Hospital Readmission Prediction of ICU Patients Using Deep Learning Algorithms

Saumil Maheshwari
Department of Information Technology)
ABV-Indian Institute of Information Technology and Management
Gwalior, India
saumlimaheshwari@yahoo.co.in

Shivam Sinha
Department of Information Technology)
ABV-Indian Institute of Information Technology and Management
Gwalior, India
ipg_2014082@iiitm.ac.in

Dr. Ritu Tiwari
Department of Information Technology)
ABV-Indian Institute of Information Technology and Management
Gwalior, India
tiwariritu2@gmail.com

***Abstract*—Deep learning healthcare applications have evolved over the last few years. This advancement leads to new applications and possibilities in the field of health care. Due to different variants of deep learning algorithms like convolutional and recurrent neural network these advancements in healthcare application made possible.**

***Index Terms*—Deep learning, LSTM, Healthcare, Recurrent Neural Network, GRU.**

I. Introduction

A. Hospital Readmission

Hospital Readmissions are identified as a signal of the poor quality of care, such as insufficient discharge planning and care coordination. Hospital readmission[1] happens when a patient discharge from the hospital is readmitted within a specified interval. Readmission can occur for any planned or unplanned reason. These factors pushed Hospital organizations to think about controlling readmission rates an increase in readmission rates leads to an increase in penalties. There has been an increased usage of Readmission rates as an outcome measure in health services research and as a quality benchmark for health systems. For Medicare patients, hospitalizations are very stressful, and it is even more when they result in subsequent readmissions. A lot of research studies proved that hospitals could engage in several activities like clarifying patient discharge instructions, coordinating with post-acute care providers, vibrant cleaning mechanism to reduce the rate of readmissions of patients. But these individual follow-ups can be costly.

B. Machine Learning For Health Care

Machine learning leads to provide intelligence by providing new knowledge. The healthcare sector is producing an enormous mass of data, and it is beyond human capability to manually analyze this vast data and produce inferences based on that. The machine learning algorithms automatically find patterns in the data, which enable to get inferences from the data easily. Machine learning can assist healthcare providers in a variety of tasks like hospital readmission of patients suffering from chronic illness. This can not only lower the lifetime risk of the patient but also can save millions of dollars if the algorithm can predict efficiently and correctly. Other applications of machine learning in the healthcare domain includes drugs combination which should be avoided taking together, classifying images for different diseases like skin cancer. Keeping this in view, it can be said that machine learning can revolutionize the healthcare sector.

C. Deep Learning

Deep learning is one of the extended branches of Machine learning, which works on a specific machine learning algorithm, that is the artificial neural network. Its leveraging the recent advancements in computing power and thus using specific purpose neural network to learn from a plethora of data and make predictions based on the patterns detected. Deep neural networks are specialized at solving problems including data which is greatly structured. It also promises to replace hand-engineered features with feature extraction techniques that are hierarchical, unsupervised or semi-supervised. Multiple hidden units are being used 4 between input and output in case of Deep Neural Network (DNN). Similar to artificial neural network, the complex non-linear relationships can be modeled using DNN. .[2] Similar to artificial neural network, the complex non-linear relationships can be modeled using DNN.

D. Recurrent Neural Network

An artificial neural network has a special type called recurrent neural network (RNN), and it has directed cycles between connection. The dynamic temporal behavior is exhibited by the internal state of the network. Unlike ANN, arbitrary sequences can be processed by using the internal state of RNN. This is the fundamental architecture developed in the 1980s: neuron-like units in a network, a directed connection between every neuron. There are input, output, and hidden nodes. One input vector is supplied at every time steps. At every time step, the nonlinear function of the weighted sum of all the activations of connected units is calculated by each non-input unit.[3].

E. Long Short term Memory

The Long short-term memory (LSTM) network, a deep learning RNN, is used by numerous researchers, published by

Schmidhuber & Hochreiter in 1997[4]. This deep learning system overcomes the vanishing gradient problem of traditional RNN. The recurrent gates called forget gates are augmented in LSTM. The vanishing or exploding of backpropagated error is prevented in LSTM RNNs. LSTM can learn from events that occurred thousands of time steps ago so it can be used for very deep learning tasks. LSTM has a unique architecture that consists of a memory cell, which can maintain the state which it is currently in, over time. Over time, many improvements have been suggested and made in the standard architecture of LSTM to make it more efficient. Today, LSTMs are being used in a variety of learning problems which differ in scale and nature significantly when compared to the problems which were initially solved by LSTM.

