1 Introduction

Ulcerative colitis (UC) refers to a subdivision of inflammatory bowel disease (IBD) which causes prolonged and incurable irritation and ulcers in the digestive tract. Common symptoms of UC include diarrhea with blood, abdominal pain, and unexplainable fever. Meanwhile, osteoporosis, dehydration, and colon cancer can be developed with UC as complications. Available studies show that approximately 1 in 400 North American lives under the impact of UC [1]. While age, race, and weight are reported as risk factors, the exact cause of UC still remains unknown. Notably, UC has a variable course, as symptoms can erupt, disappear and relapse for the whole life. Some patients have a mild course with long periods of remission, while others suffer from much more persistent and aggressive illnesses [2]. Therefore, the research of UC’s temporal progression proliferates.

UC progression can broadly include remission status and severity level. Remission refers to a status with little to no symptoms, and severity is determined by the amount of inflammation in the digestive tract. The severer the disease is, the more aggressive the treatments need to be; some excessive therapies like colectomy can have unpleasant side effects [3]. Lacking consideration to currently available treatments does not only increase patients’ morbidity and mortality but also presses unnecessary financial burdens [2]. If the UC progression becomes predictive, clinicians can develop personalized therapeutic strategies guiding optimal medical care.

Although the cause and its progression of UC is currently uncharted, people have learned that the gut microbiome is a key link between genetic susceptibility and UC progression [4]. Gut microbiomes refer to the micro-organisms inhabiting the intestinal tract. The composition of these microbiomes can be quantified and profiled by gene sequencing techniques. Thus, more rigorous classifications models have been developed, which utilize gut microbiome profiles to predict the UC progression and inform intervention with novel treatments.

Longitudinal microbiome data have been widely used in studying disease prognosis and microbial dynamics within an ecosystem such as the gut and cavity. [5]. Such data track the microbiome taxonomic profiles of the same patient at different time points. Constructing predictive models from longitudinal microbiome samples is not easy since extracting information from time trends is a challenge. Traditional classification models like logistic regression (LR), K-nearest neighbours (KNN), or decision trees fail to handle the dimension of time. Recurrent Neural Network (RNN) is a deep learning model designed for learning temporal behaviour for a time-series event. Nevertheless, its performance can be restricted when the time sequence is long due to the issue of vanishing gradients. Hence, an improved version of RNN, namely Long Short-Term Memory (LSTM) network, has been invented to learn temporal behaviour and overcome the vanishing gradient problem. Another worry with microbiome data is a large number of variables. Microbiomes usually have intricate compositions, significantly boosting the dimension of taxonomic profiles. This causes the classifiers to be inefficient and overfitting. Therefore, people need approaches to select features and allow them to support classification wisely.

We aim to develop LSTM-related neural networks to predict UC progression (remission and severity) from longitudinal gut microbiome profiles. Based on different modelling strategies, we respectively construct a basic LSTM network, an encoder-decoder LSTM network, and a convolutional LSTM network. We also implement feature selection techniques with a key consideration in a deep learning method, namely auto-encoder. For assessment purposes, we introduce traditional classifiers and feature selection methods as baselines. We validate each model’s performance on predicting the UC progression at the last time point. We hypothesize that the end-to-end LSTM networks with encode-decoder architecture for feature reduction outperform baseline classifiers with feature selection procedure directly (end-to-end / one-step models or two-step models).

2 Data Material and Methods

**2.1 Data Specification & Exploration**

The data were longitudinally collected from 405 pediatric, new-onset, and treatment-naive UC patients. These patients are observed for one year upon the initiation of a 5-ASA mesalamine or corticosteroid treatment. Fecal samples or rectal biopsies were collected at baseline (week 0, prior to treatment) and 3 follow-up time points (4, 12, and 52 weeks after treatment initiation) [4]. Once a sample of a given patient is obtained at a specific time point, its microbial taxonomy is analyzed and profiled into 1015 numeric variables via 16S rRNA gene amplicon sequencing. These microbial features are potential predictors. Additionally, each patient’s disease progression is recorded at each time point. Remission status is documented as a binary indicator, while severity is measured in four levels: inactive, mild, moderate, and severe. These two labels are response variables. Furthermore, the data include some demographical features like age, gender, and race, which are also considered when selecting predictors.

