RECURRENT NEURAL NETWORKS FOR SUPERVISED LEARNING OF MULTI-MORBIDITY PATTERNS

A REPORT SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF BACHELOR OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2024

Dinil Tandel

Supervised by Dr Mauricio Álvarez

Department of Computer Science

Contents

Word Count: 8564

List of Tables

List of Figures

Abstract

RECURRENT NEURAL NETWORKS FOR SUPERVISED LEARNING OF MULTI-MORBIDITY PATTERNS
Dinil Tandel
A report submitted to The University of Manchester
for the degree of Bachelor of Science, 2024

This project investigates the use of different Recurrent Neural Network architectures to make predictions for future diagnoses in the medical domain. Two distinct approaches were evaluated using standard performance metrics to assess their ability to make meaningful predictions. The first uses a single model to train and predict all the input diseases, whereas the second approach uses a separate model for each disease, independently trains and predicts before finally combining these predictions. This builds on previous research that uses Temporal point processes, Large language models, or Convolutional Neural Networks for the same task.

Evaluation of the two approaches showed the low performance of the first approach and the significant boost in performance when slightly changing just the target variable for the prediction. This however has the downside that more resources are used and a longer training time is needed. However, further research is needed to assess whether the first approach can be viable if given enough data. Suggestions are given for future work that might address this issue, as well as various enhancements to this work.

Declaration

No portion of the work referred to in this report has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Abbreviations and Acronyms

RNN Recurrent Neural Network

LSTM Long Short Term Memory

BiLSTM Bidirectional Long Short Term Memory

BPTT Backpropagation Through Time

EHR Electronic Health Record

QoL Quality of life

OOP Out Of Pocket

Acknowledgements

I would like to thank my supervisor, Dr. Mauricio Álvarez for his continued support throughout the research and implementation of this project. He helped guide my work in the right direction and gave useful pointers when things got stagnant, while also giving me the freedom to explore interesting and novel ideas. I would also like to thank Dr. Pranav Tandel, a professional in the medical domain, whose valuable insights provided a crucial turning point for the project. Lastly, I would like to thank all my friends and family for their support and encouragement over the course of my education.

Chapter 1

Introduction

This chapter gives some context behind this project and some motivation for why it is an important problem that should be researched further(1.1). Next, the Project aims along with their justification are listed(1.2). Finally, a brief structure for the rest of the report is given(1.3).

1.1 Context and Motivation

Multimorbidity refers to the occurrence of two or more chronic or acute conditions in a single patient. It is increasingly common due to changes in lifestyle, notably lack of physical activity and obesity.[3, 4, 5, 6] As such, it is more prevalent in older populations with a global average of 37%.[7] A study showed that more than half of the adult population worldwide above 60 years of age had suffered from multiple conditions.[7] However, though age is the biggest factor for multimorbidity, in absolute numbers, more people less than the age of 65 years of age have multimorbidity than those higher, partly because more people in the general population are in that age group. This emphasizes that multimorbidity is not just a feature of ageing.[4, 6]

1.1.1 Impact on Quality of Life

People with multiple conditions usually have poor health, lower quality of life, and are at an increased risk of hospitalisation. Studies have shown that the co-existence of multiple conditions is associated with an increase in disability and functional decline[8, 9, 10] and an increased risk of mortality even after accounting for age.[11, 12, 13, 14]

Moreover, different combinations of conditions affect an individual's QoL differently.[15] These combinations have been found to have a larger negative impact than if they were considered independently.[16, 17, 15] These are also associated with complex treatment plans, involving multiple medications. This is termed as "Treatment Burden", which is the negative impact on a patient's time and energy due to accessing such care.[18, 19] This is an important clinical concern as patients who feel overwhelmed are less likely to adhere to medications and leads to reduced perception of the care provided.[20, 21] Evidence based on UK patients show that those with multiple conditions have a less positive experience with primary care when compared with those with single or no chronic conditions.[22]

1.1.2 Impact on Caregivers and Healthcare Professionals

Evidence shows that caring for someone with a chronic condition is associated with increased rates of both mental health and physical conditions in the carer, and is associated with increased mortality.[23, 24, 25] A study on the experiences of patients in their last years of life reported that both they and their caregivers struggle with managing multiple medications, along with frustrations as a result of poor communication and lack of coordination between specialities.[26]

It also poses substantial challenges for healthcare professionals (HCP), who are often limited in their available time and resources and may face difficulties when trying to adhere to multiple clinical guidelines with a single patient.[27, 28, 29] These guidelines often focus on the management of a single condition and HCPs may face difficulties in adjusting treatment plans when dealing with multimorbidity and co-prescribing outside their specialities. A review found that clinicians face a range of problems, most of which stem from the fact multimorbid patients may require care/treatments from multiple specialties at the same time, which increases complexity and could introduce barriers to shared decision making.[30] There is also evidence suggesting that these difficulties could lead to reduced quality of care provided.[31]

A large proportion of workload in clinics is from patients with multimorbidity. Studies show an increased number of primary care visits and hospital admissions among these patients, with several reporting that this is independent of factors such as age, sex and socioeconomic status.[32, 33, 34] Evidence also show that unplanned

hospital admissions for a physical condition are particularly high for patients suffering from mental health issues like depression[35], dementia, schizophrenia, and learning disabilities as well as those who suffer from problems with alcohol and psychoactive substances. [36]

1.1.3 Economic Impact

The association between multimorbidity and healthcare costs is reported to be almost exponential[34], suggesting that the cost of care for patients with multimorbidity is higher than would be predicted based on the individual components. Data from the UK indicates that multimorbidity is a key aspect of health and social care costs, more than explained by age alone.[37]

The financial burden is found to be highest for patients who have to pay on their own. Evidence from the USA shows that out-of-pocket (OOP) expenses increase with the number of chronic conditions, regardless of age.[38, 39] A direct relationship between the number of chronic conditions and OOP spending has also been reported in older adults in a number of other countries where patients are, at least partially, responsible for covering their own healthcare costs.[32, 40, 41]

1.2 Project Aims

This project aims to build a predictive model for patient health records. The model should be able to use past diagnoses to find patterns and relationships between various multi-morbidity diseases. The implementation aims to do so while also being versatile enough so that it can be used within various healthcare systems, independent of which illnesses may be prevalent in them. This would allow for widespread adoption in diverse healthcare settings, maximising its utility and impact. The main goals of this project is as follows:

- To design and implement a simple yet effective predictive system for electronic health records.
- To evaluate the system using standard evaluation metrics such as accuracy, precision, recall etc.
- To have the system perform well enough that it can be applicable in real world scenarios.

The first objective is the includes the data preprocessing steps. The ideal scenario is to create a black-box solution, where the relevant databases are added as input, and a solution is provided as output, without needing to format the data in a ready state or needing to know the exact inner workings of the model. It also aims to keep the model computationally efficient so it can be used in healthcare systems where large computation resources may not be available. This is to ensure the accessibility of the system.

The second and third objectives go hand in hand in that the model should achieve high performance. This is to mitigate any chances of wrong predictions and thus, misdiagnoses, which is a common concern and a tricky problem to overcome when it comes to multimorbidity.

1.3 Structure

This report consists of 6 chapters and 1 appendix. Details of each chapter are listed below:

- Chapter 1 gives an introduction to the project along with some context and motivation. It also lists the goals and objectives for the project.
- Chapter 2 details some background to get up to speed in the Machine Learning techniques used, Namely RNN and LSTM. Additionally, it lists some related work.
- Chapter 3 describes the data used to train the models along with some preprocessing steps taken.
- Chapter 4 describes the structure of the networks used. It also lists the metrics used to evaluate the performance of these models.
- Chapter 5 presents the results of the experiments run. It briefly covers the first approach before moving on to a more successful one, showing the results for different models when trained on different amounts of data. It then provides a brief analysis of these results and tests the model with a larger list of possible diseases
- Chapter 6 discusses the conclusions to the project along with proposing some future work that can be done on top of this project.
- Appendix A shows the code listings for the tweaked functions for approach two.

Chapter 2

Background and Related Work

This chapter serves as a way to get up to speed with the primary technologies used in the implementation and report. It assumes basic knowledge in the Machine Learning domain. It begins with an overview of Recurrent Neural Networks (RNN), followed by an overview of Long Short Term Memory (LSTM) networks. Finally, it goes over some related work in the domain.(2.4)

2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of neural networks designed to work with sequential data. Unlike feedforward neural networks, where information flows in one direction, RNNs take information from prior inputs to influence the current input and output. While traditional deep neural networks assume input and output are independent of each other, the output of RNNs depends on previous elements in the sequence.

2.1.1 Architecure

The basic architecture of an RNN consists of a loop that allows information to be passed from one step of the network to the next. Each step takes as input both the current input and the output from the previous step, allowing the network to maintain a state or memory of previous inputs. Moreover, parameters are shared across each layer in the network. While feedforward networks have different weights across each layer, RNNs share the same weights between each layer. These parameters are still updated during backpropagation and gradient descent for reinforcement learning.[42, 1]

Backpropagation in RNNs is also slightly different, using backpropagation through time (BPTT), to determine gradients as this is specific to sequential data. The principles remain the same, training is done by calculating errors between output and input using a loss function, which allows the model to adjust the parameters and weights accordingly. BPTT sums errors at each time step as the parameters are shared across layers.[42, 1]

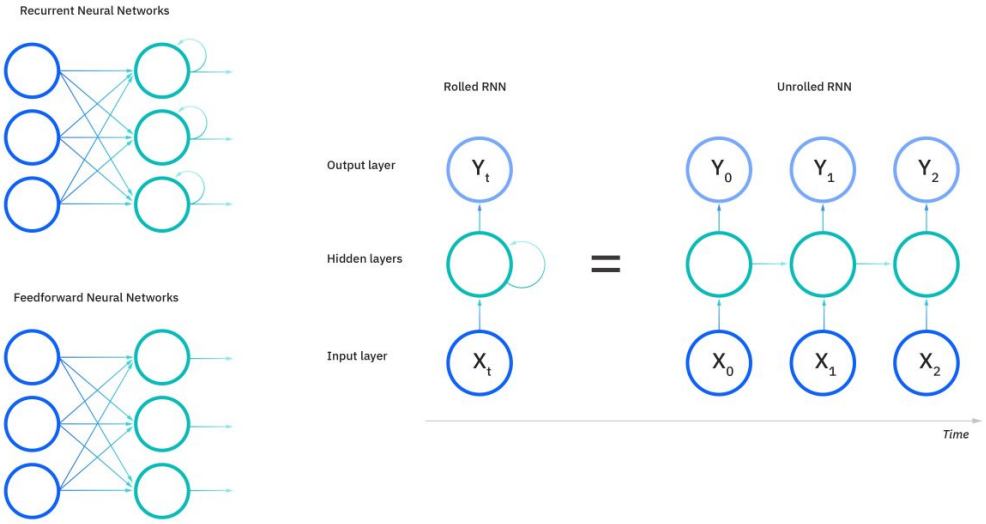

Figure 2.1: RNN diagram [1]

2.1.2 Challenges

- **Vanishing Gradient Problem:** RNNs often struggle to learn long-range dependencies due to the vanishing gradient problem, where gradients diminish exponentially as they propagate backwards through time. When the gradient is too small, it continues to become smaller, and the weights keep getting smaller until they become 0. When this happens, the model is no longer learning and begins to underfit, which means the model isn't able to make accurate predictions.[43]
- **Exploding Gradient Problem:** Exploding gradients can occur when the gradient is too large and the weights grow exponentially, usually to NaN (not a number), leading to an unstable model that makes erratic predictions and overfitting. Overfitting is when a model can make accurate predictions with training

data but not with test data, or in other words, memorises data rather than learning from it.[43]

- **Difficulty Capturing Long-Term Dependencies:** RNNs have a limited ability to capture long-term dependencies in sequential data, which can impact their performance on tasks requiring such dependencies.

