidden state that should be combined with the current input. These gating mechanisms allow GRUs to capture longer-term dependencies in the data and alleviate the vanishing gradient problem to some extent.

- **LSTM:** Long Short-Term Memory (LSTM) networks are another advanced RNN architecture designed specifically to address the vanishing gradient problem. LSTMs have a memory cell that includes an input gate, a forget gate, and an output gate. These gates control the flow of information into, out of, and within the memory cell, allowing LSTMs to store information for longer durations. LSTMs have proven to be particularly effective in learning long-range dependencies and handling complex temporal tasks.

LSTMs are often preferred over Vanilla RNNs and GRUs for complex temporal tasks, such as multivariate time series forecasting, because of their ability to learn and retain long-range dependencies. The gating mechanisms in LSTMs allow them to effectively manage the flow of information, enabling them to model complex temporal patterns and capture relationships over extended periods. As a result, LSTMs can provide more accurate and stable predictions in tasks where understanding long-term dependencies is crucial.

As such, four LSTM neural network architectures were trained on the preprocessed MIMIC-IV dataset for the purpose of multivariate time series forecasting. The first one is the baseline model while the Seq2Seq model tries to effectively handle the data sparsity problem mentioned earlier. The last two add-on functionalities such as Teacher-Forcing and attention to the previous Seq2Seq model.

4.1 Baseline Forecaster LSTM

It is meant to be a simple "One-Directional" LSTM model which takes in `X_train` and `Y_train` of shape $(num\_patients, num\_timesteps, num\_features)$. Training LSTMs for multivariate time series forecasting typically involves using the shifted version of the input sequence `X_train` as the target `Y_train`. By time-shifting the target one step ahead, the LSTM model is trained to predict the next value in the sequence given the past values. This approach is known as supervised learning, where the model learns from labeled training data and attempts to generalize to unseen data. The target value is shifted one step ahead to provide the model with the correct context for prediction. This way, the LSTM model can learn the temporal dependencies and patterns in the input sequence and use that knowledge to make predictions on unseen data.

The model architecture of this baseline LSTM has four main components:

1. Input Layer: This layer accepts the input size. The input size is specified in the input_size argument in the `init` method.

2. LSTM Layer: This layer is implemented using the `torch.nn.LSTM module` from PyTorch. The LSTM layer accepts the input size, hidden size, number of layers, and a dropout rate as arguments. The hidden size is specified in the hidden_size argument in the `init` method. The number of layers is specified in the num_layers argument in the init method. The dropout rate is specified in the dropout_rate argument in the init method. The output of the LSTM layer has the shape (batch_size, seq_length, hidden_size).

3. Fully Connected Layer: This layer is implemented using the torch.nn.Linear module from PyTorch. The fully connected layer accepts the hidden size and output size as arguments. The output size is specified in the output_size argument in the `init` method. The output of the fully connected layer has the shape (batch_size, seq_length, output_size) representing the predicted time series for all time steps.

4.2 Seq2Seq Encoder-Decoder Model

The purpose of this model is to handle our sparse data mentioned in section 3.3. Handling missing or sparse data is an important aspect of building machine learning models. One way to handle sparse data is to use a technique called imputation. Imputation is the process of replacing missing values with estimated ones. There are several imputation methods available, such as mean imputation, median imputation, and multiple imputation. In the case of time series data, a common method is to use forward-fill or backward-fill imputation. Forward-fill imputation replaces missing values with the last known value in the time series, while backward-fill imputation replaces missing values with the next known value in the time series. As discussed in the Literature Review section 2.1.1, these techniques are not appropriate for the purpose of time series forecasting. Hence, a robust way to address this problem is by using a sequence-to-sequence (seq2seq) model. Seq2seq models are designed to handle sequential data, and can be trained to predict missing values in a time series [35].

The seq2seq model can be combined with a technique called ”masked loss” where we only calculate the loss function on the non-missing values and masking the missing values. This can be done by creating a custom loss function which takes the mask as input. One way to perform masked loss using the Mean Squared Error (MSE) loss function is to first create a binary mask from the input tensor, where all non-zero elements are assigned a value of 1 and all zero elements are assigned a value of 0. Then, you can use this mask to zero out the corresponding elements in the predicted output and the target before computing the MSE loss. The impact of using a masked loss function in training an LSTM for multivariate time series forecasting includes: improved handling of missing values, better model performance, and more robust predictions.

4.3 Seq2Seq with Teacher Forcing

The Seq2Seq with Teacher Forcing model builds upon the basic Seq2Seq model described in Section 4.2. This model aims to improve the learning of the decoder by using teacher forcing during training, with a specified probability. Teacher forcing is a method where the ground truth values are used as input to the decoder at each time step instead of using the model's own predictions. This approach can help the decoder learn more effectively, particularly during the early stages of training when its predictions may be poor.

The model architecture is similar to the basic Seq2Seq model, with the addition of the teacher forcing mechanism in the decoder part of the model. The model's components are as follows:

1. Encoder LSTM: Processes the input time series and generates hidden states.
2. Decoder LSTM: Produces the output time series, using the hidden states from the encoder and applying teacher forcing with a specified probability.
3. Fully Connected Layer: Transforms the hidden states of the decoder LSTM to the desired output shape.

4.4 Seq2Seq with Attention

The Seq2Seq with Attention model is another extension of the basic Seq2Seq model (Section 4.2), which incorporates an attention mechanism to improve the model's ability to capture long-range dependencies and handle missing values more effectively. The attention mechanism computes context vectors based on the encoder hidden states and the current decoder hidden state, and these context vectors are used as additional input to the decoder [1]. This approach allows the model to focus on different parts of the input sequence as it generates each output step, resulting in more accurate predictions.

The model architecture consists of the following components:

1. Encoder LSTM: Processes the input time series and generates hidden states.
2. Attention Mechanism: Computes context vectors based on the encoder hidden states and the current decoder hidden state.
3. Decoder LSTM: Produces the output time series using the concatenation of the context vectors and its input (which can be either ground truth or its own previous output, depending on the teacher forcing ratio).
4. Fully Connected Layer: Transforms the hidden states of the decoder LSTM to the desired output shape.

4.5 Model Selection

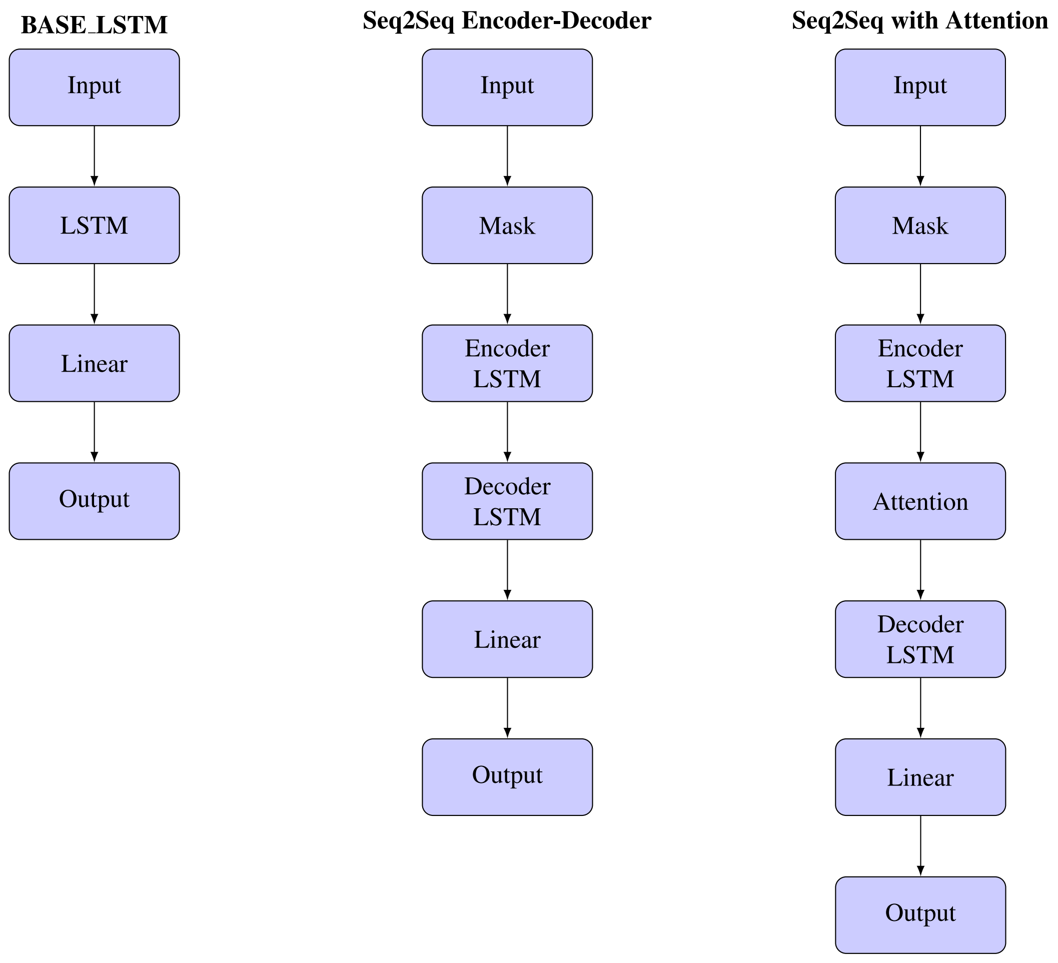


Figure 2: Model Architectures: Base-LSTM, Seq2Seq, and Seq2Seq with Attention

The above figure illustrates the model architectures talked about in the previous sections. Here, Seq2Seq with Teacher Forcing is not shown as it has the same architecture as the Seq2Seq Encoder-Decoder model with Teacher Forcing used as an added functionality within the Decoder LSTM. Furthermore, in the context of multivariate time series forecasting, we compare the following models: Baseline Forecaster LSTM, Seq2Seq Encoder-Decoder Model, Seq2Seq with Teacher Forcing, and the Seq2Seq with Attention model

1. **Complexity:** The Seq2Seq model and its variants (with Teacher Forcing and Attention) are more complex than the BASE_LSTM model because they contain both an encoder and a decoder LSTM layer. This added complexity allows these models to capture more intricate relationships between input and output sequences.