We plot distributions of the UC progression at each time point in Fig 2.1 a&b. Both the severity level and remission status show a tendency of relief at the baseline time point, indicating that the treatments are helpful. However, their distributions tend to be constant without any trend for the rest of the follow-up. We also draw these distributions for different classes of gender and race in Fig 2.1 c-f, but see no pattern across these groups. We notice that the classes of race are imbalance, as the majority are white people. Furthermore, we plot the number of non-zero observations for each microbial profile in Fig 2.2, because we observe lots of zeros when first glancing the data. This plot manifests that the most of the microbiome features are sparsely distributed, which heralds the need of feature selection.



Fig 2.1 The distribution of label



Fig2.2 Number of non-zero observations for each microbiome profile

**2.2 Method**

**2.2.1 Data Pre-processing and Quality Check**

There are two issues related to the raw data. First, some patients have missing samples at some specific time points. For example, one patient can have collected microbiome profiles or samples at the 2nd and 3rd time points, but completely lost the observations on the 1st and the 4th time points. It leads to an uneven number of timepoints along the subjects’ longitudinal timeline, which limits the time length that can be traced back by the proposed LSTM network (discussed below). Data imputation is difficult to carry out since those samples have no measurements on all more than 1000 OTU features and their outcomes. Hence, we delete the subjects whose time series is incomplete. Second, some patients have two samples at one time point. One is fecal sample and the other one comes from rectal biopsies. In such cases, we drop the biopsy sample since the faseces is the major sample type and we want to keep the data as consistent as possible.

Subsequently, we make the data adaptive to the deep learning models. First, we code each multi-class categorical predictor into multiple binary indicators. Second, we normalize and standardize the numeric features so that they have the same scale and distribution. Unscaled input variables result in a slow or unstable learning process for neural networks and distance-based traditional models. Finally, we index the microbial profiles using ‘m.1’ to ‘m.1015’ and rename the remission status and severity level as ‘label1’ and ‘label2’ to improve the conciseness.

The processed data contains 432 samples from 108 patients measured at 4 time points. Each sample includes values from 1028 predictors and 2 response labels. Therefore, the large P (number of predictors) and small N (sample size) is a characteristic of the data.

2.2.2 Modelling Strategy

When performing classification tasks, high-dimensional data can negatively affect predictive models. The existence of irrelevant or redundant predictors does not only raise the risk of overfitting, but also lowers the efficiency of learning. Thus, it is critical to seek out a subset of predictors that captures as much information from the data as possible.

Feature selection refers to a model simplification process of filtrating the top relevant features through specifically designed algorithms. This technique cooperates with classifiers under two strategies. It can be either implemented in the nature of the classifier or performed separately from the classifier. Models constructed in the former way are termed as one-step models, and models resulted from the latter way are called two-step models. For instance, a LASSO regression is a one-step model while a regression with AIC stepwise selection is counted as a two-step model.

2.2.3 Two-step models

For two step models, the simplest feature selection method is removing low-variance features. It is to delete those features whose sample variance is lower than a certain threshold. A low variance indicates that the feature is nearly constant. Such features provide very limited information for discrimination. Principal components analysis (PCA) is another baseline data compression technique. It first represents the data using linear combinations, then projects the data to a lower dimension space using orthogonal transformations.