  RNNs are at risk of vanishing and exploding gradients when they process longer sequences.

2.2 Long Short Term Memory

Long Short-Term Memory (LSTM) networks are a type of RNN designed to address the shortcomings of traditional RNNs in capturing long-term dependencies. LSTMs use a more complex architecture that includes mechanisms for selectively remembering and forgetting information over time.[1, 2, 44]

2.2.1 Architecure

LSTMs add a special memory block called cells in each hidden layer and a cell state. The key components of an LSTM cell are the input gate, forget gate and output gate. These gates control the flow of information into and out of the cell, allowing LSTMs to regulate the information they store and process.[1, 2, 44]

The cell state is a way for information to go through from input to output. There are only a few interactions with other components, allowing data to pass unchanged.

LSTMs have the ability to add or remove information to the cell state, controlled by the three gates.

Input Gate

The input gate decides which information to store in the memory cell. It opens when the input is important and closes when it is not. the current input is multiplied by the hidden state and the weights from the previous loop. Next, important information is added to the cell state and a new state is formed. This new cell state then becomes the current state which will be used in the next loop.[44, 43]

Forget Gate

The forget gate decides which information to remove from the memory cell. It opens when the information is no longer important and closes when it is. These gates use a sigmoid operation, which outputs a number between 0 and 1, indicating how much of the information to include. 1 means include everything and 0 is include nothing.[44, 43]

Output Gate

The output gate is responsible for deciding which information to use for the output of the LSTM. It opens when the information is important and closes when it is not. The output is calculated in the hidden state. To do this, a sigmoid function is used which decides what information comes through the gate and then the cell state is multiplied after being activated with a tanh function.[44, 43]

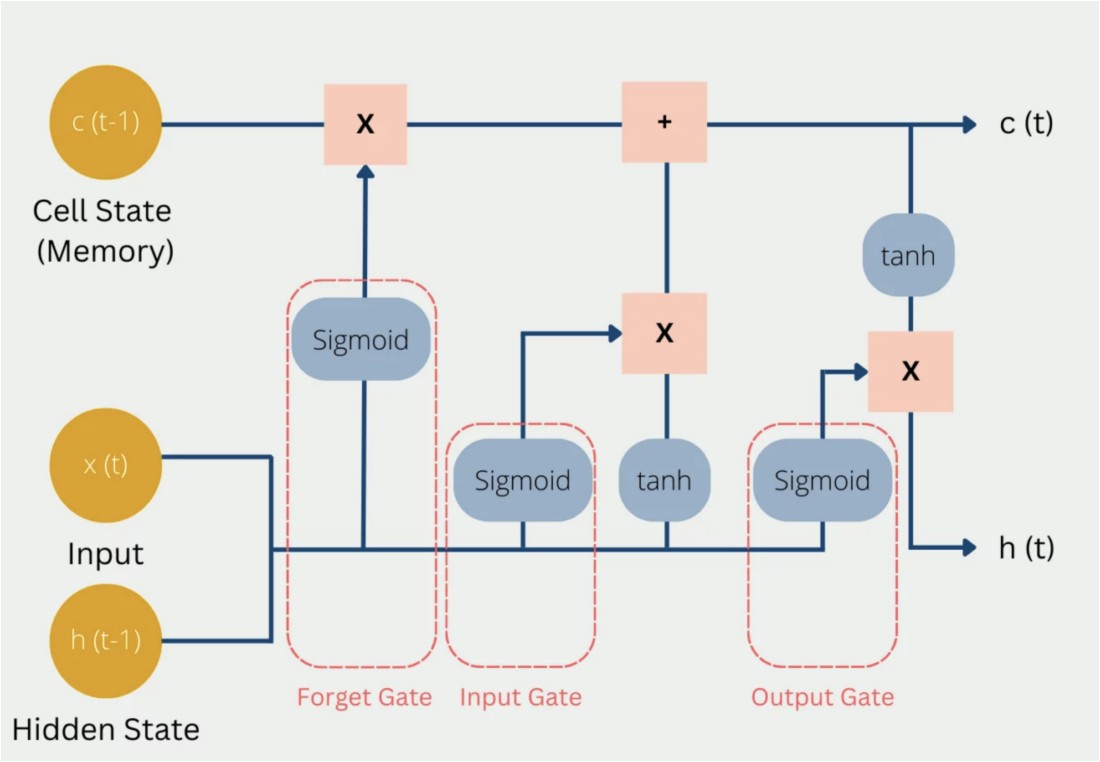

Figure 2.2: LSTM diagram[2]

2.2.2 Advantages

- **Better Long-Term Dependency Handling:** LSTMs are capable of learning and remembering long-range dependencies in sequential data, making them more effective than traditional RNNs for tasks requiring such capabilities.

- **Reduced Vanishing Gradient Problem:** LSTMs mitigate the vanishing gradient problem through the use of carefully designed gating mechanisms that allow gradients to flow more freely during training.

- **Flexibility and Versatility:** LSTMs can be adapted to a wide range of sequential data tasks and have been successfully applied in domains like natural language processing, speech recognition, and time series prediction.

- **Variants and Extensions:** Researchers have developed various extensions and variants of the basic LSTM architecture, including peephole connections, gated recurrent units (GRUs), and different attention mechanisms.

2.2.3 Applications

- **Language Modeling:** LSTMs are widely used for tasks like text generation, machine translation, and language understanding.

- **Speech Recognition:** LSTMs can process audio data and convert speech signals into text, enabling applications like virtual assistants and speech-to-text systems.

- **Time Series Prediction:** LSTMs are effective for predicting future values in time series data, making them useful for applications like financial forecasting and weather prediction.

2.2.4 Challenges

LSTMs suffer from the problem of overfitting, especially when working with small datasets. To overcome this, regularisation techniques like dropout which introduce a degree of randomness to an ideally deterministic output. Conversely, weight decay may be used, with the disadvantage that this slows down learning rate, which results in a longer time for the weights to converge to optimal values.

2.3 Bidirectional LSTM

Bidirectional LSTMs (BiLSTM) extend the capabilities of LSTM by training the input data twice, in forward and backwards directions. This helps overcome the limitations of traditional LSTMs that the current state can be constructed in forwards as well as backwards context.

2.4 Related Work

[45] proposes the use of Convolutional Neural Network (CNN) and K nearest Neighbor (KNN) to predict chronic diseases. The method achieves 95% classification accuracies which is higher than existing algorithms such as Naïve Bayes, decision tree, and logistic regression.

[46] incorporates Large Language Models (LLM) as zero shot time series forecasters. This works on multiple time series datasets and MIMIC-III was one of the datasets compatible with the algorithm. They call this method LLMTime.

[47] proposes several neural network parameterisations of Temporal Point Processes (TPP) and evaluates them on synthetically generated Electronic Health Records. Their aim was to predict the future health interactions of a patient given their medical history, determine missing data in a patient EHR and provide representations for EHRs so they can be searched/retrieved semantically/efficiently.

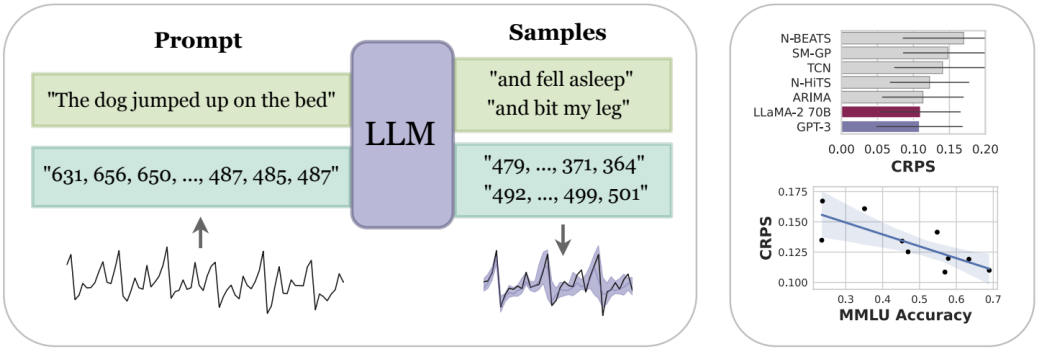

Figure 2.3: LLMtime

Chapter 3

Data and Preprocessing

The dataset used is the MIMIC-IV dataset obtained from PhysioNet. [48]

The data is described in [48] as follows:
MIMIC-IV is sourced from two in-hospital database systems: a custom hospital wide EHR and an ICU specific clinical information system. The creation of MIMIC-IV was carried out in three steps:

- **Acquisition:** Data for patients who were admitted to the BIDMC emergency department or one of the intensive care units were extracted from the respective hospital databases. A master patient list was created which contained all medical record numbers corresponding to patients admitted to an ICU or the emergency department between 2008 - 2019. All source tables were filtered to only rows related to patients in the master patient list.

- **Preparation** The data were reorganized to better facilitate retrospective data analysis. This included the denormalization of tables, removal of audit trails, and reorganization into fewer tables. The aim of this process is to simplify retrospective analysis of the database. Importantly, data cleaning steps were not performed, to ensure the data reflects a real-world clinical dataset.

- **Deidentify** Patient identifiers as stipulated by HIPAA were removed. Patient identifiers were replaced using a random cipher, resulting in deidentified integer identifiers for patients, hospitalizations, and ICU stays. Structured data were filtered using look up tables and allow lists. If necessary, a free-text deidentification algorithm was applied to remove PHI from free-text. Finally, date and times were shifted randomly into the future using an offset measured in days. A single date shift was assigned to each subject_id. As a result, the data for a single

patient are internally consistent. For example, if the time between two measures in the database was 4 hours in the raw data, then the calculated time difference in MIMIC-IV will also be 4 hours. Conversely, distinct patients are not temporally comparable. That is, two patients admitted in 2130 were not necessarily admitted in the same year.

After these three steps were carried out, the database was exported to a character based comma delimited format.

The database contains various documents with different data relating to patients, diagnoses, etc. However, for the purposes of this project, only data from the ”admissions” and ”diagnoses” documents are needed.

The data from these documents need to be further preprocessed in order to be used by the network. The steps taken are:

3.1 Filtering

First, the only relevant data is contained in subject ID, hospital admission ID and admit time. All other data is disregarded. Following this, a threshold is applied to remove patients who have less than a certain number of admissions. Doing so ensures that later down the line, each patient has enough data for the model to adequately learn from.

3.2 Keep ICD-9 diagnoses

Next, only diagnoses under the version ICD-9 are considered. The classification has had a new revision and ICD-10 is the current version being used. However, the database contained many more data points under ICD-9 than it did for 10. Again, this ensures there is enough training and testing data. Under real world conditions, however, any version can be used as long as the data stays consistent throughout the training and testing process.

3.3 Compiling Admissions

Each diagnosis for a patient in the same admission is stored in a separate row. This step aggregates these diagnoses so admissions and subject ID become unique. This allows a single admission to store information about every currently relevant disease.

3.4 Merge codes and admissions

So far, data such as admission time and the specifics of diagnoses are stored separately. These are put together into a single table. Further processing is done on this newly constructed table.

3.5 Process Dates

In this step, admissions with more than a certain number of dates are filtered out. admissions with a large number of days in between may introduce other factors or uncertainties which might pollute the data.

3.6 Build History

Generally, only the medical conditions that explain the present evident complaints are given as diagnoses. We want to predict future diagnoses based on historical data. So history is built in a cumulative way. If a disease is present in an admission, it will show positive in all subsequent admissions.