2. **Information flow:** In the BASE_LSTM model, the information flow is relatively simple, with the LSTM layer processing the input sequence and producing the output sequence directly. In the Seq2Seq models, the information flow is more involved, with the encoder LSTM capturing the input sequence's information and the decoder LSTM using this information to generate the output sequence. The attention mechanism in the Seq2Seq with Attention model further refines the information flow by selectively focusing on different parts of the input sequence.

3. **Masking:** The Seq2Seq model has an additional masking function that handles sparse input data by ignoring missing values (zeros) when processing the input sequence. This helps the model focus on the non-zero values and make more accurate predictions. The BASE_LSTM model does not have this functionality.

4. **Handling sparsity:** The Seq2Seq models are better suited for handling sparse input data because of the masking function. The BASE_LSTM model does not have a built-in mechanism to handle sparse data and may require additional preprocessing to fill in missing values.

5. **Prediction granularity:** In a multivariate time series forecasting scenario, both the BASE_LSTM and Seq2Seq models can predict all variables at once, as their output size is the same as the input size. However, the Seq2Seq models, especially with attention mechanism, may be better suited for predicting more complex relationships between input and output sequences due to their encoder-decoder architecture and selective focus

on input data.

6. **Teacher Forcing:** The Seq2Seq with Teacher Forcing model introduces an additional technique that helps improve the learning of the decoder by using the ground truth values from the previous time step as input during training instead of using the predicted values. This can lead to faster convergence and better performance.

In theory, both the BASE_LSTM and Seq2Seq models, along with their variants, can be used for multivariate time series forecasting. The Seq2Seq models are more complex and better suited for handling sparse input data and capturing more intricate relationships between input and output sequences However, before making a definitive model selection, the performance was compared in practice to several baseline multivariate time series forecasting tasks. This was done on a subset of the preprocessed sparse MIMIC-IV dataset and the results are summarized below.

	0.29 Mortality Rate	**0.36 Mortality Rate**	**0.40 Mortality Rate**
BASE_LSTM RMSE	0.049458612	0.048404146	0.048691675
Seq2Seq RMSE	0.050548963	0.050244108	0.04909417
Seq2Seq + Teacher Forcing RMSE	0.495781231	0.049736734	0.049238052
Seq2Seq + Attention RMSE	0.050014589	0.049886066	0.049589423

Table 2: Comparison of the Time Series Forecasting Performance across the Different LSTM Architectures

Based on the results above, we justify the choice of the Baseline Forecaster LSTM based on Occam's razor principle. Occam's razor suggests that, among competing hypotheses, the simplest one should be selected, given that it can adequately explain the observed data. In our case, all models have an RMSE of approximately **0.05** even when the mortality rate in the dataset varies meaning the performance is comparable. The Baseline Forecaster LSTM is simpler than the Seq2Seq Encoder-Decoder Model, and in many cases, it is sufficient for capturing the complex temporal patterns and dependencies in the data. Consequently, the Baseline Forecaster LSTM is computationally more efficient and easier to train, while still providing satisfactory results. Thus, according to Occam's razor, we prefer the Baseline Forecaster LSTM for multivariate time series forecasting tasks when it can achieve comparable performances to the various Seq2Seq Encoder-Decoder Models.

4.6 Overview of the End-to-End Framework

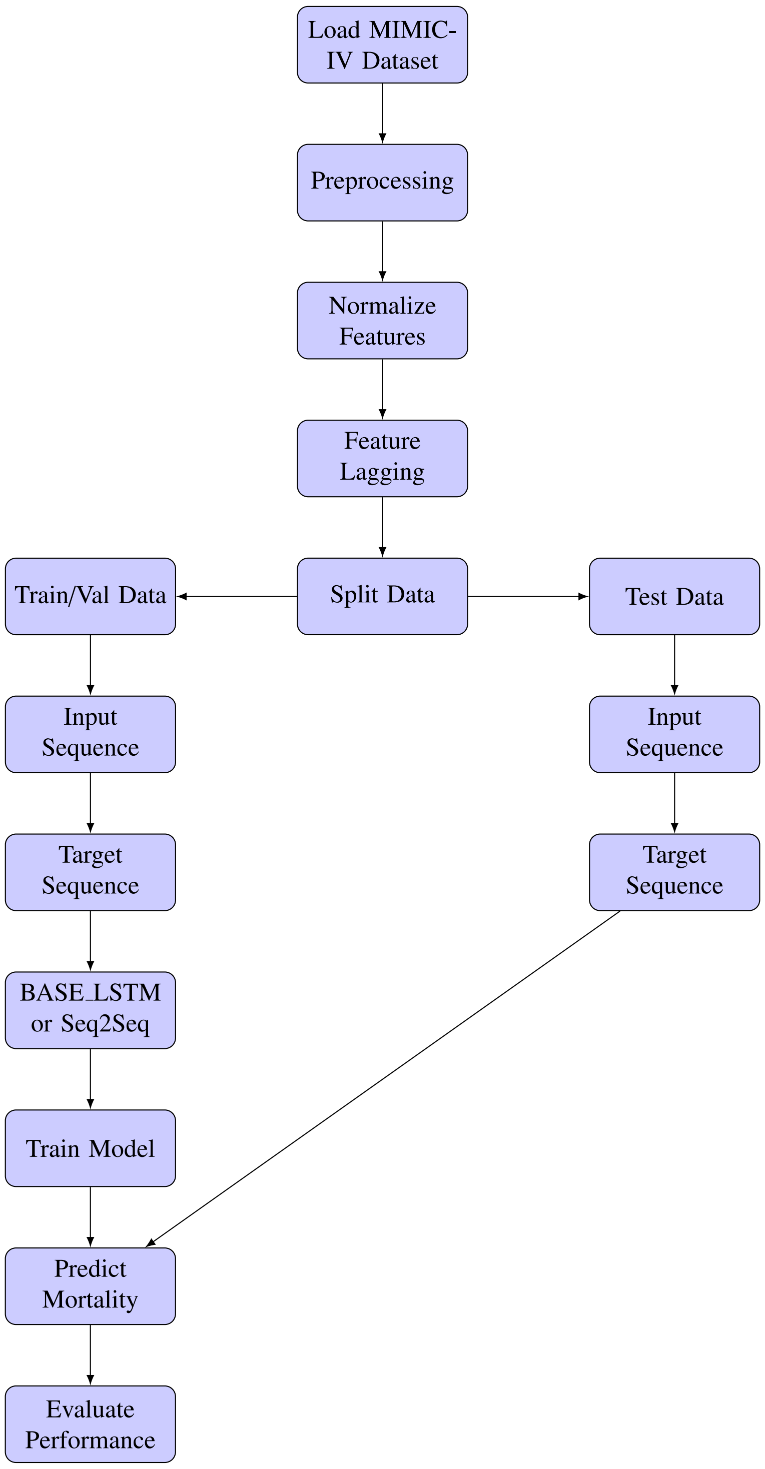


Figure 3: Flowchart of the End-to-End Code Base

In this flowchart, we depict the process of multivariate time series forecasting using our various neural network models: the `BASE_LSTM` model, and the `Seq2Seq Encoder-Decoder` variant models. Both models are designed to predict latent variables characterizing a patient across all time steps using the MIMIC-IV dataset. As such, the multivariate time series forecasting process involves the following steps:

1. **Data Preprocessing:** The MIMIC-IV dataset is preprocessed, which includes feature scaling, handling missing values, and creating lagged features to capture temporal dependencies.

2. **Model Training:** The LSTM/Seq2Seq model is trained with an input size of 83, a hidden size of 256, a depth of 3 layers, an output size of 83, and a dropout rate of 0.2. The model is trained using the Adam optimizer with a learning rate of *1e-3* and weight decay of *1e-6* for 50 epochs. Early stopping is enabled to stop the training if there are more than 5 epochs with no improvement in validation loss, helping prevent overfitting and saving time. Here the period of time, we wait before terminating (5 epochs) is called **patience** and is an additional hyperparameter that can be further tuned.

3. **Model Evaluation:** After training, the models are evaluated using various metrics, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and the coefficient of determination (R^2).

4. **Time Series Forecast:** The trained models are used to predict the value of all 83 features including mortality for the next time step. This is done across the entire time horizon. For the mortality prediction, a threshold is applied to determine the probability of mortality for each patient and the corresponding accuracy.

5. **Performance Evaluation:** The models' performance in predicting mortality and all other features is assessed using metrics such as accuracy, Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

In summary, the end-to-end process involves data preprocessing, model training, evaluation, mortality prediction, and performance assessment using various metrics to ensure accurate multivariate time series forecasting and reliable mortality prediction.