Auto-encoders are unsupervised neural networks which are capable of learning compressed latent representations of data. Its structure is shown in Fig2.3. It has a symmetric architecture composed of three layers: one input layer, one hidden layer, and one output layer. The number of neurons in the first and the third layers both equal to the number of raw features. Comparatively, the hidden layer consists of a reduced number of neurons. The mapping from the input layer to the hidden layer is the process of encoding, which forces a condensed knowledge representation of the original input. The transformation from the hidden layer to the output layer is a step of reconstructing the input, which is called decoding. These two processes are represented using the following equations:

(2.1)

(2.2)

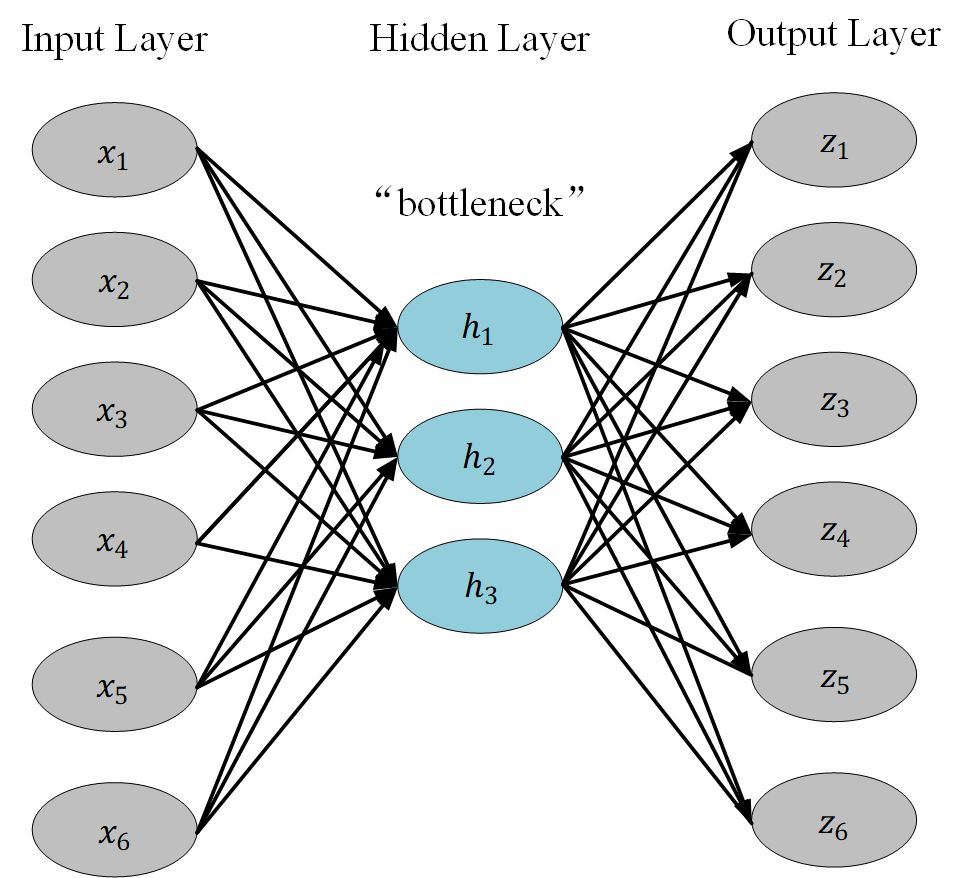


Fig2.3 The structure of Auto-encoder

The variable represents the raw data, which are treated as nodes in the input layer. It is encoded to a dimension-reduced vector (Eq 2.1), which represents the nodes from the hidden layer. Then the vector is decoded to an output vector that shares the same dimension as (Eq 2.2). Here , , , are respectively the weight matrices and bias vectors in the encoding and decoding steps, and the function is the tanh activation function.

The learning purpose is to reconstruct data as similar as possible to the raw data . Hence, the model is trained by comparing the input and output using the following loss function:

(2.3)

When training is completed, we discard the output layer and use the hidden layer as our compressed data.