3.7 Threshold number of Admissions

After the past steps, some patients may have too few admissions. So a threshold is applied again to make sure all patients have at least a certain number of admissions. For patients with a higher number of admissions, only the latest n admissions are used. This is because the model expects all the data to have an equal number of data points. Since history is being used, minimal information is removed by doing this. However, this may remove information about dependencies between certain diseases.

3.8 Make sequences

So far the diagnoses have been ICD codes. The model might have performance issues when using these codes. So these codes are encoded into vectors of 1s and 0s. Each vector contains every possible code where 1 indicates the presence of a disease and 0 the absence. This is achieved using a multilabel binariser, which can be used later on in the predicted vector to get the codes back using a reverse transformation.

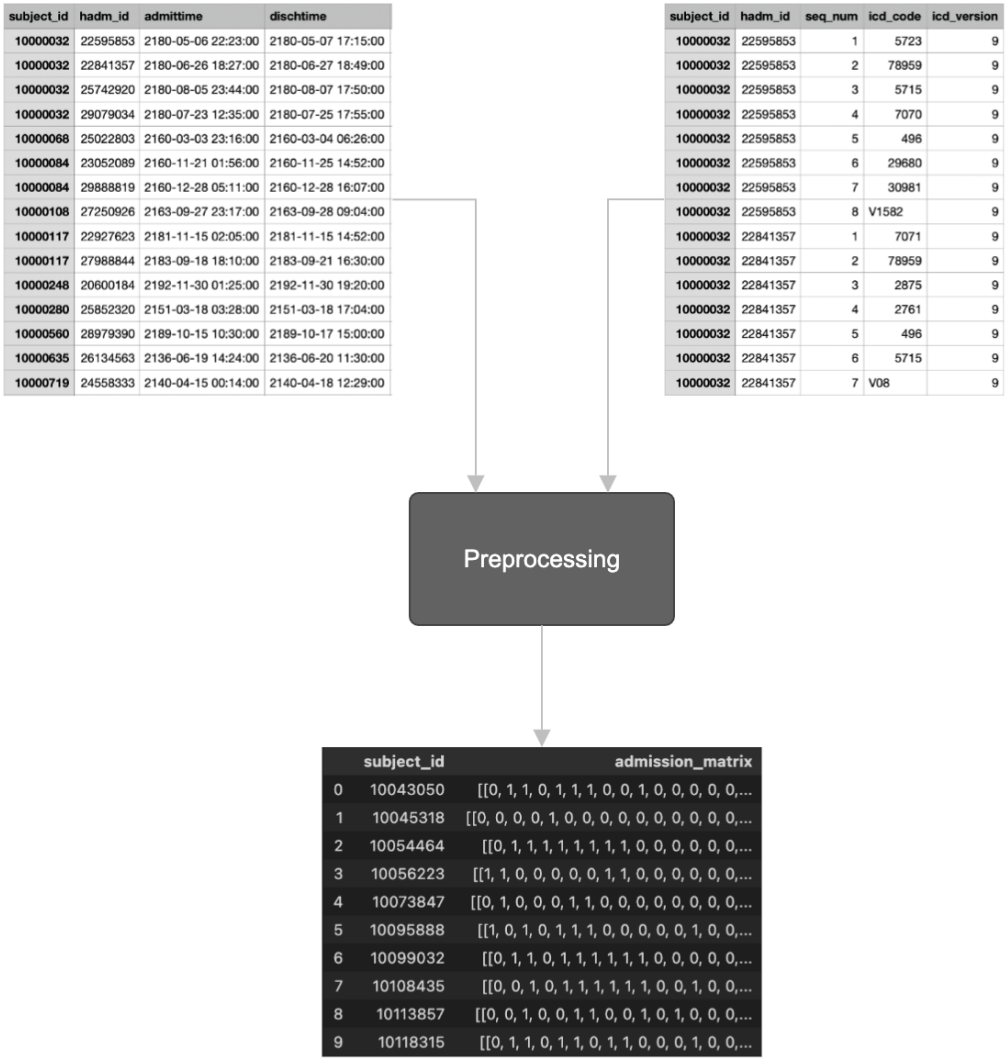

Figure 3.1: Preprocessing example

Chapter 4

Project Methodology

4.1 Network topology

The model starts with a BiLSTM layer. As discussed previously, LSTM is a type of RNN that is designed to handle sequential data, as is being used in this case. The bidirectional wrapper means the layer processes the input sequences in both forward and backward directions, allowing the model to capture information from past and future contexts simultaneously, increasing its ability to find temporal patterns. The layer's output dimensionality is set to 256, meaning it produces a tensor of shape (batch_size, sequence_length, 256), where batch_size represents the number of samples in each training batch, and sequence_length is the length of input sequences.

Following the first Bidirectional LSTM layer, a dropout layer is added with a dropout rate of 0.2. Dropout is a regularization technique used to prevent overfitting by randomly dropping a fraction of input units during training. By doing so, dropout encourages the model to learn more robust features and reduces its reliance on specific inputs rather than memorising the training data, thus improving generalization performance.

The model then proceeds with additional Bidirectional LSTM layers, each followed by a dropout layer with the same dropout rate of 0.2. These additional layers allow the model to capture increasingly complex patterns within the data by sequentially processing information through multiple layers of BiLSTM units. The second and third Bidirectional LSTM layers have 128 units each, while the fourth one has 64 units. This

decreasing number of units in the deeper layers helps in reducing computational complexity and mitigating overfitting.

After the final Bidirectional LSTM layer, a dense layer with a single unit and a sigmoid activation function is added. This dense layer serves as the output layer of the model, producing a single output value representing the predicted probability of the diseases occurring. The sigmoid activation function scales the output into the range [0, 1], making it suitable for binary classification tasks, where the model predicts the probability of binary outcomes by rounding the output to either 0 or 1.

Finally, the model is compiled using the Adam optimizer and binary cross-entropy loss function. Adam is an adaptive learning rate optimization algorithm known for its effectiveness in training deep neural networks. Binary cross-entropy loss is commonly used for binary classification problems, measuring the difference between the predicted probability distribution and the true distribution of the labels.

Figure 4.1 shows a flowchart of the model design used.

Figure 4.1: Model Design

4.2 Metrics used

The metrics used to measure performance are:

4.2.1 Accuracy

Accuracy measures the proportion of correctly predicted outcomes (both true positives and true negatives) among all predictions made by the model. While accuracy provides an overall assessment of the model's performance, it may not be the most informative metric, especially in imbalanced datasets where one class dominates the other. In healthcare settings, where the prevalence of certain diseases may be low, accuracy alone might not adequately reflect the model's effectiveness.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

4.2.2 Precision

Precision quantifies the proportion of true positive predictions among all positive predictions made by the model. It focuses on the correctness of positive predictions and is particularly relevant in scenarios where false positives can have significant consequences. In healthcare, precision is crucial for ensuring that predicted diagnoses are accurate and reliable, as incorrect predictions could lead to unnecessary treatments, tests, or interventions.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

4.2.3 Recall

Recall, also known as sensitivity, measures the proportion of true positives that were correctly identified by the model among all actual positives in the dataset. It emphasizes the model's ability to capture all relevant instances of a particular class, thereby minimizing false negatives. In healthcare applications, recall is essential for ensuring that all relevant diagnoses are identified, as missing critical diagnoses could result in delayed treatment or adverse patient outcomes.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

4.2.4 F1 Score

It combines both precision and recall into a single score, providing a balanced measure of a model's ability to correctly classify instances of both positive and negative classes. The F1 score is calculated as the harmonic mean of precision and recall:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.4}$$

The harmonic mean ensures that the F1 score gives equal weight to both precision and recall. This means that the F1 score will be high only if both precision and recall are high.

In all the above formulae,
TP represents the number of true positives (correctly predicted positive cases).
TN represents the number of true negatives (correctly predicted negative cases).
FP represents the number of false positives (incorrectly predicted positive cases).
FN represents the number of false negatives (incorrectly predicted negative cases).

Chapter 5

Results and Analysis

When making the predictions, two distinct approaches were investigated. In the first approach(5.1), the entire list is predicted by the same model. In the second approach(5.2), each disease is predicted one at a time. This is done using multiple models.

5.1 Approach 1: Predicting all diseases

The output from the model is changed slightly. In this approach, instead of having a single output, the size is set to how many diseases are present in category mapping. For the following results, category mapping is set to ICD-9 categories resulting in 19 total diagnoses.

The first factor that could affect performance is how many admissions per patient are used during training.

Table 5.1: Varying min admissions using approach 1

admissions per patient	total train admissions	Accuracy	Precision	Recall	F1-Score
3	11808	0.1352	1.0000	0.0217	0.0425
4	6068	0.0763	1.0000	0.0214	0.0418
5	3600	0.1222	1.0000	0.0252	0.0492
6	2508	0.0857	1.0000	0.0127	0.0251

The table 5.1 and the figure show that the precision is suspiciously high. To check the values, the predicted output was printed. It predicted a label of 0 for every disease. This indicates that the model is unable to learn any meaningful patterns.

This issue may have occurred due to the fact that there wasn't much training data. Since the preprocessing makes the most use of the data as possible, this approach cannot be further used. It may be beneficial to do the experiment again when more data is available, like in a real hospital scenario.

5.2 Approach 2: Predicting one disease at a time

A few of the functions have been tweaked (shown in Appendix A)to work with this approach. These are:

Getting the target variable

The function is designed to work in a loop. The training data remains the same across all iterations, assuming *n* is the number of admissions per patient, the train data contains the first *n - 1* admissions. It iterates over patients, excluding the last admission, and extracts the target variable for a specified disease from the final admission. The function returns training data with all but the last admission and the corresponding target variable.

Training the models

Since this approach trains multiple models, a way to train individual models had to be made. In this training function, first, the training data and target variable are acquired from the previously mentioned function. Iterating over each disease, a single model is trained and stored. Parameters such as epochs, batch size, and validation split are configurable for optimization. Following training, the models are evaluated using test data, and their performance metrics are recorded. Next, it computes aggregate metrics, including average loss, average accuracy, and total training time, providing insights into the overall model performance. Finally, these models can be saved as keras objects for later use.

Making Predictions

The stored models are used one at a time to predict the occurrence of their associated disease. For each disease, the function utilizes the first *n - 1* admissions from the input data to generate predictions. These predictions are then put together for each patient aligning with the input sequence format. If the input data is in a 2D format, it

is reshaped into a compatible 3D format. The function returns a matrix containing the predicted disease occurrences for all patients.

5.2.1 RNN

The RNN models were trained on the training data and evaluated on the test data. The average metrics after each evaluation can be seen in table 5.2. The largest values for each metric is highlighted in bold. table 5.2 shows results when the number of admissions per patient is changed.

Table 5.2: Varying min admissions for RNN

admissions per patient	total train admissions	Accuracy	Precision	Recall	F1-Score
3	11808	0.9346	**0.9927**	0.8695	0.9270
4	6068	**0.9472**	0.9779	0.9204	**0.9482**
5	3600	0.9193	0.9557	0.8931	0.9234
6	2508	0.9028	0.9481	0.8837	0.9148
7	1988	0.8925	0.9139	0.9011	0.9075
8	1416	0.8784	0.9275	0.8759	0.9010
9	1152	0.9112	0.9393	0.9199	0.9295
10	980	0.8968	0.9018	**0.9498**	0.9252
11	836	0.8026	0.7893	0.9325	0.8549
12	684	0.8284	0.8610	0.8703	0.8656

The RNN model shows a significant boost in performance over the previous approach.

Accuracy initially increased between 3 and 4, reaching its peak of 0.9472 at 4. Following this it continually decreases for the rest of the tests, reaching its minimum of 0.8026 at 11 before increasing again for the last. There are exceptions at 9 and 10, where performance is among the highest of all tests.