5 Results and Limitations

In the Results section, we first present the findings from our experiments on the MIMIC-IV dataset. We have chosen the Base LSTM model for multivariate time series forecasting based on Occam's razor principle outlined previously. We conducted two sets of experiments using the MIMIC-IV dataset: (1) Base LSTM without mortality, where we predict feature values in the next timestep for all 82 latent variables/features, and (2) Base LSTM with mortality, where we perform the same task as before but also include mortality prediction across all timesteps as the 83rd feature.

After evaluating the model's performance on the MIMIC-IV dataset, we test the limitations of the model by examining its robustness to random noise and sensitivity to dataset size. To do this, we perform additional experiments on a synthetic dataset, which allows us to explore the model's performance under edge cases. In the Limitations section, we will discuss the observed limitations in the model's performance on the MIMIC-IV dataset experiments and analyze the model's robustness to random noise and sensitivity to dataset size based on the synthetic dataset experiments. Additionally, we will address any other limitations discovered during the analysis of the model's performance across various experimental settings. The above structure is further illustrated in Figure 4.

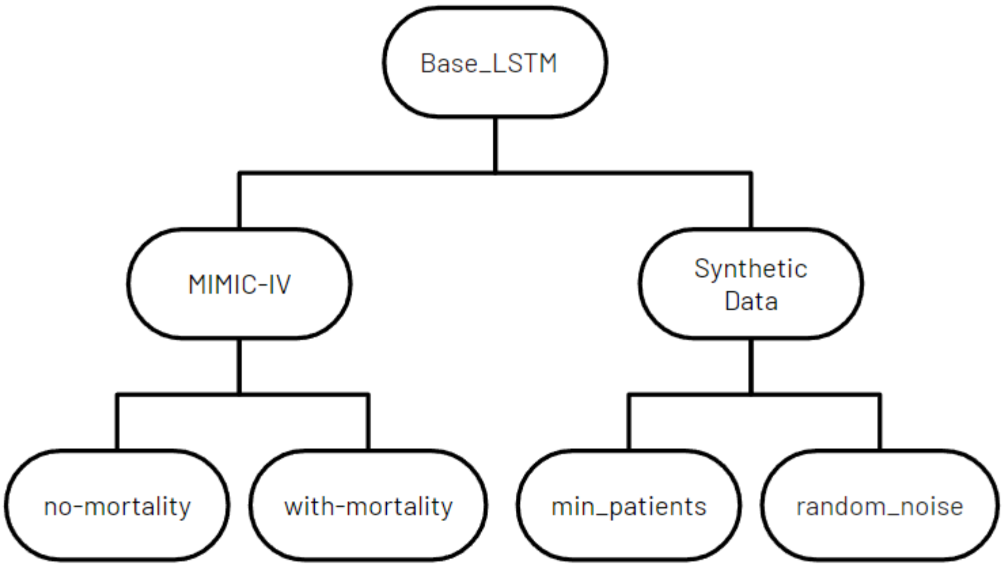


Figure 4: Summary of Experiments

5.1 Multivariate Time Series Forecasting on MIMIC-IV

5.1.1 Forecasting without Mortality

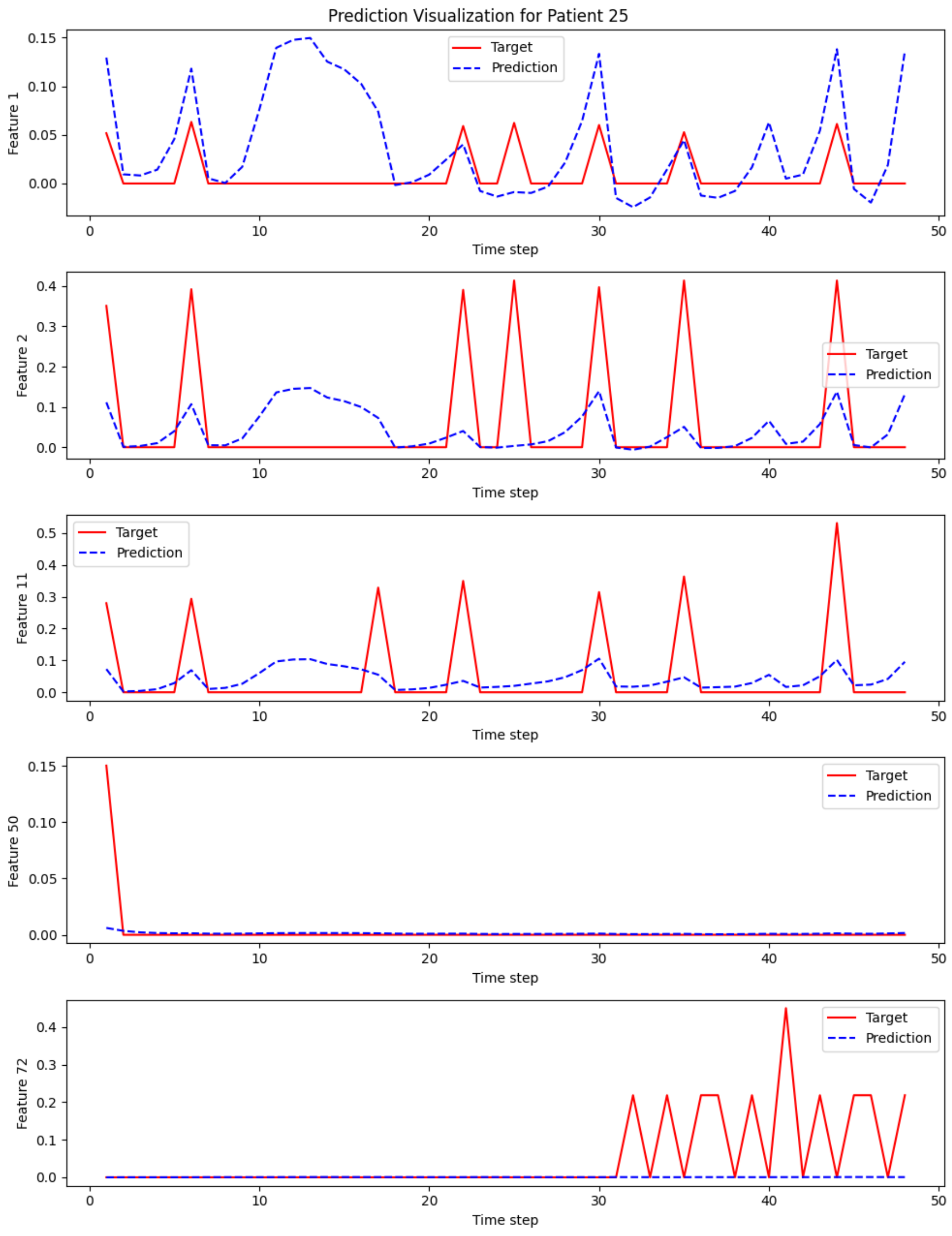


Figure 5: Forecasting Visualization without Mortality

Figure 5 presents the predictive performance of the LSTM model for a small, randomly chosen set of features (1, 2, 11, 50, and 72) across all time steps. This visualization aims to illustrate the model's predictive capability across various features, which can be generalized to the remaining features in the dataset (a total of 82 features).

As observed in Figure 5, the LSTM model demonstrates a satisfactory ability to capture the underlying patterns in the data and closely follows the trends. The root mean square error (RMSE) of the model's performance across several runs is approximately **0.05**, which supports the claim that the model generally performs well in predicting the trends of most features. However, it is important to note that the model does not perfectly capture the patterns for all features. For instance, the LSTM model fails to predict the trend for `feature_72`, as evidenced by the flat line in the figure.

Despite the model's limitations, it still manages to capture the general progression of the trends and can provide valuable insights for predictions one time step ahead. This performance highlights the potential of the LSTM model to contribute meaningfully to forecasting tasks in the context of the given dataset.

5.1.2 Forecasting with Mortality

This section builds on the analysis from the previous section 5.1.1, where the LSTM model demonstrated satisfactory performance in capturing underlying patterns and following trends in the data. The new addition is the inclusion of an extra (83rd) variable to predict patient mortality. The mortality label is appended to the final timestep in the training data, with each patient initially having a mortality label of 0 (alive). Their corresponding mortality value (0 for alive, 1 for dead) is added to the last timestep, enabling the model to learn trends that differentiate dead and alive patients at the end of a 48-hour window in an ICU.

As before, the model makes predictions one timestep into the future. It is worth noting that this experiment was conducted on datasets with varying mortality rates ranging from 0.3 to 0.5. The summarized results in Figures 6 and 7 are based on a dataset with a mortality rate of 0.36.