However, in traditional auto-encoders people need to manually set the hidden layer’s dimension even in lack of experience. Sparse auto-encoder provides an alternative architecture. It allows the hidden layer to have a greater dimension than the input data by only activating a small number of neurons. The structure is shown in Fig2.4. The network is trained to find an optimal level of activation within the hidden layer. Therefore, we add a Kullback-Leibler (KL) divergence term in the loss function to panelize activations:

(2.4)

(2.5)

Here represents the average activation of neuron in the hidden layer over all data points, is the expected activation, is the weight of the activation penalty, and represents the total number of neurons in the hidden layer. When training is finished, the partially activated neurons are extracted from the hidden layer and passed to the classifiers for predicting the UC progression.

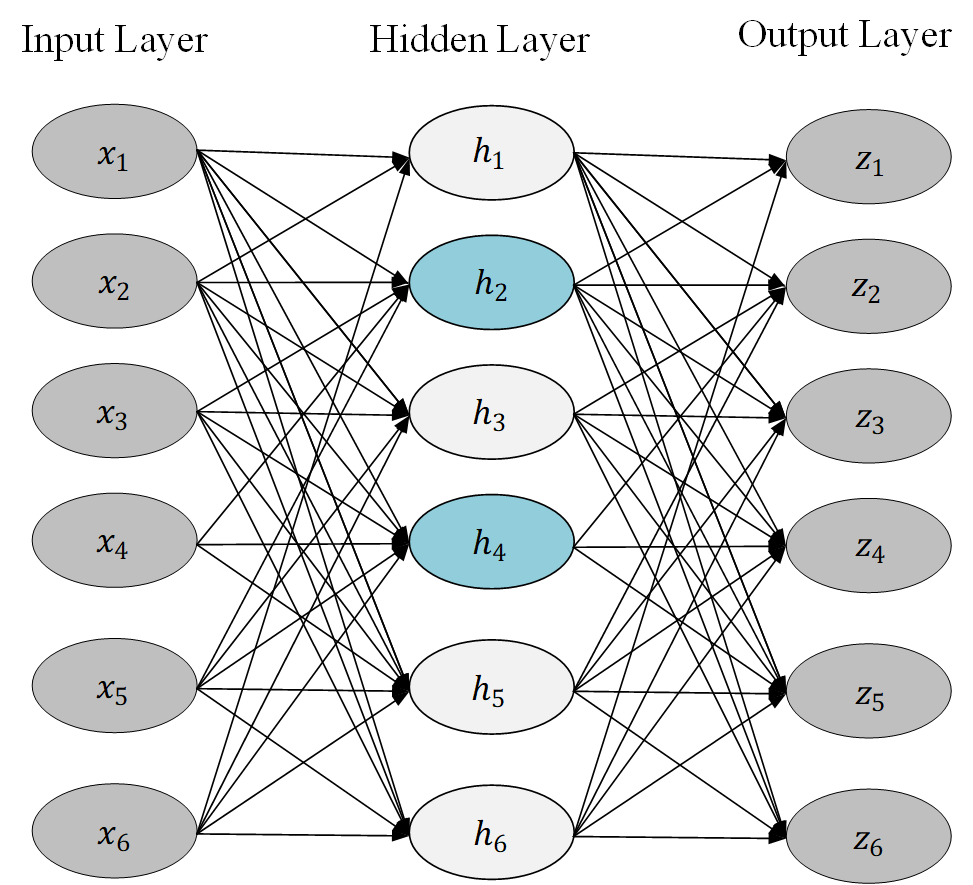


Fig 2.4 The structure of Sparse auto-encoder

After the feature selection step is completed, the extracted data in the new feature space are fed to classifiers. Our baseline classifiers include K nearest neighbours (KNN), support vector machines (SVM), and decision trees. KNN assigns an observation with the class which is the majority among its neighbours in the feature space. Its performance can be surprisingly good given its simple mechanism. SVM works by seeking hyperplanes in the feature space that separate data from different classes as accurately as possible. SVM is well known for its capability to handle both linear and non-linear classifications. Decision tree is a tree-based method that split the feature space in a recursive way using one feature at a time. One remarkable characteristic of decision tree is that the space segmentation can be visualized by a tree-like structure.