II. Literature Review

During the last decade several research work is carried out in the field of healthcare. Hospital readmission prediction is important and crucial application of healthcare for the improved quality of life of an individual.

After stunning successes in cognitive domains, deep learning is expected to transform healthcare [21]. Most remarkable results thus far in health have been in diagnostic imaging [7], [9], which is a natural step given recordbreaking results in computer vision. However, diagnostic imaging is only a small part of the story. A full intelligent medical system should be able to reason about the past (historical illness), present (diagnosis) and future (prognosis). Here we adopt the notion of reasoning as algebraically manipulating previously acquired knowledge in order to answer a new question [1]. For that we learn to embed discrete medical objects into continuous vectors, which lend themselves to a wide host of powerful algebraic and statistical operators

Author [5] developed an approach that helps clinicians determine the optimal initial dose of a drug to safely and quickly reach a therapeutic aPTT window. In [6] author uses a super learner algorithm is used for predicting hospital mortality in patients. In [7] author used a novel called Interpretable Mimic Learning is used, to learn interpretable phenotype features for making robust prediction while mimicking the performance of deep learning models metaheuristics. In [8] Combination of convolution and recurrent neural network model is used for object detection. In [9] authors objective was to reduce overfitting using batch normalization. In [10] author introduced interpretable mimic learning that uses gradient boosting trees to learn interpretable models and at the same time achieves strong prediction performance as deep learning models. In [11] author found the proportional split of data is useful for deep learning applications when dealing with large volumes of data. In [12] author leveraged longitudinal EHR data for early detection of heart failure using Recurrent Neural Network.Researchers have recently begun attempting to apply neural network based methods (or deep learning) to EHR to utilize its ability to learn complex patterns from data. Previous studies such as phenotype learning or representation learning , however, have not fully addressed the sequential nature of EHR is especially related to our work in that both studies use RNN for sequence prediction. However, while uses regular times series of real-valued variables collected from ICU patients to predict diagnosis codes, we use discrete medical codes (e.g. diagnosis, medication, procedure) extracted from longitudinal patient visit records. Also, in each visit we make a prediction about predict diagnosis, medication order in the next visit and and the time to next visit. In [13] author uses Long short term memory (LSTM) RNN to predict disordered proteins, such as SPOT-disorder. In [14] State-of-the-art deep and recurrent neural network models were applied to achieve robust results on classifying opioid users. In [15] author used Gated Recurrent Unit (GRU) to solve the vanishing gradient problem of a standard RNN. In [16] author used LSTM to overcomes the vanishing gradient problem of traditional RNN. In [17] author Combines the use of time frequency heat map representations with a deep convolutional neural network (CNN) to classify heart sounds. In [18] author used Statistical Analysis is used to find the mortality and readmission rates. In [19] author implemented a 13-layer deep learning CNN algorithm for the automated EEG analysis.

III. Methodology

A. Data description: MIMIC-III

Medical Information Mart for Intensive Care (MIMIC-III)[20] consists of data about patients admitted to various critical care units in a large hospital. MIMIC-III is generally viewed as a large and single-center database. A large number of different parameters are present in MIMIC III database. These parameters include information such as vital signs, medications, laboratory measurements, observations, fluid balance, procedure codes, diagnostic codes, imaging reports, hospital length of stay, survival data, and others. The database consists of information of around 58,576 distinct patients who were admitted to various critical care units of the hospital between 2001 and 2012. The data comprises of patients aged 16 years or above only. Descriptive statistics were performed on this dataset, and it was found that the median age is 65.8 years for adult patients(Q1-Q3: 52:877:8). Out of total patients, there were 55.9% male patients and only 11.5% of the cases had in-hospital mortality. It was calculated from the data that on an average a patient stayed for 2.1 days in ICU and had an average of 6.9 Days of hospital stay. MIMIC-III database have several idiosyncratic properties about it, and these are mentioned below:

- The dataset is a huge database accumulated throughout more than a decade and consists of very detailed information of each patient under care.
- MIMIC-III database requires a user to oblige by a user data agreement and once itis accepted the analysis is boundless.
- It contains discharge summaries as well as reports of ECG, imaging studies and information about various codes like International Classification of Disease, 9thEdition (ICD-9) codes, Diagnosis Related Group (DRG) codes, and Current Procedural Terminology (CPT) codes.

- It is a time series data. Clinical variables are recorded concerning time for each patient. Figure 1 shows the chart of timing clinical variables recorded in the dataset.