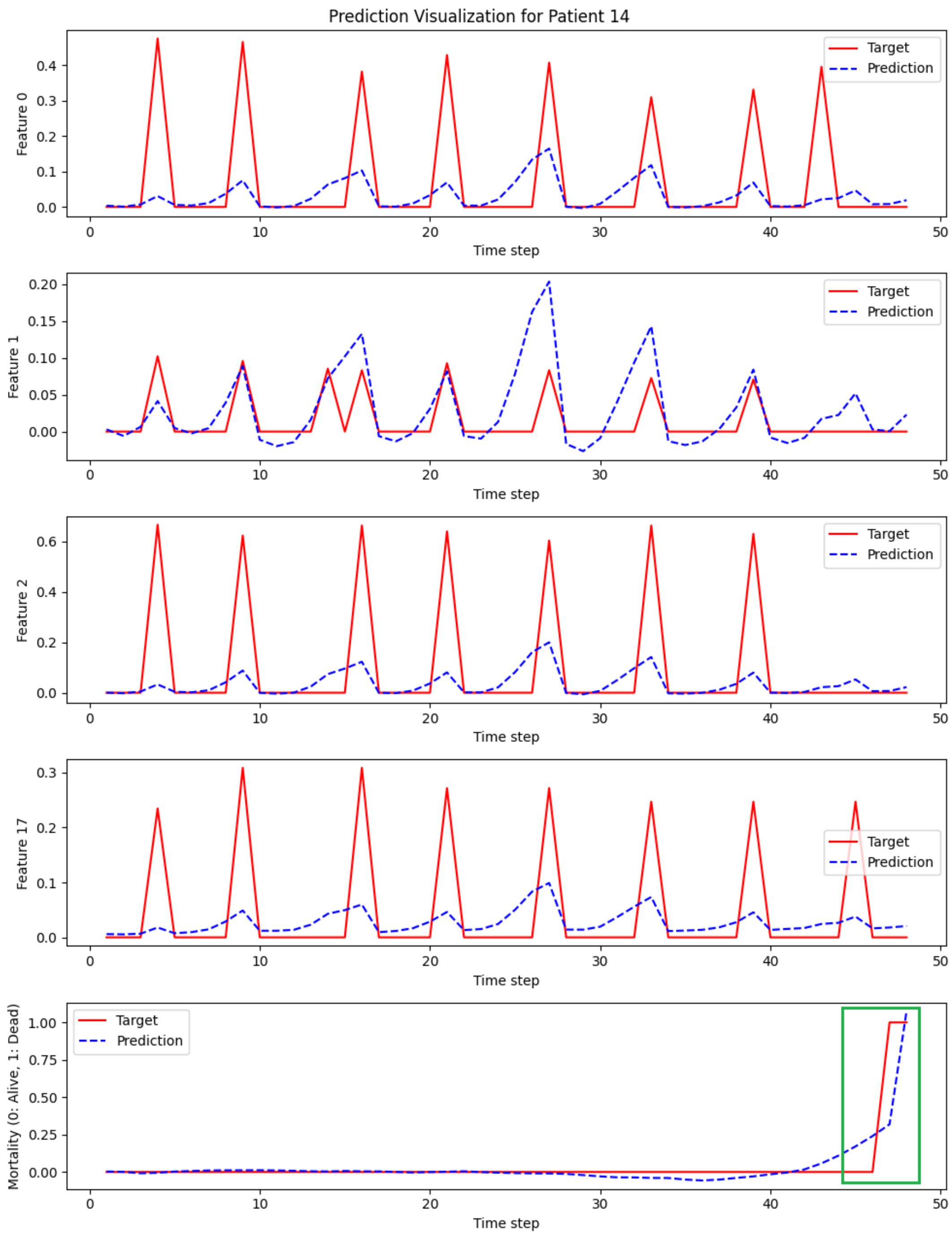


Figure 6: Forecasting Visualization for Mortality = 1

Figure 6 demonstrates the model's accuracy in predicting mortality of 1 (dead) for a specific patient. This accuracy is highlighted by the green box on the final feature, where the dotted blue leading edge tracks the red target. Similarly, Figure 7 shows the model's ability to accurately predict mortality of 0 (alive), as the final feature remains close to 0 throughout the entire time series. In this case, values consistently remain below 0.04, indicating the model's high confidence in predicting mortality = 0.

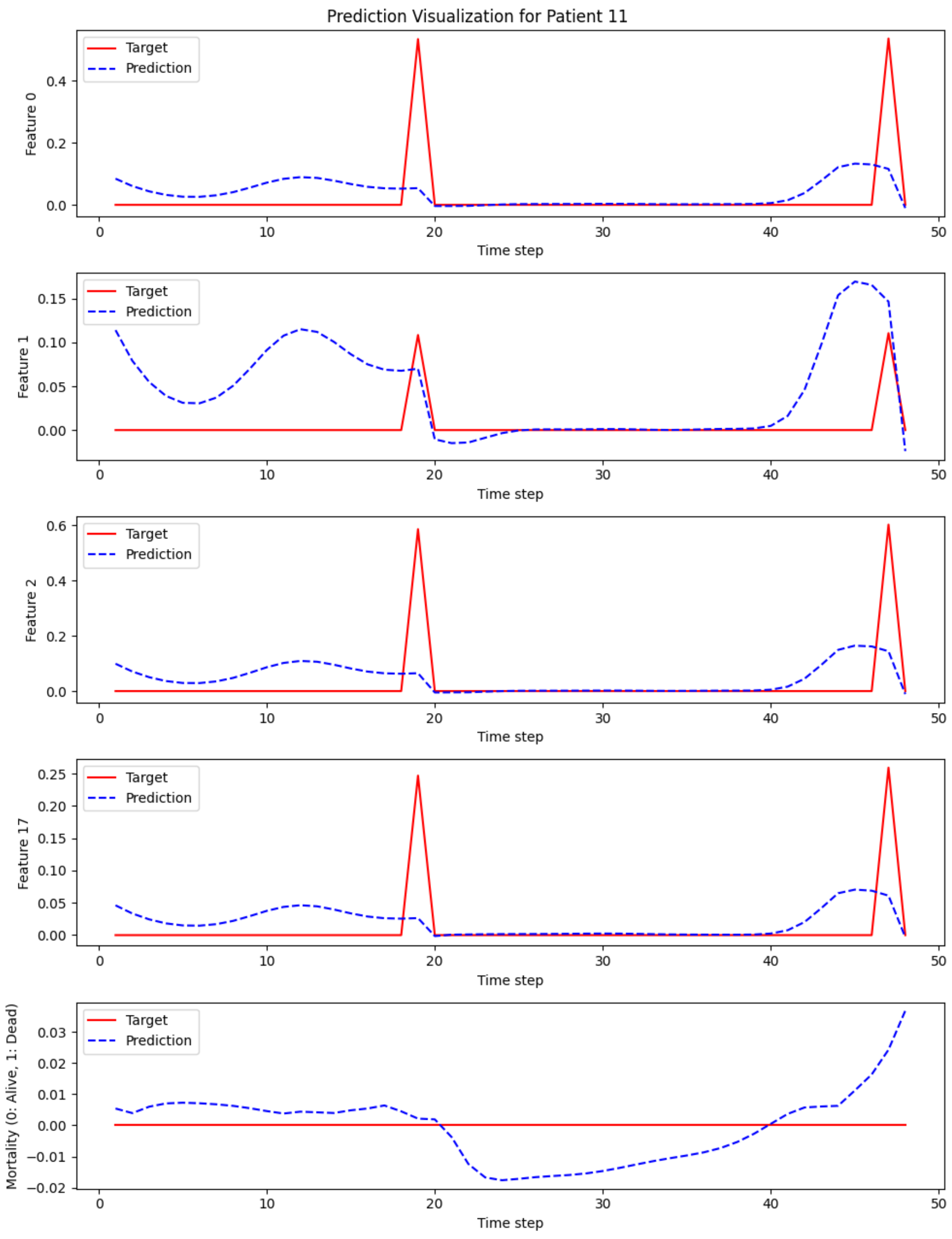


Figure 7: Forecasting Visualization for Mortality = 0

These figures serve as strong evidence of the LSTM's ability to capture high-dimensional, complex temporal data to make predictions about how various latent variables influence patient mortality. The model demonstrates an understanding of how changes in health variables (features 1-82) affect patient mortality (feature 83), suggesting potential applicability in continuous health monitoring use cases, especially in ICUs where this type of insight is highly valuable. However, before this model can be employed in practical settings, it is crucial to refine its performance further. While the current results are promising, there is no perfect overlap between predictions and actual data, indicating room for improvement. The goal is to enhance the system's predictive performance before it can be considered usable. Several limitations, which make this task challenging, are listed in section 5.3.

To complete this section, we can also compare the mortality accuracy percentage of the LSTM model across several mortality rates to assess performance. This was done by thresholding the 83rd feature so that any value under 0.1 was categorized to be alive and anything over 0.8 was categorized to be dead. The upper bound is very lenient as of now but with further improvements to this model, we are hoping to be closer to 0.95 mark to get a more reliable mortality prediction.

	0.29 Mortality Rate	**0.36 Mortality Rate**	**0.40 Mortality Rate**
Total Patients in Test Set	3904	3904	3904
predictions_mortality >0.8	612	798	853
targets_mortality >0.8	1132	1405	1562
predictions_mortality <0.1:	182458	181903	180658
targets_mortality <0.1	186235	184554	183786
Accuracy	0.74	**0.81**	0.77

Table 3: Comparison of the Mortality Accuracy Performance across the Different Mortality Rates

The results look promising with prediction accuracy in the mid to high 70s. But this is with lenient bounds and for usability in a practical setting, one would require confidence bounds closer to either the 95% mark or 99% mark. This is something that can be improved with future work on the model and other feature engineering techniques.

5.2 Multivariate Time Series Forecasting on Synthetic Data

After evaluating the model's performance on the MIMIC-IV dataset, we test the limitations of the model by examining its robustness to random noise and sensitivity to dataset size. To do this, we perform additional experiments on a synthetic dataset, which allows us to explore the model's performance under edge cases. In the Limitations section, we will discuss the observed limitations in the model's performance on the MIMIC-IV dataset experiments and analyze the model's robustness to random noise and sensitivity to dataset size based on the synthetic dataset experiments.