Precision was at its highest at 3 with a value of 0.9927. It continually decreases in an almost linear fashion other than at 8 and 9, where there is a bump in performance. Following this, it decreases again, reaching its minimum of 0.7893 at 11, before finally increasing again.

Recall values fluctuate with increasing the number of admissions. There don't

seem to be any discernable patterns from the results. It peaks at 10 with a value of 0.9498, which is unusual when taking the accuracy and Precision into context. The lowest value is 0.8695, at the smallest number of admissions.

F1-Score is taken as the harmonic mean of Precision and recall. Due to the fluctuation in the recall values, the F1-Score also fluctuates. However a general pattern is that after its peak of 0.9482 at 4, it gradually decreases until the end, with the exceptions of 9 and 10 which show high performance.

Figure 5.1 provides a visualisation for the F1-score metric for each test and Figure 5.2 shows a visualisation of the results from the table 5.2.

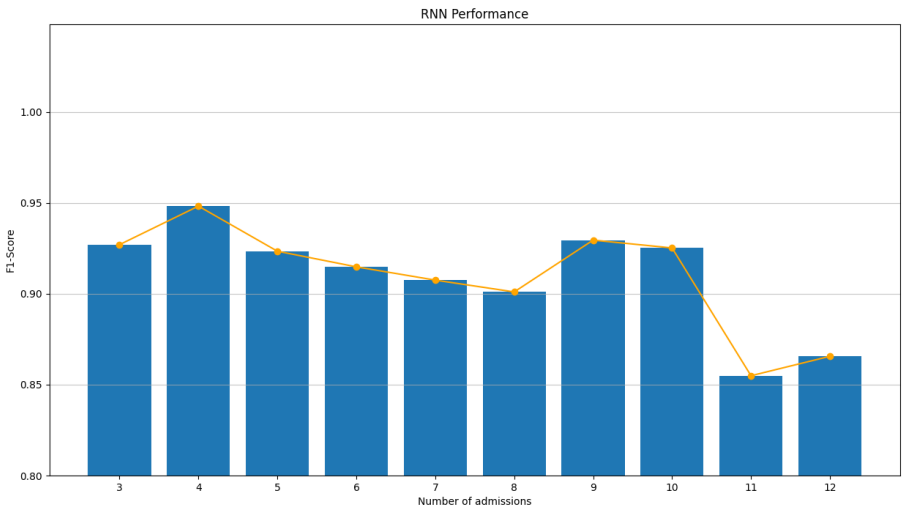

Figure 5.1: F1 Score for RNN

The best performance (using F1 Score) is taken and the best and worst performance is recorded in table 5.3

Table 5.3: Best and Worst RNN performance

	Accuracy	Precision	Recall	F1-Score
Best	1	1	1	1
Worst	0.7895	0.7500	0.5000	0.6000

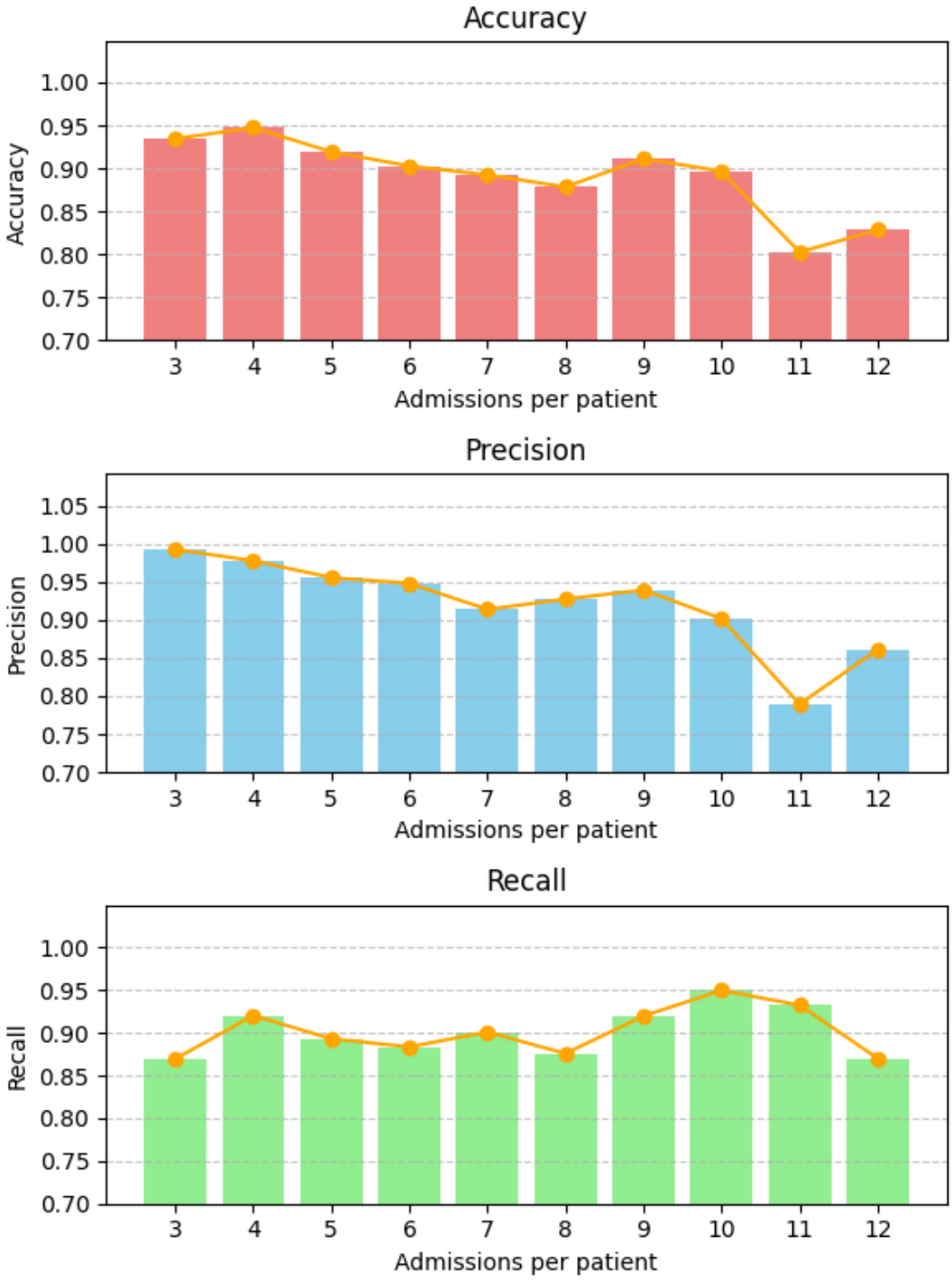

Figure 5.2: Metrics for RNN

5.2.2 LSTM

The LSTM models were trained on the training data and evaluated on the test data. The average metrics after each evaluation can be seen in table 5.4. The largest values for each metric is highlighted in bold. table 5.4 shows results when the number of admissions per patient is changed.

Table 5.4: Varying min admissions for LSTM

admissions per patient	total train admissions	Accuracy	Precision	Recall	F1-Score
3	11808	0.9344	**0.9984**	0.8640	0.9264
4	6068	**0.9378**	0.9720	0.9077	**0.9388**
5	3600	0.8985	0.9517	0.8571	0.9019
6	2508	0.8747	0.9150	0.8684	0.8911
7	1988	0.8621	0.8690	0.8999	0.8842
8	1416	0.8550	0.8611	0.9185	0.8889
9	1152	0.8405	0.8251	0.9509	0.8836
10	980	0.8758	0.8693	**0.9592**	0.9121
11	836	0.7711	0.7534	0.9409	0.8368
12	684	0.7825	0.7808	0.9243	0.8465

Accuracy initially increased between 3 and 4, reaching its peak of 0.9378 at 4. Following this it continually decreases for the rest of the tests, reaching its minimum of 0.7711 before increasing again for the last. 10 shows an exception, where there is a significant increase in accuracy, before immediately decreasing again.

Precision was at its highest at 3 with a value of 0.9984. It continually decreases in an almost linear fashion other than at 10, where there is a bump in performance. However, unlike the RNN, this bump is not very close to the peak performance. Following this, it decreases again, reaching its minimum of 0.7534 at 11, before finally increasing again.

Recall values show an increase as the number of admissions grows larger. It peaks at 10 with a value of 0.9592. Following this, it decreases again.

F1-Score is taken as the harmonic mean of Precision and recall. When combining Precision and recall, the highest value of 0.9388 is seen at 3. The high precision and recall performance at 10 makes it among the highest-performing values, which is reflected in the F1-Score.

Figure 5.3 provides a visualisation for the F1-score metric for each test and Figure 5.4 shows a visualisation of the results from the table 5.4.

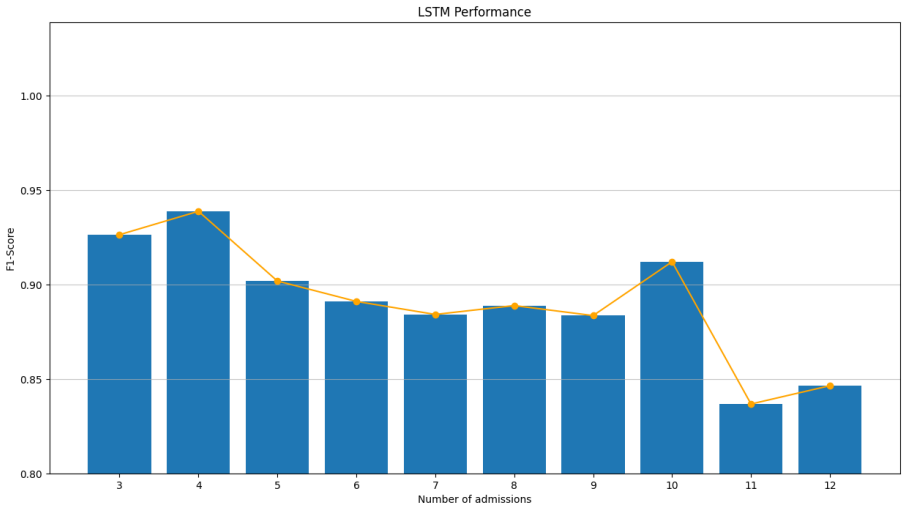

Figure 5.3: F1 Score for LSTM

The best performance (using F1 Score) is taken and the best and worst performance is recorded in table 5.5

Table 5.5: Best and Worst LSTM performance

	Accuracy	Precision	Recall	F1-Score
Best	1	1	1	1
Worst	0.8421	0.3333	0.5000	0.4000