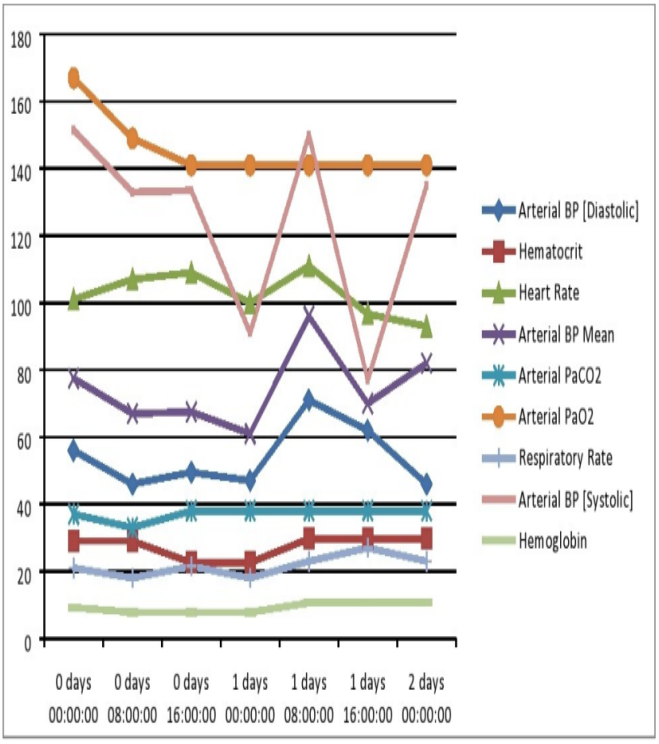

Fig. 1. Some of the Clinical recordings associated with each patient[20]

B. Data Preprocessing

1) Data Extraction: MIMIC-III dataset consisted of around 58,576 patients who were diagnosed as suffering from different types of disease and hence mortality due to those diseases. These diseases include Pulmonary disease, Circulatory disease, Trauma, a disease of the digestive system and many more. Since the dataset is very large, we only consider data of those patients who were readmitted again which gives the details of 7,534 patients. The data set is divided into two classes:-

- Patients readmitted in 30 days.
- Patients readmitted after 30 days.

2) Normalization: Normalization is used to remove the bias among the features. It brings the data on a standard scale. It standardizes the range of independent features or variables of data, called feature scaling. We use min-max normalization[21] here.

- Min-Max Normalization Let a matrix "A":-

$$A = \begin{Vmatrix} a11 & a12 \\ a21 & a22 \end{Vmatrix} \tag{1}$$

So for L2 normalization we calculate

$$a = min(a11, a21), \tag{2}$$

$$b = max(a11, a12) \tag{3}$$

$$c = min(a12, a22), \tag{4}$$

$$d = max(a21, a22) \tag{5}$$

And "A" becomes:-

$$A = \begin{Vmatrix} (a11 - a)/(b - a) & (a12 - c)/(d - c) \\ (a21 - a)/(b - a) & (a22 - c)/(d - c) \end{Vmatrix} \tag{6}$$

C. Feature Selection

Features are selected through Nature Inspired Algorithms:

1) Bat Algorithm: Bat Algorithm is an optimization algorithm inspired by the echolocation behavior of microbats[22]. Bats uses echolocation to sense distance, and they also distinguish be-tween food/prey and other background barriers. Bats fly randomly with velocity v_i at location x_i with a fixed frequency f_{min}, varying wavelength and loudness A_0 to search for the victim. They can automatically regulate the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r in the scale of [0, 1], depending on the closeness of their target. Although the loudness can diversify in many ways, we assume that the loudness varies from a high (positive) A_0 to a minimum constant value A_{min}. Rules to update position x_i and velocity v_i of bats at time step t is given by:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{7}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_b)f_i \tag{8}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{9}$$

where β $\in$[0,1] is a random vector drawn from a uniform distribution and x_b is the current global best solution which is found after comparing all the solutions among all the n bats. For the local search part once a solution is selected among the current best solutions, a new solution for each bat is generated locally using a local random walk:

$$x_{new} = x_{old} + \epsilon A^t \tag{10}$$

where ϵ $\in$[-1,1] is a random number. A^t is the average loudness of all the bats at that time step.
Variation of Loudness and Pulse Emission:

$$A_i^{t+1} = \alpha A_i^t \tag{11}$$

$$r_i^{t+1} = r_i^0[1 - exp(-\gamma t)] \tag{12}$$

where α & γ are constant.