5.2.1 Determining Effect of Random Noise

The first synthetic dataset, `correlation1_2_rest_0`, allows us to observe if the LSTM can capture the correlation between the first two features without any noise in the other features. In this experiment, the rest of the features are set to 0, simulating a clean dataset with no noise. By analyzing the model's performance on this dataset, we can determine if the LSTM is capable of identifying patterns in the data when the noise is minimized.

On the other hand, the second dataset, `correlation1_2_rest_random`, is designed to test the model's performance under maximum noise. All features from the third feature onwards are assigned random values between 0 and 1. This experiment allows us to assess the model's robustness to random noise and its ability to capture existing patterns despite the presence of substantial noise.

Lastly, the third dataset, `correlation1_2_some_random`, provides a controlled noise environment. In this case, only some of the features (ex: 10, 30, etc) from the third feature onwards are assigned random values, while the rest are set to 0. This scenario allows us to explore the LSTM's performance when certain latent variables have noisy data and assess if the noisy data affects the model's ability to capture existing patterns in the data that it otherwise would have been able to identify without the noise.

In each of the synthetic dataset configurations, we investigate the LSTM model's performance by comparing its predictions with the ground truth values for each configuration. It is to be noted that the dataset size had been preserved to be the same as the one we used for MIMIC-IV [39481, 49, 83] previously to assess the LSTM performance under similar conditions but with added noise. This allows us to analyze the model's robustness to random noise, as well as its ability to capture the underlying relationships between features and forecast future time steps based on these relationships.

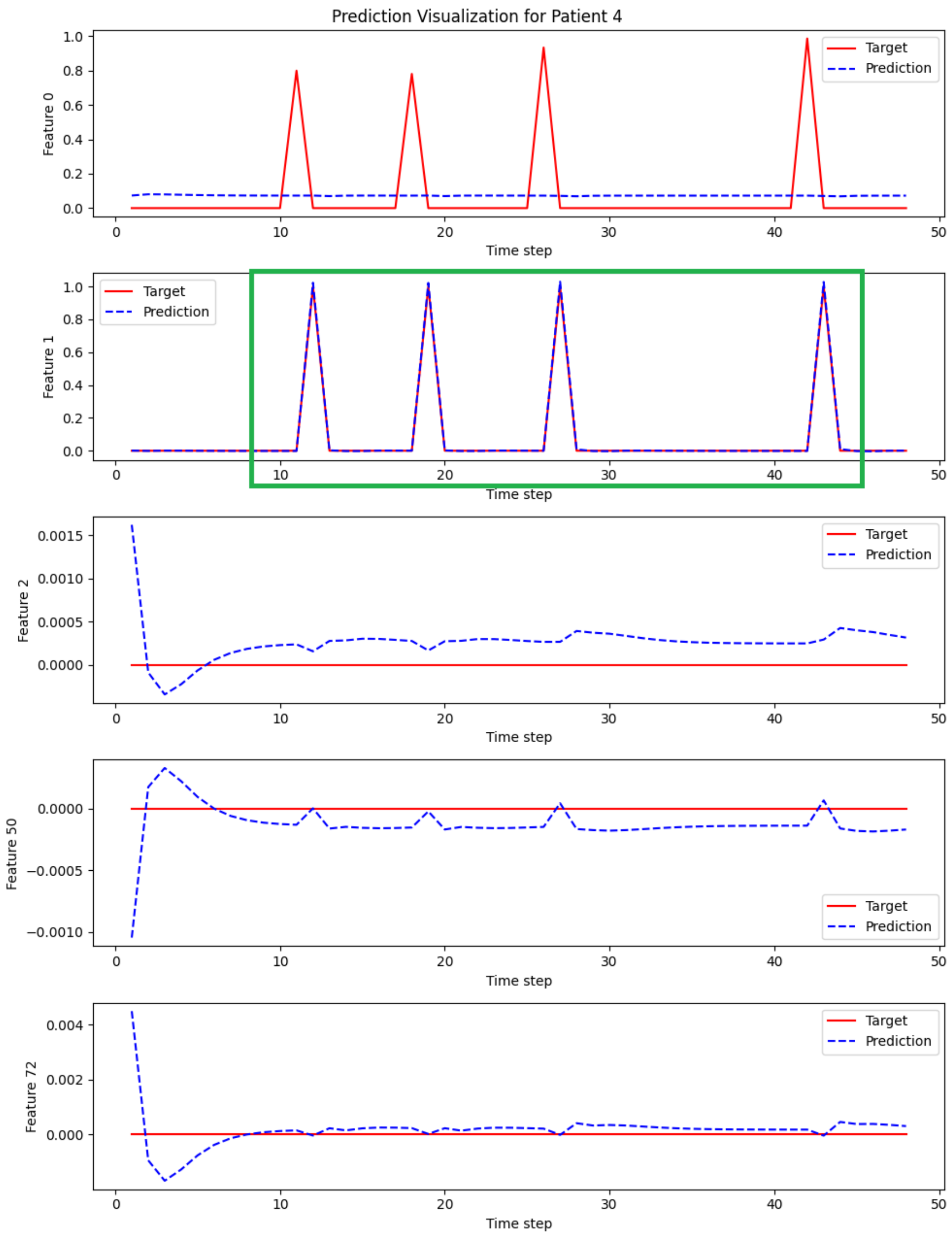


Figure 8: Features 1 and 2 are correlated, rest are 0 (No Noise)

Figure 8 illustrates a scenario where the LSTM successfully captures the correlation between `feature_0` and `feature_1` (boxed in green) when all other features (2 - 83) are 0. The correlation between `feature_0` and `feature_1` is described as follows : `feature_0` will have a value between 0.5 and 1 with a probability of 30%, and a value of 0 with a probability of 70%. If `feature_0` > 0.5 at any time step, `feature_1` will be 1 in the next timestep. The

LSTM model accurately captures this pattern, indicating that it can identify structure in the data when there is no noise present.

Next, we evaluate how the LSTM performs when all other features (2 - 83) are populated with random noise, as shown in Figure 9.

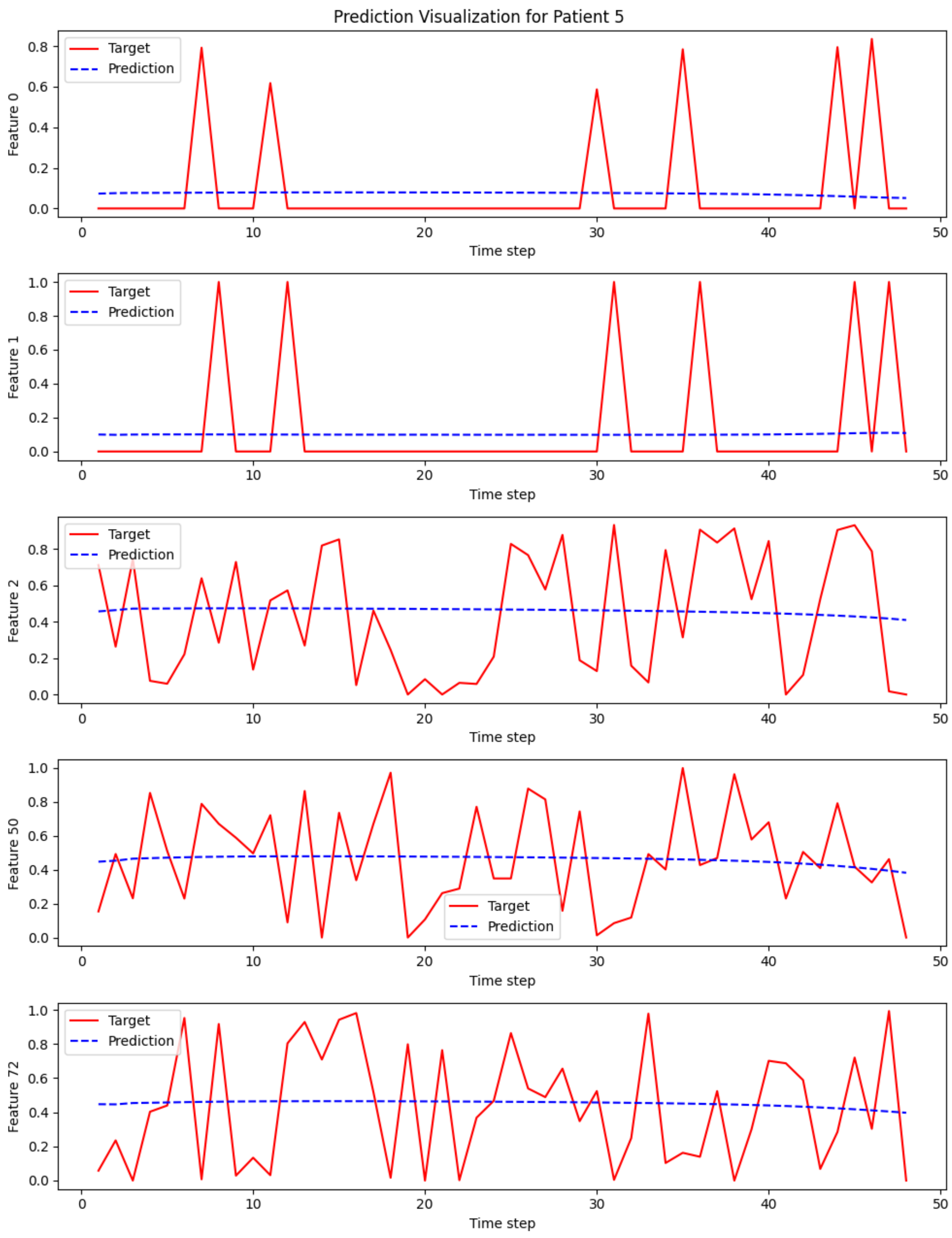


Figure 9: Features 1 and 2 are correlated but rest are random noise

As we see in figure 9, the presence of random noise disrupts the LSTM's ability to capture the correlation between `feature_0` and `feature_1` because `feature_1` is a flat line now. Also, model, cannot model the noise which makes sense as it is randomly generated via `np.random.uniform(0, 1)` and has no structure to it. But the key here is that our correlation between `feature_0` and `feature_1` is not captured anymore. Thus we can run experiments to determine what is the max amount of noise we can add before the LSTM fails to capture the correlation. This is done below by reducing the number of remaining features that have random noise across all time steps. To elaborate, in the extreme case above, we had that all features except `feature_0` and `feature_1` had random noise. This means 83-2 = 81 remaining features had random noise. The experiment involved gradually decreasing this number of remaining feature with random noise from the starting point of 81 until a value was found where the LSTM continued to track the correlation described here 5.2.1. While this was done, the features without noise are set to 0. The results for this are captured in the figures below.