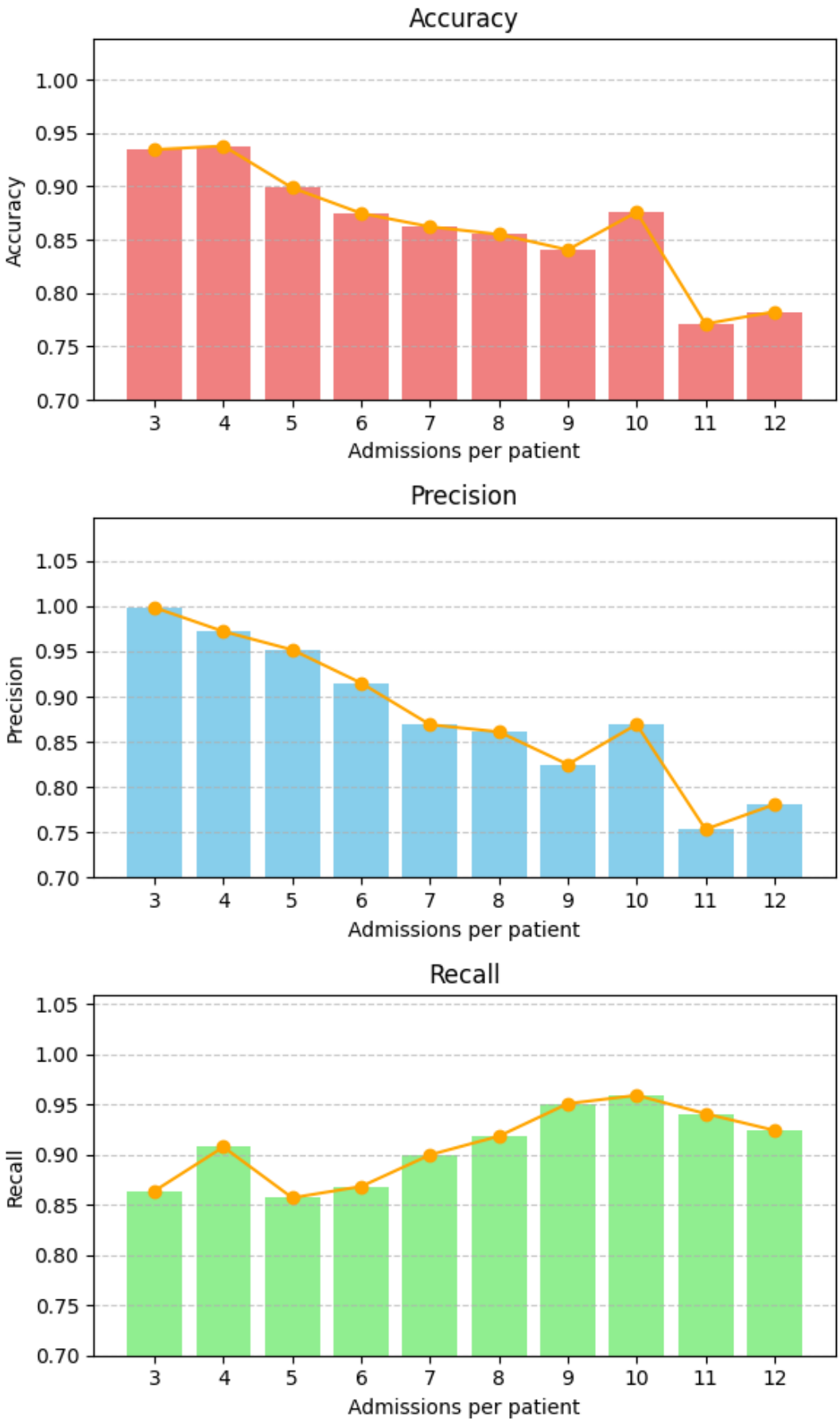

Figure 5.4: Metrics for LSTM

5.2.3 BiDirectional LSTM

The BiLSTM models were trained on the training data and evaluated on the test data. The average metrics after each evaluation can be seen in table 5.6. The largest values for each metric are highlighted in bold. table 5.6 shows results when the number of admissions per patient is changed.

Table 5.6: Varying min admissions for BiLSTM

admissions per patient	total train admissions	Accuracy	Precision	Recall	F1-Score
3	11808	0.9350	0.9987	0.8650	0.9271
4	6068	**0.9575**	**0.9991**	0.9198	**0.9578**
5	3600	0.9532	0.9913	0.9221	0.9555
6	2508	0.9459	0.9908	0.9168	0.9524
7	1988	0.9422	0.9506	0.9506	0.9506
8	1416	0.9298	0.9545	0.9333	0.9438
9	1152	0.9359	0.9372	**0.9638**	0.9503
10	980	0.9242	0.9492	0.9373	0.9432
11	836	0.8605	0.8594	0.9283	0.8925
12	684	0.8281	0.8269	0.9297	0.8753

Accuracy initially increased between 3 and 4, reaching its peak of 0.9575 at 4. Following this it continually decreases for the rest of the tests, reaching its minimum of 0.8281 at 12.

Precision was at its highest at 4 with a value of 0.9991. It continually decreases in steps, before seeing a bump in performance at 10. Following this, it decreases again, reaching its minimum of 0.8269 at 12.

Recall performance starts low but increases, reaching a peak of 0.9638 at 9, before decreasing again. An exception is at 4 where there is a sudden rise in performance, followed by an immediate decrease.

F1-Score is taken as the harmonic mean of Precision and recall. It peaks at 4 with a value of 0.9578 at 4 then slowly but gradually decreases. Due to the precision and recall being high at 9, so is the F1- Score, being among the highest here.

Figure 5.5 provides a visualisation for the F1-score metric for each test and Figure 5.6 shows a visualisation of the results from the table 5.6.

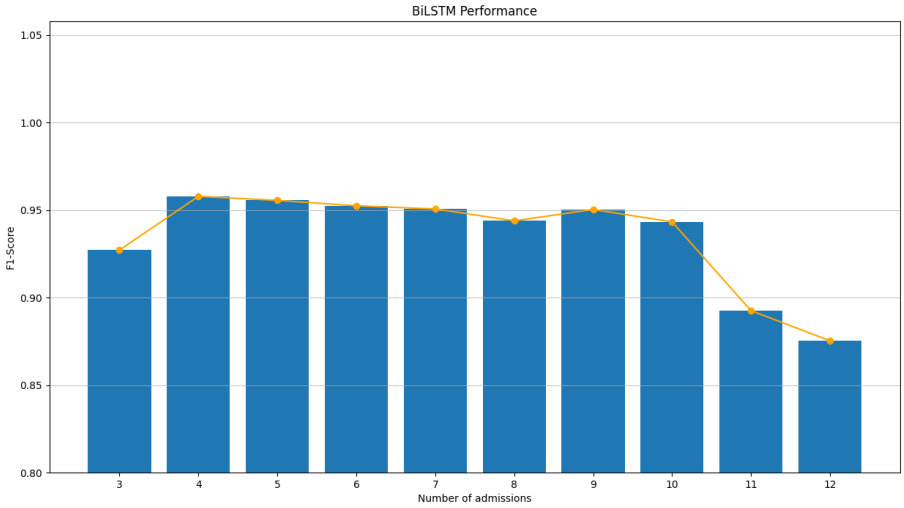

Figure 5.5: F1 Score for BiLSTM

The best performance (using F1 Score) is taken and the best and worst performance is recorded in table 5.7

Table 5.7: Best and Worst BiLSTM performance

	Accuracy	Precision	Recall	F1-Score
Best	1	1	1	1
Worst	0.7368	1	0.4444	0.6154

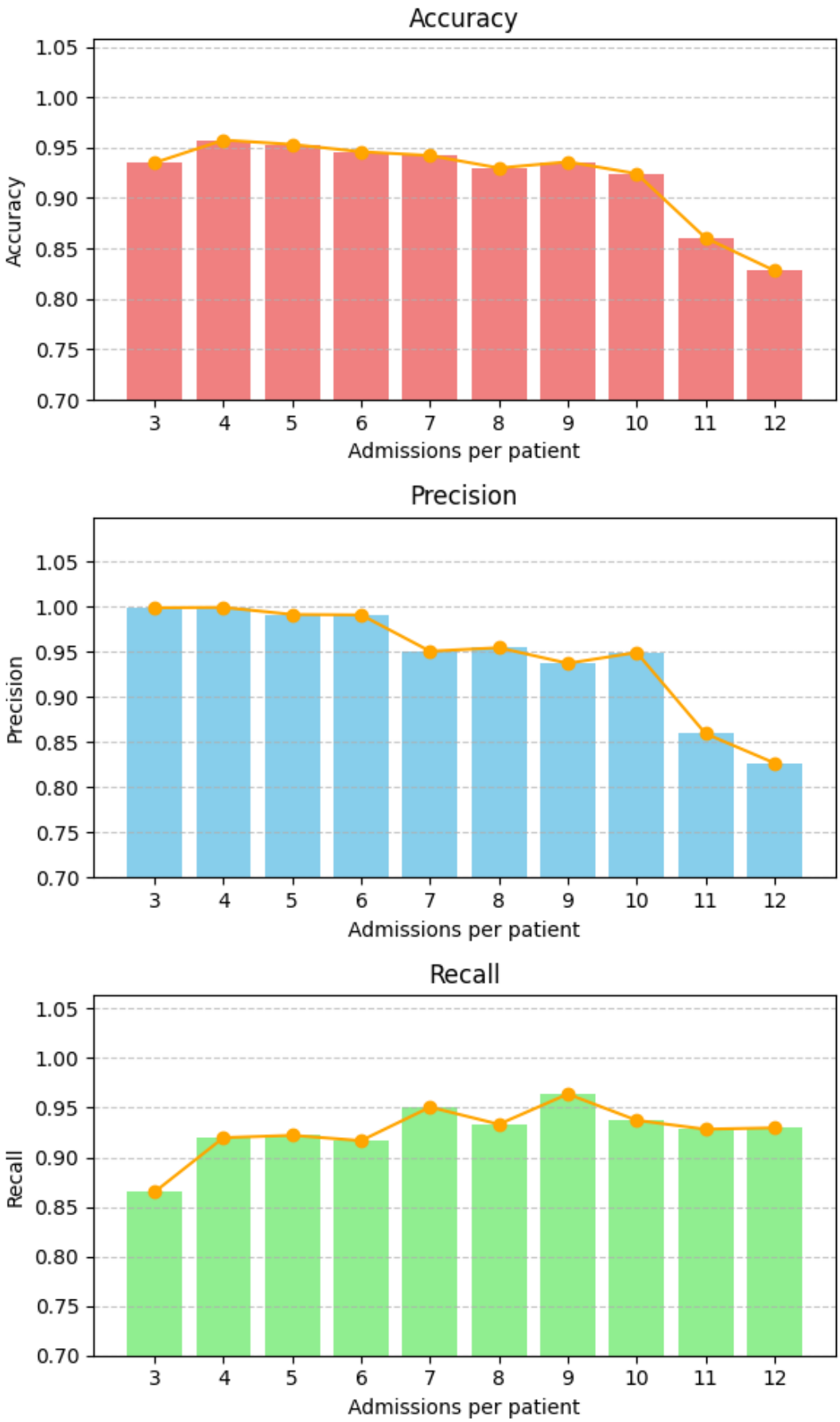

Figure 5.6: Metrics for BiLSTM

5.3 Comparison

Figures 5.2 5.4 5.6 show that a general trend for accuracy, precision and recall can be observed for each of the models and these trends stay consistent across the different architectures. Accuracy and precision start high and then start to gradually decrease. The opposite happens with recall. This could be explained by the distribution of the data. When a lower minimum number of admissions is used, this results in a smaller number sequence length but there are significantly more patients. On the other hand, the higher threshold results in a smaller number of patients that have a higher number of admissions each. When there are fewer positive instances but they are easier to detect, precision may initially be high. However, as the number of positive instances increases and becomes more challenging to detect accurately, precision may decrease. Meanwhile, recall increases because the model learns to capture a larger proportion of the positive instances, even if it makes more false positive predictions. 5.1 5.3 5.5 also have a common trend. Combining the precision and recall to get the (harmonic mean) F1 score shows that smaller sequence lengths, but many patients, are more favourable for performance.

The BiLSTM shows the best performance of the three models, though they are quite close. In terms of train time, it takes twice as long on average as the other two, so depending on how fast the models need to be trained, the RNN model could be used in place of BiLSTM. With that being said, since the medical domain benefits from having highly accurate predictions over everything else, the BiLSTM is the best of the models for this application.

5.3.1 Same category mapping

Table 5.8 shows the average metrics for each disease with the best chosen model and number of admissions set to 4.