However, a shortcoming of these baseline models is the deficiency in tracking time trends involved in a longitudinal data set, as they assume time-dependency for all features. Recurrent Neural Networks (RNN) and Long Short-Term Memory Neural Networks (LSTM) are representative techniques for capturing temporal relationships.

RNNs are neural networks with loops, allowing information to transfer over time as shown in Fig2.5. In each iteration, the RNN receives an input X to generate an output O, then stores the current information in W before transferring it to the next time stage. Hence, the RNN can utilize the historical information when giving predictions. In actual engineering, the current state can be related to some long time ago information. For such cases the RNN may encounter the issue of vanishing gradient. It occurs since the derivative of the RNN’s activation function is always less than one, leading to a nearly zero loss value in backpropagation. Fortunately, LSTM networks overcome it by incorporating gates into the state dynamics.

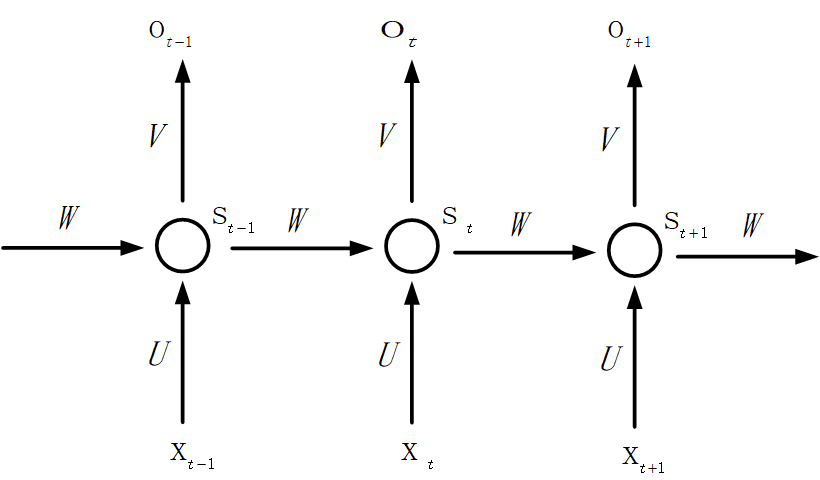


Fig.2.5 The structure of RNN

The LSTM network could be framed as a chain of repeating modules where each module corresponds to a time point. A detailed structure is shown in Fig2.6. The key component of the chain is the cell state (), which stores historical information. At each time point, the information gets updated and transmitted under the interaction of 3 gates: forgetting gate, input gate, and output gate. Every one of them is a sigmoid neural network layer.

Each time a new set of predictor values () is input to the LSTM network. The forgetting gate first decides what information to withdraw from the cell state. It generates a vector of zeros and ones () by looking at the previous output and the current input . The value ones represent retention while the value zeros mean removal.

(2.6)

Subsequently, the input gate generates a vector () of zeros and ones, which control the pass of the new information. Then a tanh layer produces a vector containing new candidate values that could be added to the cell state.

(2.7)

(2.8)

Next, we point-wisely multiply the old cell state () by , deleting the information we decided to forget. Afterwards we add it with the point-wise product of and , importing the new information we decided to import. Now the cell state has been updated.

(2.9)

After that, a vector is given by the output gate, determining which updated information will be involved in the current output.

(2.10)

Finally, we put the updated cell state through a tanh layer and point-wisely multiply it by . The resulted is our output for time .

(2.11)

This output is then fed to a dense layer with a SoftMax activation function and with the dimension of the number of classes (2x1 for label1 and 4x1 for label2). Hence each entry can represent an output probability for the corresponding class. The class with the highest probability will be assigned as the predicted class.