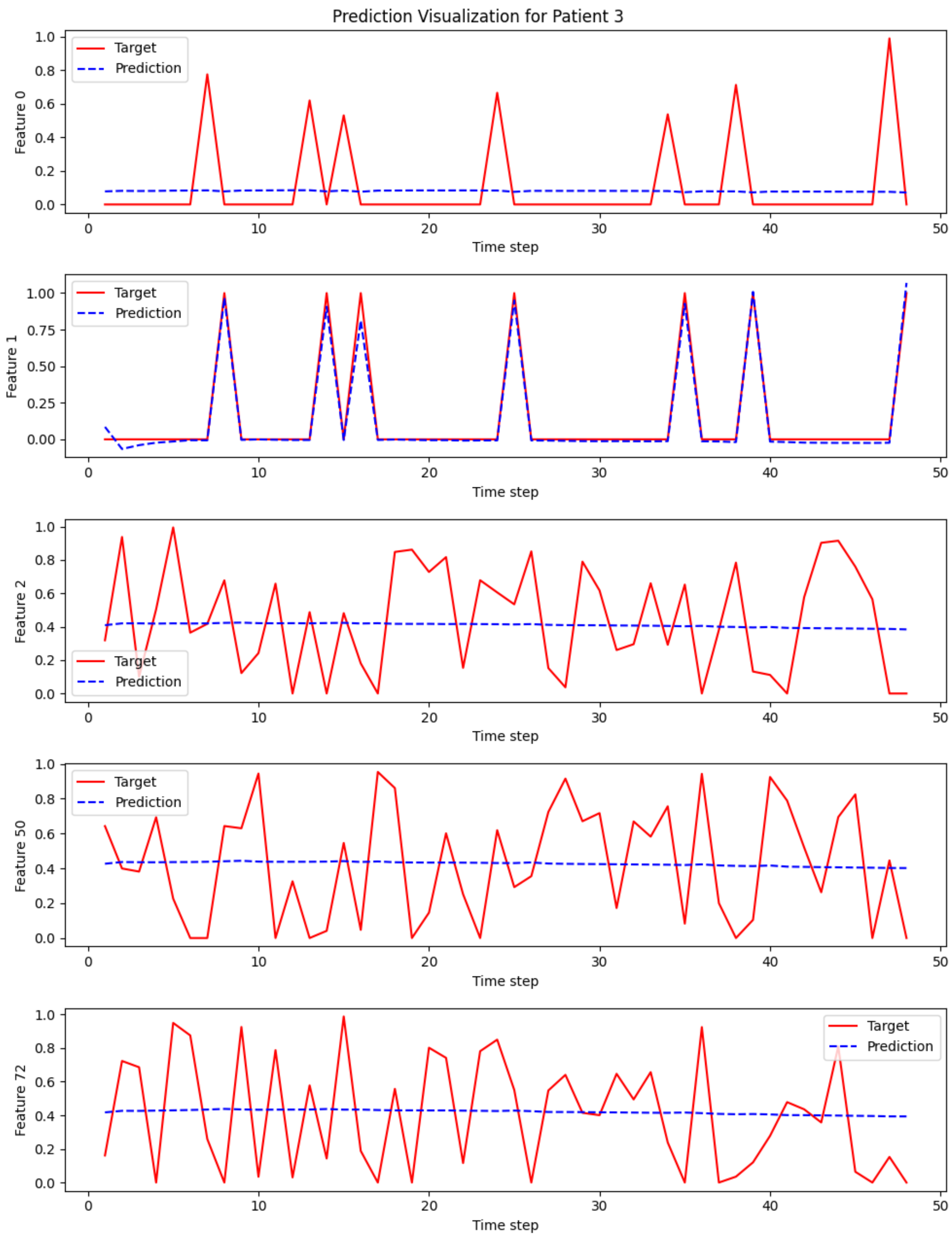


Figure 10: Features 1 and 2 are correlated but **68** of the remaining features are random noise

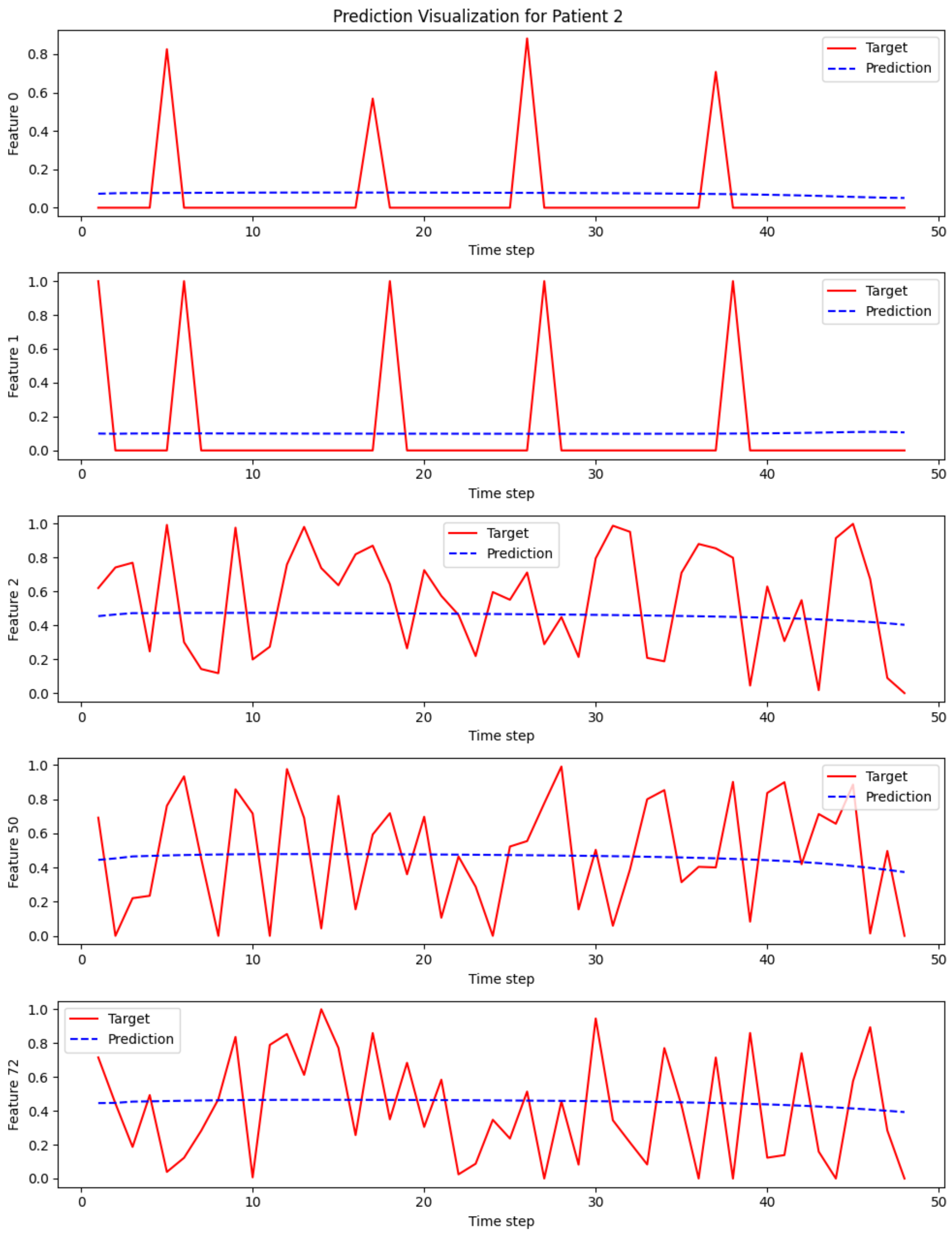


Figure 11: Features 1 and 2 are correlated but **75** of the remaining features are random noise

Comparing Figures 10 and 11, it can be determined that the breaking point lies between 68 and 75 random features. This indicates that the LSTM model can successfully capture the correlation between `feature_0` and `feature_1` despite the presence of noise in approximately 70 features out of 83 total features. This demonstrates the robustness of the model under severe noise, making it particularly useful in healthcare settings where measurements are often noisy and sparse.