The Average metrics are listed below:

- **Average Accuracy:** 0.9580
- **Average Precision:** 0.9924

- **Average Recall:** 0.9272
- **Average F1-score:** 0.9587

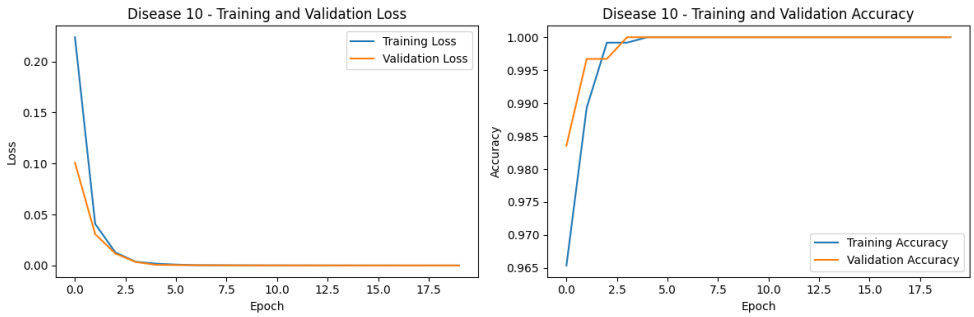

Figure 5.7: Highest accuracy disease

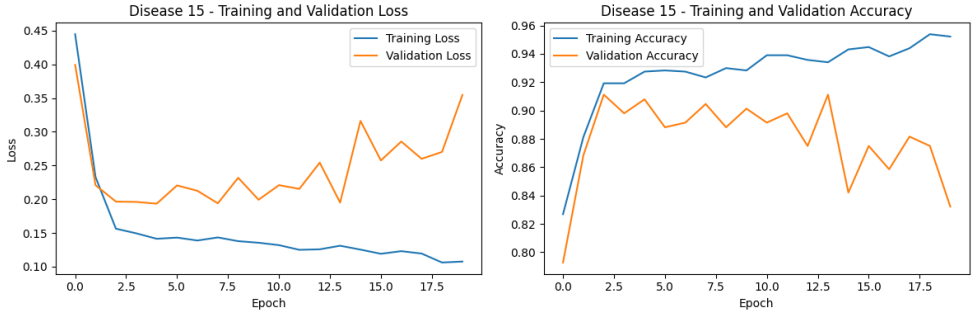

Figure 5.8: Lowest accuracy disease

Training Metrics for the the diseases with the best and worst performance according to accuracy have been plotted in figure 5.7 and figure 5.8 respectively.

5.3.2 larger category mapping

A larger list of diseases is used. The diseases are the IDC-9 Subcategories, resulting in 117 possible diseases.

- **Average Accuracy:** 0.9754
- **Average Precision:** 0.9308
- **Average Recall:** 0.8685
- **Average F1-score:** 0.8986

This shows that the model archives comparable performance. One reason it might be lower than when there were fewer categories, is the small amount of data. It could

Table 5.8: Average Metrics for Each Disease

Disease	Accuracy	Precision	Recall	F1-Score
1	0.9184	1.0000	0.8450	0.9160
140	0.9789	1.0000	0.9538	0.9763
240	0.9474	0.9933	0.9427	0.9673
280	0.9447	1.0000	0.9091	0.9524
290	0.9684	1.0000	0.9500	0.9744
320	0.9711	1.0000	0.9505	0.9746
390	0.9763	1.0000	0.9709	0.9852
460	0.9342	1.0000	0.8756	0.9337
520	0.9500	1.0000	0.9301	0.9638
580	0.9526	1.0000	0.9139	0.9550
630	1.0000	1.0000	1.0000	1.0000
680	0.9658	1.0000	0.8850	0.9390
710	0.9447	1.0000	0.8765	0.9342
740	0.9974	1.0000	0.9412	0.9697
760	1.0000	1.0000	0.9441	0.9713
780	0.9079	0.9929	0.8949	0.9414
800	0.9342	0.9831	0.8883	0.9333
V	0.9658	0.9971	0.9662	0.9814
E	0.9289	1.0000	0.8907	0.9422

also be due to the fact there are now 117 labels, which makes the classification problem more difficult and may require more training and optimisation. One way to alleviate this issue, like many others in machine learning, is to use more data.

Figure 5.9 shows the performance metrics for the same model when used with different category mappings.

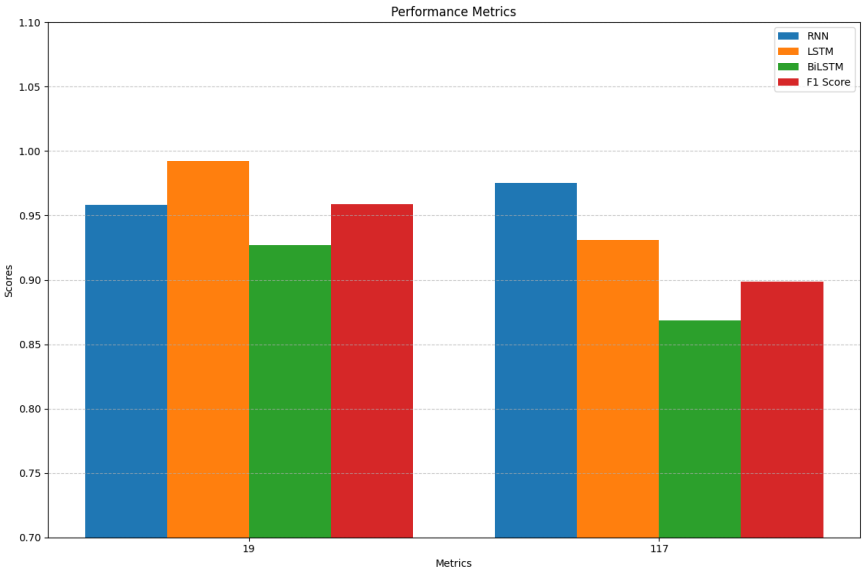

Figure 5.9: Comparison of category mappings

5.4 Result Discussion

5.4.1 Approach comparison

The first approach performed poorly as observed in table 5.1. This could be due to the lack of enough data to train any of the models used. On the other hand in the second approach, the performance was significantly better despite using the same training data. This could be due to a variety of reasons such as:

- **Data distribution:** Training a single LSTM model on the whole data might result in an imbalanced dataset or uneven distribution of classes, leading to poorer performance. By training multiple models on each disease separately, each model can focus on patterns that lead to that specific disease, potentially leading to better generalization and performance.

- **Overfitting:** Training a single LSTM model on the entire dataset may have led to overfitting, especially since the data is complex. Each model trained on a single disease is effectively trained on a smaller subset of the data, reducing the risk of overfitting and allowing each model to capture more specific patterns within the data.
- **Complexity:** The LSTM model might struggle to learn the complex patterns present in the entire dataset, especially since it contains diverse patterns. Training multiple models on smaller subsets of the data allows each model to focus on learning simpler patterns, leading to better performance overall.
- **Initialization:** Each LSTM model trained on a single disease starts with its own set of random initial weights, allowing each model to discover patterns specific to that disease and find better solutions, thus making more accurate predictions.
- **Feature importance:** Certain diseases might contain more important features or patterns than others. Training separate models allows each model to focus on the most relevant features for its disease, leading to better performance.
- **Effectively more data:** Since for each disease, each model sees all the training data for just a single target variable, there is effectively more total data that the models are being trained on. Even though the data is repeating, it could lead to more patterns being recognised and thus better predictive performance.
- **Avoid class imablance:** Not all patients will have all diseases. Some diseases are more common than others. When just a single model is trained, this could lead to biases. However since all the models see the data equally, these biases can be avoided.
- **Specialization:** Training separate models for each disease allows each model to specialize in identifying the specific patterns and features associated with that disease. This specialization can lead to better performance compared to a single model trying to learn all patterns simultaneously. The separate models can capture the interactions between different diseases more effectively. Each model can focus on the unique patterns associated with its target disease while also considering how it interacts with other diseases present in the data.
- **Scalability:** Training separate models for each disease can make the approach more scalable, especially if new diseases need to be added or if the dataset grows over time. Adding a new disease simply involves training a new model specifically for that disease, without the need to retrain a single large model on the entire dataset.

5.4.2 Runtimes

The time to train individual models is much longer than training just a single one. Average and total training times can be seen in table 5.9. The models have been tested on a system with high performance consumer hardware (NVIDIA GEFORCE RTX 4090 Graphics processing unit). GPUs are well suited for parallel computations required during neural network training, leading to faster convergence and reduced training times. They also leverage software optimisations provided by frameworks such as cuDNN for NVIDIA GPUs, and can further enhance training speed by leveraging hardware-specific optimizations.

Table 5.9: Runtimes when using GPU

Architecture	Total time (s)	Avg. time per model (s)
RNN	56.80	2.98
LSTM	91.75	4.82
BiLSTM	117.85	9.36

The models have also been tested on a system without a GPU. Training times for this can be seen in table 5.10. Note that these times have been recorded under ideal conditions, with no other processes running simultaneously, other than critical ones like the operating system.

Table 5.10: Runtimes without GPU

Architecture	Total time (s)	Avg. time per model (s)
RNN	60.68	3.19
LSTM	120.90	6.36
BiLSTM	218.93	11.52

The models have been trained for 5 epochs each. Tables 5.9 and 5.10 show that even with the lack of a graphics processing unit, training times are still reasonably fast. However, this highly depends on the category mapping provided, with a larger list of potential diseases likely taking longer to train.

5.4.3 Resource consumption

When using the system with the GPU, memory consumption was recorded and aggregated over all models. This can be seen in table 5.11. Since TensorFlow blocks the memory it will use at the beginning of execution, the information from table 5.11 indicates that a high-end GPU may not be necessary to achieve similar speeds. As long as enough VRAM is available, systems running these predictive modelling tasks should be able to keep up in terms of speed and memory efficiency.

Table 5.11: GPU memory usage in Gb

Architecture	Peak memory usage
RNN	0.4441
LSTM	1.649
BiLSTM	4.29

The memory consumptions in 5.11 suggest that RNNs typically require the least amount of GPU memory, followed by LSTMs and BiLSTMs. This means simpler model architectures may offer better memory efficiency, making them more accessible for deployment on systems with limited GPU resources. The RNN may be a viable option for quick predictions that are used as a starting point rather than final diagnoses considering its high performance.

5.4.4 Model performace

Firstly, the RNN model achieved acceptable results on average.
The LSTM performed slightly worse than the RNN .
The BiLSTM model achieved the best overall performance.

Figure 5.10 shows the performance of each model for each metric.