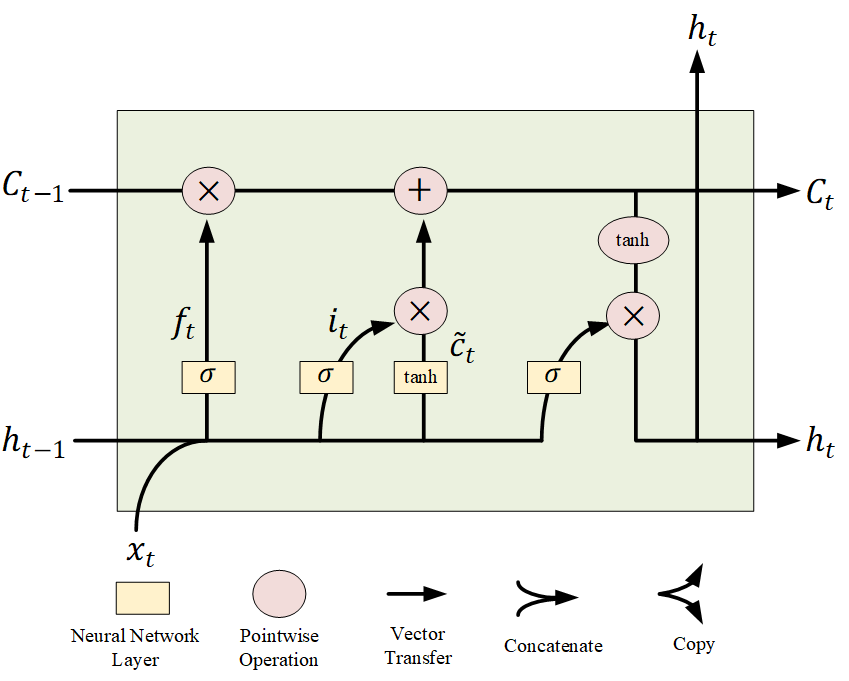


Fig2.6 The structure of LSTM

2.2.4 One-step models

The two-step modelling comes with deficiencies. First, it lowers the learning efficiency, as the feature selection and classification are performed by two separate models which are trained independently. Second, we need to try different combinations of feature selection techniques and classifiers, which is heavily time-consuming. Comparatively, one-step models can carry out feature selection and prediction simultaneously. A classic example is the LASSO-regularized logistic regression, which is one of the baseline methods in this work. LASSO selects predictors within a logistic regression by forcing some coefficients to become 0, as it imposes an L1-norm constraint on the coefficient magnitudes while maximizing the likelihood function. However, the logistic regression cannot handle time series with undesirable running time as another issue. Hence it is in need to introduce one-step modelling techniques for the proposed LSTM network.

The encoder-decoder LSTM network is a modified version of the basic LSTM model, which adds the function of feature selection. As shown in Fig 2.7, it is composed of three parts: an encoder, a decoder, and a classifier. The encoder is implemented using one LSTM layer and one drop-out layer. The former takes the raw data and maps it to a dimension-reduced vector, creating an internal representation of the data. Afterwards the latter randomly sets some entries of the output vector to 0, which helps prevent overfitting. The decoder has similar components but adding a ‘repeat vector’ layer at the front end. The ‘repeat vector’ layer can make the encoder output dimension to match with the size of the decoder input. Then an LSTM layer is set to learn the temporal dependency from the compressed data. A dropout layer is used again to keep off overfitting. After decoding, the classification is completed through a dense layer with a SoftMax activation function.