In the context of the MIMIC-IV dataset with 39,481 patients, our findings suggest that the LSTM model can effectively capture correlations and patterns in the data even when a substantial portion of the features have noise. This is a testament to the robustness of the model and its potential applicability in real-world healthcare settings.

It is important to note that the breaking point determined in this experiment is specific to the dataset and the chosen LSTM model. The performance of the LSTM may vary depending on the complexity of the patterns, the architecture of the model, and the level of noise present in the data. Further experiments can be conducted to investigate the effect of varying noise levels on model performance and to identify the most suitable LSTM model for different healthcare datasets.

In conclusion, our analysis demonstrates that the LSTM model can effectively capture correlations and patterns in the data even under the presence of a significant amount of noise. This property is highly beneficial in healthcare settings where data is often noisy and sparse, making the LSTM model a valuable tool for analyzing and predicting patient outcomes.

5.2.2 Determining Effect of Dataset Size

In this experiment, we aim to analyze the effect of dataset size on the performance of an LSTM model. We use the same MIMIC-IV dataset as before with dimensions [39481, 49, 83]. We vary the dataset size in terms of the ratio of the original dataset, ranging from 10% to 100% in increments of 10%. This allows us to observe how the model's performance is influenced by the amount of available training data. The model's performance was evaluated using four metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2) score. The results of this experiment are presented below:

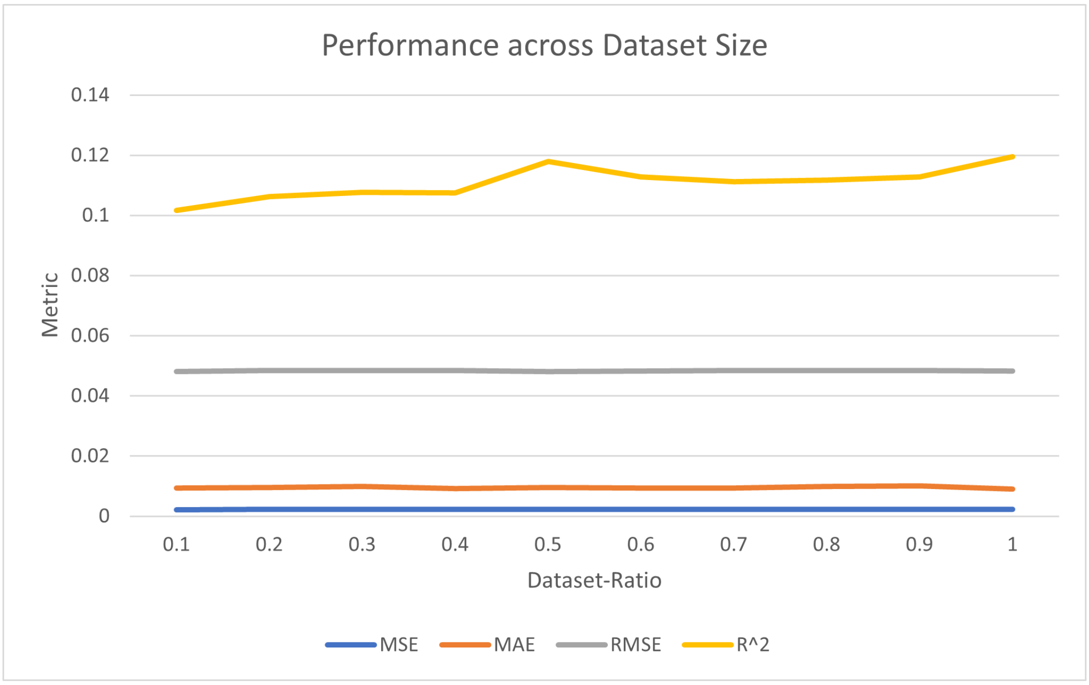


Figure 12: LSTM Model Performance Across Varying Dataset Sizes

From the results, it can be observed that as the dataset size increases, the model's performance does not show a consistent improvement or deterioration. The MSE, MAE, and RMSE values fluctuate within a relatively narrow range, indicating that the model's performance remains relatively stable across different dataset sizes.

The R^2 score, which measures the proportion of the variance in the target variable that is predictable from the input features, shows a mild increase as the dataset size increases. This suggests that the model is better able to capture the underlying relationships between features and forecast future time steps based on these relationships when trained on larger datasets.

However, the overall improvement in R^2 score from 0.1017 (10% dataset size) to 0.1196 (100% dataset size) is relatively small. This implies that the LSTM model is relatively robust to variations in dataset size and can still capture the underlying relationships between features with a smaller amount of training data.

In conclusion, the LSTM model's performance remains relatively stable across different dataset sizes, with only a slight increase in R^2 score as the dataset size increases. This suggests that the model can effectively capture underlying relationships between features and forecast future time steps based on these relationships even when trained on smaller datasets.

5.3 Limitations and Qualifications

Several limitations of this work can be attributed to the characteristics of the dataset used, while others are more general. In this section, we will discuss the dataset-related limitations that should be considered when interpreting the results, as well as the limitations of the current LSTM Model.

5.3.1 Limitations of the MIMIC-IV Dataset and current LSTM Model

The MIMIC-IV dataset, despite its usefulness, has certain drawbacks that are worth noting in the context of the conducted experiments. Additionally, the model that we have trained and obtained the previous results for is not perfect. There is still room for improvement here as well. Overall, there are five main issues that we can address:

1. MIMIC-IV contains prescription records, but the actual administration of medications to patients is not guaranteed. This discrepancy introduces noise in the data, which is difficult to address.

2. Medication prescriptions are timestamped to the nearest day, leading to ambiguity in the true order of prescriptions, particularly for short-duration admissions. Similar limitations apply to other latent variables used in the study, such as lab measurements (e.g., hemoglobin, glucose, sodium levels), vital signs, and input/output events. The granularity of these measurements may not be sufficient for the desired predictive performance, especially when making short-term predictions.

3. The MIMIC dataset is sparse, as not all features are populated across all timesteps. As such, our current modeling techniques are not sufficient, as we see that the performance is not very accurate. The targets and predictions for most of the experiments don't fully overlap as of now, and there is still room for improvement in the predictive performance either via additional modeling techniques applied to LSTMs or using an alternate framework.

4. Mortality and health predictions are currently being done for the next timestep, which is 1 hour in our case due to our time binning. However, in a practical case such as the ICU, it might be beneficial to make predictions further into the future to allow more time for preventive action. This would require more sophisticated modeling and the incorporation

of additional latent variables from the MIMIC-IV dataset to obtain a more comprehensive picture.

5. The LSTM model may not be able to fully capture the complexities and non-linear relationships between the features in the dataset. Additionally, the hyperparameters of the model, such as the number of hidden layers and units, learning rate, and activation functions, can significantly impact the model's performance. A more extensive search for the optimal combination of hyperparameters or the use of alternative deep learning architectures, such as Transformer models or attention mechanisms, may improve the predictive performance.

6 Future Work

In future work, we aim to focus on causal inference to deepen our understanding of the cause-and-effect relationships between latent variables in multivariate time series forecasting of patients' health progression in the ICU. Causal inference is a critical component in Health AI, as it enables us to make informed decisions and predictions by revealing the underlying relationships between variables. Exploring various methods for causal inference will allow us to identify the most effective techniques for this specific project.

Several methods for causal inference could be explored, such as Granger causality, propensity score matching, and Dragon-Net. Granger causality is a statistical hypothesis test that determines whether one time series can predict another, while propensity score matching is a technique used to estimate the causal effects of a treatment by accounting for confounding variables. Both methods can provide valuable insights into the relationships between variables, but they may not be the most appropriate choice for our LSTM-based model trained on the MIMIC-IV dataset.

Dragon-Net, on the other hand, stands out as a superior choice due to its ability to adapt neural networks for the estimation of treatment effects, making it highly suitable for our LSTM-based model. Dragon-Net is a cutting-edge method that combines the power of deep learning with advanced causal inference techniques, enabling more accurate and reliable estimation of treatment effects. Specifically, Dragon-Net utilizes a two-step approach [32]: first, it employs representation learning to capture complex and non-linear relationships in the data, and second, it uses these learned representations to estimate the causal effects of interventions.

The strengths of Dragon-Net lie in its flexibility, scalability, and ability to handle high-dimensional data. By leveraging the representational power of deep learning, Dragon-Net can capture intricate relationships between variables, even in the presence of confounding factors. Moreover, its scalability allows it to handle large datasets, such as the MIMIC-IV dataset, making it well-suited for this project.

Implementing Dragon-Net on a trained LSTM model involves adapting the model architecture to estimate the treatment effects. This implementation allows us to leverage the strengths of both the LSTM model, which captures complex temporal patterns, and Dragon-Net, which provides accurate causal inference. This combination has the potential to significantly improve our understanding of patients' health progression in the ICU and optimize treatment plans.

In addition to Dragon-Net, other causal inference techniques could also be explored, such as Bayesian networks, instrumental variables, or difference-in-differences estimation. Each method offers its own advantages and limitations, and a comprehensive analysis of these techniques in the context of our project could provide valuable insights into the most effective approach for estimating causal relationships in the ICU setting.