The RNN outperforming the LSTM was unexpected. This could be because the diagnosis history is built cumulatively. Any disease present in an admission will be present in all subsequent admissions. RNNs are naturally suited to capture sequential dependencies in data, and the cumulative history effectively provides a built-in mechanism for retaining relevant information over time. This alleviated the limitation of RNNs' lack of explicit long-term memory, enabling them to effectively model the sequential nature of the data without encountering the vanishing gradient problem that

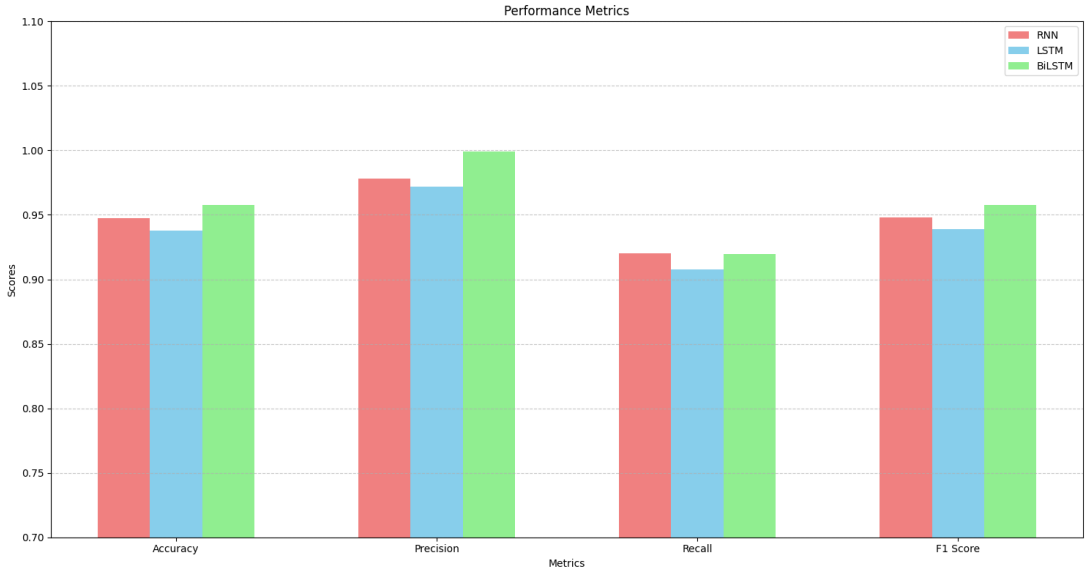

Figure 5.10: Model Comparison

LSTM architectures are designed to mitigate.

Moreover, The simpler architecture of the RNN compared to the LSTM may have played a role in preventing overfitting. LSTMs are more complex models with additional mechanisms for capturing long-term dependencies, which can lead to a higher risk of overfitting, especially in scenarios of a limited dataset. The RNN's simpler architecture may have provided a more sparing representation of the data, reducing the likelihood of overfitting and allowing it to generalize better to unseen test data. This better generalization ability of the RNN may also stem from its ability to effectively capture the underlying patterns in the data without being overly influenced by noise or irrelevant information. By focusing on the sequential nature of the data and leveraging the cumulative history, the RNN may have learned to extract and encode relevant features more efficiently, leading to improved generalization performance.

The Bidirectional LSTM was the best performing model of the three that were tested. BiLSTMs process input sequences in both forward and backward directions, allowing them to capture information from both past and future contexts. This bidirectional nature enables the model to have a more comprehensive understanding of the sequential data, incorporating information from both preceding and succeeding time steps. As a result, BiLSTMs can capture richer contextual information compared to unidirectional models like the RNN and traditional LSTM, which process sequences in only one direction. By incorporating future context, BiLSTMs are excellent at

modelling long-range dependencies in sequential data. In scenarios where long-term relationships between elements in the sequence are crucial for accurate predictions, BiLSTMs can outperform unidirectional models by effectively leveraging information from distant time steps.

Additionally, the bidirectional nature of BiLSTMs mitigates the issue of information loss that can be seen in unidirectional models, where information from one end of the sequence may not fully propagate to the other end. By processing the sequence bidirectionally, BiLSTMs ensure that information from all parts of the sequence contributes to the final prediction, resulting in more accurate models. They also generate richer feature representations by combining information from both forward and backward passes. This enables the model to capture complex patterns and relationships within the data more effectively, leading to improved generalization.

5.4.5 Further Optimisations

Even though the models are able to achieve high performance, Hyperparameters such as batch size, learning rate, training epochs, etc. can be further investigated to find performance gains. In the context of medical diagnoses, achieving higher performance in predictive models is paramount due to the critical nature of the decisions being made based on these predictions. Systematically investigating and optimizing hyperparameters enhances the performance of predictive models, which, in medical diagnosis applications, reduces the risk of mispredictions and improves patient outcomes. Continuous monitoring and refinement of hyperparameters in response to changes in data distributions or model requirements are essential for maintaining optimal performance over time.

Chapter 6

Conclusions

This chapter discusses the contributions this project has made to the domain(6.1). Next, are the achievements of the project and how they tie in with the goals set for the implementation(6.2). This is followed by a discussion of the challenges in the domain relating to data(6.3). Finally, it lists potential improvements to the work(6.4).

6.1 Contributions

The project proposes the use of keras models to predict medical diagnoses based on past history. It also makes comparisons between two distinct approaches to tackle the problem. The second approach showcases how an accurate prediction system can be built even with the lack of substantial amounts of data. Additionally, it compares different architectures for the application. The implementation includes dynamic elements such as what possible diseases to predict, and the minimum number of admissions.

6.2 Achievements

The project successfully implemented a system to predict future occurrences of diseases. Multiple model architectures such as RNN, LSTM, and BiLSTM were tested. The performance of the system was also evaluated using standard evaluation metrics. These evaluations reveal that the system archives performance high enough to be used in real-world scenarios. However, considering the sensitivity of the medical domain and the importance of accurate diagnoses, the performance may still need improvements. It also showed how changing a small aspect like a target variable makes a big

impact on performance.

6.3 Data Limitations

While the second performance showed promising results, more testing is needed to thoroughly evaluate the first approach. It may be the case that the model needs more layers to properly learn from the data. However, it is more likely that there simply isn't enough data to train a model for a task such as this.

A determining factor for the performance of any Machine Learning based method is the data. The first few weeks of the project were spent trying to find suitable data to work with. Initially, using synthetic data like [47] was investigated. However, this would present a major problem: data generation was slow. Tens of Thousands of data points would be needed, and the number of usable admissions would only decrease as the data is preprocessed. It would be infeasible to produce a dataset large enough for the proposed models to achieve good performance. Even when using MIMIC data, a significant majority of the data could not be used as these would be filtered out due to either the constraints or the model architecture needing the input data to be in a certain format. Interpolating histories by filling in admissions from their past values was also considered, however, this would lead to data imbalance and introduce biases as certain diseases would appear much more frequently.

The lack of high-quality and complete datasets is a common issue in the healthcare domain. Patients typically undergo different clinical procedures or may drop out of studies. Examples of missing data include patients lost to follow-up, partially filled-out surveys or incomplete medical records.[49] Since data used in training NNs need to be complete, most studies discard these incomplete data points.[50]

One way to combat this is to relax the constraints while preprocessing the data. This could involve imputation techniques to fill in missing values or handling missing data in a way that allows incomplete data points to be included in the training dataset. For example, missing values could be replaced with the mean, median, or mode of the feature, or by using more advanced methods such as K-nearest neighbours (KNN) or interpolation.

While relaxing constraints during preprocessing can help retain more data and improve the robustness of the model, it also introduces irregularities during training. The model learns from both complete and incomplete data, which may result in biases or inaccuracies, especially if the missing data points are not missing at random.

To conclude, it's essential to carefully consider the implications of relaxing constraints on the dataset and to evaluate the impact on the model's performance. Techniques such as cross-validation and sensitivity analysis can be used to assess the robustness of the model and identify any potential issues introduced by handling incomplete data. Additionally, domain knowledge and expert input are valuable for making informed decisions about how to handle missing data effectively while training NNs in the healthcare domain.

6.4 Further Work

Currently, the implementation is still in an early stage. Some of the proposed improvements to this project are listed below:

- **Predict date of next visit:** Currently, The models have been trained to predict what diagnoses the patient might have in their next visit, but not when that visit would occur. While logistic regression could be used for a basic prediction, a more sophisticated approach would involve building a model that learns the relationships between certain diseases and the associated time to the next admission. This could involve techniques such as time-series forecasting, allowing healthcare professionals to better prepare for patient needs and allocate resources effectively.

- **Personalized Treatment Recommendations:** Expanding the predictive models to provide personalized treatment recommendations based on patient characteristics and historical data can further support healthcare decision making. The system could suggest tailored interventions, medication regimens, or lifestyle modifications for individual patients, optimizing treatment outcomes and patient satisfaction.

- **GUI** Currently, the predictions are made through either a jupyter notebook or through a command line interface (CLI). A GUI would provide a more intuitive and visually appealing way to interact with the system, potentially lowering

the barrier to entry for users who may find CLI usage daunting. Features such as interactive data visualization and easy navigation could further improve user experience.

- **Database integration** Currently, the preprocessing stage expects data in CSV format, which may require an additional step if the data is stored in an SQL database. Integrating database functionality directly into the application can streamline this process, allowing users to directly access and preprocess data from databases without the need for manual conversion to CSV. Additionally, database integration opens up possibilities for more selective data usage during training, as users can query and filter data based on specific criteria, leading to more targeted diagnosis predictions.

- **EHR integration** Integrating the prediction software with existing Electronic Health Records (EHR) systems can streamline workflows and improve data accessibility. By directly interfacing with EHR systems, the software can seamlessly access patient records, update predictions in real-time, and facilitate data exchange between different healthcare applications, ensuring continuity of care

- **Automated training** Adding to the previous point, integrating database functionality into the application could enable automated training of the models. This automation can be triggered periodically (e.g., every few days) or in response to events such as new admissions or patient registrations. Automated training ensures that the models are regularly updated with the latest data, improving their accuracy and relevance over time. Furthermore, automated training can accommodate changes such as the addition of new diagnosis categories, ensuring that the models remain up-to-date and adaptable to evolving healthcare needs.

By implementing these improvements, the project can enhance its predictive capabilities, user experience, and operational efficiency, ultimately leading to better support for healthcare professionals in diagnosis prediction and patient care.

References

[1] IBM, "What are recurrent neural networks?." https://www.ibm.com/topics/recurrent-neural-networks.

[2] databasecamp, "Long short-term memory networks (lstm)- simply explained!." https://databasecamp.de/en/ml/lstms, 2022.

[3] C. Salisbury, "Multimorbidity: redesigning health care for people who use it," *The Lancet*, vol. 380, no. 9836, pp. 7–9, 2012.

[4] K. Barnett, S. W. Mercer, M. Norbury, G. Watt, S. Wyke, and B. Guthrie, "Epidemiology of multimorbidity and implications for health care, research, and medical education: a cross-sectional study," *The Lancet*, vol. 380, no. 9836, pp. 37–43, 2012.

[5] T. Vos, S. S. Lim, C. Abbafati, K. M. Abbas, M. Abbasi, M. Abbasifard, M. Abbasi-Kangevari, H. Abbastabar, F. Abd-Allah, A. Abdelalim, *et al.*, "Global burden of 369 diseases and injuries in 204 countries and territories, 1990–2019: a systematic analysis for the global burden of disease study 2019," *The lancet*, vol. 396, no. 10258, pp. 1204–1222, 2020.

[6] B. L. Ryan, K. Bray Jenkyn, S. Z. Shariff, B. Allen, R. H. Glazier, M. Zwarenstein, M. Fortin, and M. Stewart, "Beyond the grey tsunami: a cross-sectional population-based study of multimorbidity in ontario," *Canadian Journal of Public Health*, vol. 109, pp. 845–854, 2018.

[7] S. R. Chowdhury, D. Chandra Das, T. C. Sunna, J. Beyene, and A. Hossain, "Global and regional prevalence of multimorbidity in the adult population in community settings: a systematic review and meta-analysis," *eClinicalMedicine*, vol. 57, p. 101860, 2023.

[8] L. P. Fried, K. Bandeen-Roche, J. D. Kasper, and J. M. Guralnik, "Association of comorbidity with disability in older women: The women's health and aging study," *Journal of Clinical Epidemiology*, vol. 52, no. 1, pp. 27–37, 1999.

[9] E. A. Bayliss, M. S. Bayliss, J. E. Ware, Jr, and J. F. Steiner, "Predicting declines in physical function in persons with multiple chronic medical conditions: what we can learn from the medical problem list," *Health Qual Life Outcomes*, vol. 2, p. 47, sep 2004.