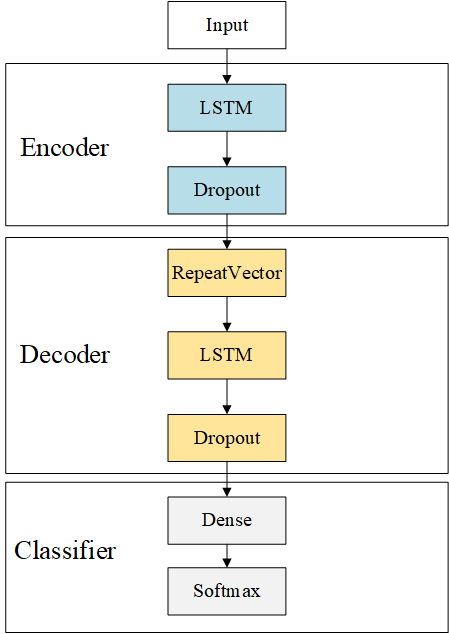


Fig 2.7 The structure of The encoder-decoder LSTM

An alternative one-step LSTM model is the convolutional long short-term memory network (CNN-LSTM). It utilizes convolutional neural network (CNN) layers for feature selection combined with LSTMs to support classification [6]. As shown in Fig 2.8, the CNN-LSTM shares a similar structure with the encoder-decoder LSTM. The only difference is that a CNN layer is used as the encoder rather than using the LSTM layer. The CNN is much more computationally efficient since it uses spatial convolution and pooling operations and performs parameter sharing [7]. It has shown success in the fields of speech recognition and natural language processing.

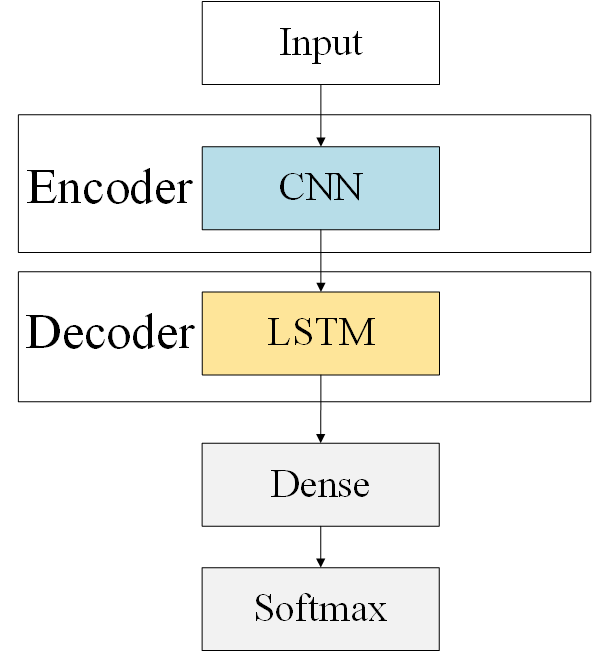


Fig 2.8 The structure of The CNN-LSTM

2.2.5 Model Evaluation

Each feature selection and classifier combination are validated using split set approach. The patients are randomly split into a training set and a validation set under a ratio of 7:3. Then the model is trained using the training set and tested on the validation set. According to the randomness in the segmentation, this procedure is repeated 10 times and the test result is recorded each time. The final performance is estimated by averaging the results. We choose this method since it is easy to understand and interpret. Additionally, it can moderate the overfitting within the performance estimates since the training set and validation set are separate in each iteration.

2.2.6 Performance Metric

In binary prediction, the model estimates the probability of belonging to one of the classes for each subject. Then the observation is classified by comparing the estimated probability with a threshold probability. Higher thresholds give better specificity but worse sensitivity, and vice versa. Thus, we plot the sensitivity and 1-specificity at all possible threshold values in a curve called ROC curve. The area under this curve represents the probability that the model correctly orders the risks of two individuals with and without an event (See from eq2.12). We use this area to evaluate the label 1 (remission) prediction.

(2.12)

For the label2 (severity) prediction, we introduce a new metric namely weighted F-1 score. The F-1 score combines precision and recall into a single metric to overcome the trade-off between these two metrics:

(2.13)

For multi-class prediction, F-1 score is the arithmetic mean of the per-class F-1 scores. However, naively averaging the per-class scores cannot account for label imbalance. Alternatively, we can weight the measurement of each class by the number of samples from that category. It is shown in the following formula:

(2.14)

2.3 Software

The data is processed in R, as R has abundant built-in functions for data manipulation without loading too many extra packages. The prediction models are constructed, trained, and validated using scikit-learn, keras, and tensorflow packages in Python. Python is chosen since one can easily find Python-based open resources of deep learning models. The figures are made using Orgin since it provides convenience in adding annotations.