The importance of causal inference in Health AI cannot be understated, as it facilitates the identification of effective interventions, leading to better patient outcomes. By focusing on causal inference in our project, we seek to contribute valuable insights to the field and motivate further research in this area. Implementing causal inference on our LSTM model trained on the MIMIC-IV dataset will enable us to investigate and validate the causal relationships in multivariate time series forecasting of patients' health in the ICU, paving the way for more targeted and effective healthcare solutions. By exploring and comparing different causal inference techniques, including the powerful Dragon-Net, we hope to advance the state of the art in Health AI and improve the quality of care for patients in the ICU.

7 Conclusion

In this thesis, we addressed several limitations faced by traditional methods for multivariate time series forecasting, specifically: (a) complex temporal dependencies, (b) non-linear relationships, (c) large-scale/high-dimensional data, and (d) missing data. By employing Long Short-Term Memory (LSTM) networks, we were able to overcome these challenges and perform multivariate time series forecasting on Electronic Health Records (EHR) data.

Throughout this work, we successfully accomplished the following goals: (a) evaluated the use of LSTM as a viable choice for performing multivariate time series forecasting, and (b) uncovered the limitations of the model. We demonstrated the effectiveness of LSTM networks in handling large-scale, high-dimensional data, as well as their ability to model complex temporal dependencies and non-linear relationships. Moreover, we showed how LSTMs can deal with missing data much better than traditional methods, which is an issue that is often encountered in EHR datasets.

The final goal of this thesis, (c) identifying the underlying cause-and-effect relationships between the latent variables of the model, has been proposed as future work. This task holds great significance, as understanding causal relationships in health data is crucial for personalized treatment and decision-making. By performing causal inference on the trained LSTM models, we can gain valuable insights into the underlying dynamics of patient health progression in the ICU.

The work presented in this thesis is novel and significant in that it addresses a gap in the existing literature by focusing on generative modeling for multivariate time series forecasting in the context of health data. By developing and evaluating LSTM models on real-world EHR data, we have laid a foundation for further research in this domain.

In summary, our key novel contributions include the successful application of LSTM networks for multivariate time series forecasting using EHR data, addressing limitations of traditional methods, and setting the stage for future work in causal inference. This work has the potential to make a breakthrough in predictive health monitoring, ultimately contributing to improved patient care and outcomes in the ICU and beyond.

References

[1] E. Choi, A. Schuetz, J. Kulas, M. T. Bahadori, J. Sun, and W. Stewart, *Retain: An interpretable predictive model for healthcare using reverse time attention mechanism*, 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/231141b34c82aa95e48810a9d1b33a79-Paper.pdf.

[2] J. S. Hatchimonji, E. J. Kaufman, C. E. Sharoky, L. Ma, A. E. Garcia Whitlock, and D. N. Holena, *Failure to rescue in surgical patients: A review for acute care surgeons*, Sep. 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6711800/.

[3] A. Johnson, L. Bulgarelli, T. Pollard, S. Horng, L. A. Celi, and R. Mark, *Mimic-iv v2.0*, Jun. 2022. [Online]. Available: https://physionet.org/content/mimiciv/2.0/.

[4] M. K. Baowaly, C. Lin, C. L. Liu, and K. T. Chen, *Synthesizing electronic health records using improved generative adversarial networks*, Mar. 2019. [Online]. Available: https://academic.oup.com/jamia/article/26/3/228/5235390.

[5] D. Charles, M. Gabriel, and M. F. Furukawa, "Adoption of electronic health record systems among us non-federal acute care hospitals: 2008-2012," Tech. Rep., 2013.

[6] Y. N., J. E., and H. E., "Office-based physician electronic health record adoption," Tech. Rep., 2016.

[7] T. Tran, W. Luo, D. Phung, *et al.*, "A framework for feature extraction from hospital medical data with applications in risk prediction," *BMC Bioinformatics*, vol. 15, p. 6596, Dec. 2014. DOI: 10.1186/s12859-014-0425-8.

[8] H. Suresh, N. Hunt, A. Johnson, L. A. Celi, P. Szolovits, and M. Ghassemi, "Clinical event prediction and understanding with deep neural networks," in *Proceedings of Machine Learning for Healthcare*, vol. 68, Jan. 2017.

[9] T. Pham, T. Tran, D. Phung, and S. Venkatesh, "Predicting healthcare trajectories from medical records: A deep learning approach," *Journal of Biomedical Informatics*, vol. 69, pp. 218–229, 2017, ISSN: 1532-0464. DOI: 10.1016/j.jbi.2017.04.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046417300710.

[10] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, "Doctor ai: Predicting clinical events via recurrent neural networks," *CoRR*, vol. abs/1511.05942, 2015. [Online]. Available: http://arxiv.org/abs/1511.05942.

[11] D. A. Kaji, J. R. Zech, J. S. Kim, and et al., "An attention based deep learning model of clinical events in the intensive care unit," *PLoS One*, vol. 14, 2019. DOI: 10.1371/journal.pone.0211057. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6373907/#pone.0211057.s003.

[12] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel, "Learning to diagnose with lstm recurrent neural networks," *arXiv: 1511.03677 [cs.LG]*, 2017. [Online]. Available: http://arxiv.org/abs/1511.03677.

[13] W. Farhan, Z. Wang, Y. Huang, S. Wang, F. Wang, and X. Jiang, "A predictive model for medical events based on contextual embedding of temporal sequences," *JMIR Med Inform*, vol. 4, 2016. DOI: 10.2196/medinform.5977. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5148810/#app1.

[14] A. E. W. Johnson and R. G. Mark, "Real-time mortality prediction in the intensive care unit," *AMIA Annu Symp Proc.*, vol. 2017, pp. 994–1003, Apr. 2018.

[15] L. Zhou, P. Zhao, D. Wu, and et al, "Time series model for forecasting the number of new admission inpatients," *BMC Medical Informatics and Decision Making*, vol. 18, p. 39, 2018. DOI: 10.1186/s12911-018-0616-8.

[16] D. Wang, J. Li, Y. Sun, *et al.*, "A machine learning model for accurate prediction of sepsis in icu patients," *Frontiers in Public Health*, vol. 9, p. 754 348, 2021. DOI: 10.3389/fpubh.2021.754348.

[17] S. B. Golas, T. Shibahara, S. Agboola, and et al, "A machine learning model to predict the risk of 30-day readmissions in patients with heart failure: A retrospective analysis of electronic medical records data," *BMC Medical Informatics and Decision Making*, vol. 18, p. 44, 2018. DOI: 10.1186/s12911-018-0620-z.

[18] L. Su, Z. Xu, F. Chang, *et al.*, "Early prediction of mortality, severity, and length of stay in the intensive care unit of sepsis patients based on sepsis 3.0 by machine learning models," *Frontiers in Medicine*, vol. 8, p. 664 966, 2021. DOI: 10.3389/fmed.2021.664966.

[19] X. Chen, Y. Chen, Z. Zhang, W. Gao, J. Li, and J. Shi, "Forecasting icu stays using multivariate time series models," *Journal of Medical Systems*, vol. 43, no. 12, pp. 715–723, 2019.

[20] J. Kim, Y. Cho, Y. Lee, K. Kim, H. Kim, and J. Kim, “Multivariate time series analysis for mortality prediction in intensive care units,” *Journal of Medical Systems*, vol. 46, no. 3, pp. 220–230, 2020.

[21] M. Ali. “Tutorial: Time series forecasting.” (2022), [Online]. Available: `https://www.datacamp.com/tutorial/tutorial-time-series-forecasting`.

[22] J. Brownlee, *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Jason Brownlee, 2018.

[23] J. Brownlee, *Long Short-Term Memory Networks with Python*. Jason Brownlee, 2019.

[24] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.

[25] H. Cheng, Y. Dong, X. Li, and W. Wang, “A dual-stage attention-based recurrent neural network for time series prediction,” *arXiv preprint arXiv:1704.02971*, 2017.

[26] Y. Shi, Z. Bai, L. Yao, *et al.*, “Convolutional lstm network: A machine learning approach for precipitation nowcasting,” in *Advances in Neural Information Processing Systems*, 2015.

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

[28] K. Cho, B. van Merriënboer, C. Gulcehre, *et al.*, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[29] C. W. Granger, “Investigating causal relations by econometric models and cross-spectral methods,” *Econometrica: Journal of the Econometric Society*, vol. 37, no. 3, pp. 424–438, 1969.

[30] H. Lutkepohl, *New introduction to multiple time series analysis*. Springer Science Business Media, 2005.

[31] C. Brooks, *Introductory econometrics for finance*. Cambridge University Press, 2015.

[32] C. Shi, D. M. Blei, and V. Veitch. “Adapting neural networks for the estimation of treatment effects.” (Oct. 2019), [Online]. Available: `https://arxiv.org/pdf/1906.02120.pdf`.

[33] E. De Brouwer, J. Simm, A. Arany, and Y. Moreau, “Gru-ode-bayes: Continuous modeling of sporadically-observed time series,” *arXiv preprint arXiv:1905.12374v2*, 2019.

[34] B. Roy. “All about feature scaling,” Towards Data Science. (Apr. 2020), [Online]. Available: https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35.

[35] J. Brownlee. “How to handle missing timesteps in sequence prediction problems with python,” Machine Learning Mastery. (Aug. 2020), [Online]. Available: https://machinelearningmastery.com/handle-missing-timesteps-sequence-prediction-problems-python/.

A Appendix Code

All the code used for this research is available on Github via the following link: `https://github.com/Shardul-Ghuge/Predicting-future-medical-diagnoses-with-LSTM`.
All the documentation for this is available in README.