[10] A. Ryan and et al., "Multimorbidity and functional decline in community-dwelling adults: a systematic review," *Health and quality of life outcomes*, vol. 13, p. 168, 2015.

[11] A. Menotti and et al., "Prevalence of morbidity and multimorbidity in elderly male populations and their impact on 10-year all-cause mortality: The fine study (finland, italy, netherlands, elderly)," *Journal of clinical epidemiology*, vol. 54, no. 7, pp. 680–686, 2001.

[12] M. Fortin, H. Soubhi, C. Hudon, E. A. Bayliss, and M. van den Akker, "Multimorbidity's many challenges," *BMJ*, vol. 334, pp. 1016–1017, May 2007.

[13] B. P. Nunes and et al., "Multimorbidity and mortality in older adults: A systematic review and meta-analysis," *Archives of gerontology and geriatrics*, vol. 67, pp. 130–138, 2016.

[14] M. Hall and et al., "Multimorbidity and survival for patients with acute myocardial infarction in england and wales: Latent class analysis of a nationwide population-based cohort," *PLoS medicine*, vol. 15, no. 3, p. e1002501, 2018.

[15] V. Walker and et al., "Effect of multimorbidity on health-related quality of life in adults aged 55 years or older: Results from the su.vi.max 2 cohort," *PloS one*, vol. 11, no. 12, p. e0169282, 2016.

[16] M. Hunger and et al., "Multimorbidity and health-related quality of life in the older population: results from the german kora-age study," *Health and quality of life outcomes*, vol. 9, p. 53, 2011.

[17] P. Arokiasamy and et al., "The impact of multimorbidity on adult physical and mental health in low- and middle-income countries: what does the study on

global ageing and adult health (sage) reveal?," *BMC medicine*, vol. 13, p. 178, 2015.

[18] R. O'Brien and et al., "The 'everyday work' of living with multimorbidity in socioeconomically deprived areas of scotland," *Journal of comorbidity*, vol. 4, pp. 1–10, 2014.

[19] F. S. Mair and C. R. May, "Thinking about the burden of treatment," *BMJ (Clinical research ed.)*, vol. 349, p. g6680, 2014.

[20] A. Townsend and et al., "Managing multiple morbidity in mid-life: a qualitative study of attitudes to drug use," *BMJ (Clinical research ed.)*, vol. 327, no. 7419, p. 837, 2003.

[21] M. Rosbach and J. S. Andersen, "Patient-experienced burden of treatment in patients with multimorbidity - a systematic review of qualitative data," *PloS one*, vol. 12, no. 6, p. e0179916, 2017.

[22] C. A. M. Paddison and et al., "Why do patients with multimorbidity in england report worse experiences in primary care? evidence from the general practice patient survey," *BMJ open*, vol. 5, no. 3, p. e006172, 2015.

[23] R. Schulz and S. R. Beach, "Caregiving as a risk factor for mortality: the caregiver health effects study," *JAMA*, vol. 282, no. 23, pp. 2215–2219, 1999.

[24] R. Schulz and P. R. Sherwood, "Physical and mental health effects of family caregiving," *The American journal of nursing*, vol. 108, no. 9 Suppl, pp. 23–27, 2008.

[25] R. D. Adelman and et al., "Caregiver burden: a clinical review," *JAMA*, vol. 311, no. 10, pp. 1052–1060, 2014.

[26] B. Mason and et al., "'my body's falling apart.' understanding the experiences of patients with advanced multimorbidity to improve care: serial interviews with patients and carers," *BMJ supportive & palliative care*, vol. 6, no. 1, pp. 60–65, 2016.

[27] P. Bower and et al., "Multimorbidity, service organization and clinical decision making in primary care: a qualitative study," *Family practice*, vol. 28, no. 5, pp. 579–587, 2011.

[28] R. O'Brien and et al., "An 'endless struggle': a qualitative study of general practitioners' and practice nurses' experiences of managing multimorbidity in socioeconomically deprived areas of scotland," *Chronic illness*, vol. 7, no. 1, pp. 45–59, 2011.

[29] E. Søndergaard and et al., "Problems and challenges in relation to the treatment of patients with multimorbidity: General practitioners' views and attitudes," *Scandinavian journal of primary health care*, vol. 33, no. 2, pp. 121–126, 2015.

[30] C. Sinnott and et al., "Gps' perspectives on the management of patients with multimorbidity: systematic review and synthesis of qualitative research," *BMJ open*, vol. 3, no. 9, p. e003610, 2013.

[31] D. M. Zulman and et al., "Quality of care for patients with multiple chronic conditions: the role of comorbidity interrelatedness," *Journal of general internal medicine*, vol. 29, no. 3, pp. 529–537, 2014.

[32] L. G. Glynn and et al., "The prevalence of multimorbidity in primary care and its effect on health care utilization and cost," *Family practice*, vol. 28, no. 5, pp. 516–523, 2011.

[33] C. Salisbury and et al., "Epidemiology and impact of multimorbidity in primary care: a retrospective cohort study," *The British journal of general practice : the journal of the Royal College of General Practitioners*, vol. 61, no. 582, pp. e12–e21, 2011.

[34] T. Lehnert and et al., "Review: health care utilization and costs of elderly persons with multiple chronic conditions," *Medical care research and review : MCRR*, vol. 68, no. 4, pp. 387–420, 2011.

[35] E. A. Guthrie and et al., "Depression predicts future emergency hospital admissions in primary care patients with chronic physical illness," *Journal of psychosomatic research*, vol. 82, pp. 54–61, 2016.

[36] R. A. Payne and et al., "The effect of physical multimorbidity, mental health conditions and socioeconomic deprivation on unplanned admissions to hospital: a retrospective cohort study," *CMAJ : Canadian Medical Association journal = journal de l'Association medicale canadienne*, vol. 185, no. 5, pp. E221–E228, 2013.

[37] P. Kasteridis, A. Street, M. Dolman, L. Gallier, K. Hudson, J. Martin, and I. Wyer, "Who would most benefit from improved integrated care? implementing an analytical strategy in south somerset," *Int J Integr Care*, vol. 15, p. e001, 2015.

[38] S. Crystal and et al., "Out-of-pocket health care costs among older americans," *The journals of gerontology. Series B, Psychological sciences and social sciences*, vol. 55, no. 1, pp. S51–S62, 2000.

[39] W. Hwang and et al., "Out-of-pocket medical spending for care of chronic conditions," *Health affairs (Project Hope)*, vol. 20, no. 6, pp. 267–278, 2001.

[40] I. McRae and et al., "Multimorbidity is associated with higher out-of-pocket spending: a study of older australians with multiple chronic conditions," *Australian journal of primary health*, vol. 19, no. 2, pp. 144–149, 2013.

[41] L. Picco, E. Achilla, E. Abdin, S. A. Chong, J. A. Vaingankar, P. McCrone, H. C. Chua, D. Heng, H. Magadi, L. L. Ng, M. Prince, and M. Subramaniam, "Economic burden of multimorbidity among older adults: impact on healthcare and societal costs," *BMC Health Serv Res*, vol. 16, p. 173, 2016.

[42] AWS, "What is rnn?." `https://aws.amazon.com/what-is/recurrent-neural-network/`.

[43] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[44] C. Olah *et al.*, "Understanding lstm networks," 2015.

[45] R. Alanazi, "Identification and prediction of chronic diseases using machine learning approach," *J Healthc Eng*, vol. 2022, p. 2826127, 2022.

[46] S. Q. Nate Gruver, Marc Finzi and A. G. Wilson, "Large language models are zero shot time series forecasters," in *Advances in Neural Information Processing Systems*, 2023.

[47] J. Enguehard, D. Busbridge, A. Bozson, C. Woodcock, and N. Y. Hammerla, "Neural temporal point processes for modelling electronic health records," 2020.

[48] A. Johnson, L. Bulgarelli, T. Pollard, S. Horng, L. A. Celi, and R. Mark, "Mimic-iv (version 2.2)." PhysioNet, 2023.

[49] M. Marino, J. Lucas, E. Latour, and J. D. Heintzman, "Missing data in primary care research: importance, implications and approaches," *Fam Pract*, vol. 38, no. 2, pp. 200–203, 2021.

[50] Y.-W. Chang, L. Natali, O. Jamialahmadi, S. Romeo, J. B. Pereira, and G. Volpe, "Neural network training with highly incomplete datasets," 2021.

Appendix A

Code Listings

```
def get_target(data, disease):
    train_admissions = data.shape[1] -1
    # print(train_admissions)
    a = data[:, :train_admissions, :] # first (num_admissions -1)
    b = []
    # print(data.shape[0])
    for i in range(data.shape[0]):  # number of patients
        # a.append(data[i][:num])  # first (num_admissions -1)
        b.append(data[i][-1][disease])   # final admission
        # print(disease)

    return np.array(a), np.array(b)
```

Listing A.1: Function to get target variable

```
def train_models(iterations):
    losses = []
    accuracies = []
    models = []
    runtimes = []

    for i in range(0,iterations):

        name = "Disease_"+ str(i)
        print(f"Disease {i+1}(code {diseases[i]})")

        X_train , y_train = get_target(train_data, i)
        X_test , y_test = get_target(test_data, i)

```

```
        X_train = X_train.astype('float32')
        y_train = y_train.astype('float32')
        X_test = X_test.astype('float32')
        y_test = y_test.astype('float32')

        model = compile_model(name, X_train)


        start_time = time.time()
        history = model.fit(X_train, y_train, epochs=20, batch_size=64, validation_split=0.2)
        finish_time = time.time() - start_time

        runtimes.append(finish_time)

        models.append(model)

        loss, accuracy = model.evaluate(X_test, y_test)
        losses.append(loss)
        print("Test Loss:", loss)

        accuracies.append(accuracy)
        print("Test Accuracy:", accuracy)
        print()

        # Plot Accuracy and Loss
        plt.figure(figsize=(12, 4))

        plt.subplot(1, 2, 1)
        plt.plot(history.history['loss'], label='Training Loss')
        plt.plot(history.history['val_loss'], label='Validation Loss')
        plt.title(f'Disease {i} - Training and Validation Loss')
        plt.xlabel('Epoch')
        plt.ylabel('Loss')
        plt.legend()

        plt.subplot(1, 2, 2)
        plt.plot(history.history['accuracy'], label='Training Accuracy')
        plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
```

```
        plt.title(f'Disease {i} - Training and Validation Accuracy')
        plt.xlabel('Epoch')
        plt.ylabel('Accuracy')
        plt.legend()

        plt.tight_layout()

        # Save the individual disease plots
        plot_filename = f'./plots/Disease_{i}_Training_Validation.
    png'
        plt.savefig(plot_filename)
        plt.close()


    print("Average loss:", np.average(loss))
    print("Average accuracy:", np.average(accuracies))
    print(f"Total Training time: {np.sum(runtimes)} and Average
    training time = {np.mean(runtimes)}")

    return models


iterations = train_data.shape[2]
print(iterations)

models = train_models(iterations)
```

Listing A.2: Function to train models

```
def predict_diseases(models, data):

    if len(data.shape) == 2:
        print("Reshaping data into compatible format.")
        data = np.reshape(data, (1, data.shape[0], data.shape[1]))
        print(data.shape)

    all_predictions = [[] for _ in range(data.shape[0])]
    iterations = data.shape[-1]

    for i in range(iterations):
        # Use the first n-1 admissions to make prediction
        X_test , _ = get_target(data, i)
        X_test = X_test.astype('float32')
```

```
15
16          model = models[i]
17
18          predictions = model.predict(X_test)
19          rounded_predictions = np.round(predictions).astype(int)
20          all_predictions = np.concatenate((all_predictions,
    rounded_predictions), axis=1).astype(int)
21
22      return all_predictions
```

Listing A.3: Function to make predictions