2) Grey Wolf Optimizer: Grey Wolf Optimizer (GWO) Algorithm inspired by grey wolves[23]. The GWO algorithm mimics the leadership hierarchy and hunting mechanism of grey wolves in nature. Four types of grey wolves are employed for simulating the leadership hierarchy such as alpha, beta, delta, and omega. Also, the three main steps of hunting, searching for prey, encircling prey, and attacking victim, are implemented.
Alpha wolf is a leader male or female. They make decisions like the sleeping place, hunting, etc. Other wolves acknowledge alpha wolf by their tail down. Beta wolf help alpha

wolf in making decisions. They are an advisor for alpha and discipliner for the pack. They also ensure all subordinate obey the order of alpha and give feedback to alpha. Delta wolves also called subordinate. They dominate omega wolves.
Categories of Delta wolves:-

- Scouts - Watch boundaries.
- Sentinels - Protect pack.
- Elders - Aplha or beta wolf sometimes.
- Hunters - Help aplha and beta wolf in hunting.
- Care Taker - Care ill weak and wounded wolves.

Omega wolves are like the scapegoat in the pack. They are last allowed wolves to eat having weak fitness.
Search Process in GWO Algorithm:-

- Searching
- Encircling
- Attacking the prey

a) Searching:

- Search according to aplha, beta and delta wolves. They diverge to search for prey and converge to attack prey.
- Modeled by utilizing A random variable greater than 1 or less than -1.
- When $|A| > 1$ wolves are forced to diverge from prey to find better solution.

b) Encircling Prey: It is the process in which wolves encircle the prey for attack. It is modeled as:

$$\overrightarrow{\mathrm{D}} = |\overrightarrow{\mathrm{C}}\,\overrightarrow{\mathrm{X}}_p(t) - \overrightarrow{\mathrm{X}}(t)| \tag{13}$$

$$\overrightarrow{\mathrm{X}}(t+1) = \overrightarrow{\mathrm{X}}_p(t) - \overrightarrow{\mathrm{A}}\,\overrightarrow{\mathrm{D}} \tag{14}$$

Where t is the current iteration, $\overrightarrow{\mathrm{A}}$, $\overrightarrow{\mathrm{C}}$ are coefficient vectors and $\overrightarrow{\mathrm{X}}_p$ is position of prey and $\overrightarrow{\mathrm{X}}$ is the position of grey wolf.

Updation of Coefficients

$$\overrightarrow{\mathrm{A}} = 2\overrightarrow{\mathrm{a}}\,\overrightarrow{\mathrm{r1}} - \overrightarrow{\mathrm{a}} \tag{15}$$

$$\overrightarrow{\mathrm{C}} = 2\overrightarrow{\mathrm{r2}} \tag{16}$$

Where $\overrightarrow{\mathrm{a}}$ linearly decrease from 2 to 0 over the course of iteration. $\overrightarrow{\mathrm{r1}}$ and $\overrightarrow{\mathrm{r2}}$ are random vectors $\in$ [0,1].

c) Attacking: The hunt is usually guided by alpha. The beta and delta might also participate in hunting. We first save three best solutions i.e. alpha beta and delta which are the best solutions and update positions of other agents based on these three. It is modeled as:

$$\overrightarrow{\mathrm{D}}_\alpha = |\overrightarrow{\mathrm{C}}_1\overrightarrow{\mathrm{X}}_\alpha - \overrightarrow{\mathrm{X}}| \tag{17}$$

$$\overrightarrow{\mathrm{D}}_\beta = |\overrightarrow{\mathrm{C}}_2\overrightarrow{\mathrm{X}}_\beta - \overrightarrow{\mathrm{X}}| \tag{18}$$

$$\overrightarrow{\mathrm{D}}_\delta = |\overrightarrow{\mathrm{C}}_3\overrightarrow{\mathrm{X}}_\delta - \overrightarrow{\mathrm{X}}| \tag{19}$$

$$\overrightarrow{\mathrm{X}}_1 = \overrightarrow{\mathrm{X}}_\alpha - \overrightarrow{\mathrm{A}}_1\overrightarrow{\mathrm{D}}_\alpha \tag{20}$$

$$\overrightarrow{\mathrm{X}}_2 = \overrightarrow{\mathrm{X}}_\beta - \overrightarrow{\mathrm{A}}_2\overrightarrow{\mathrm{D}}_\beta \tag{21}$$

$$\overrightarrow{\mathrm{X}}_3 = \overrightarrow{\mathrm{X}}_\delta - \overrightarrow{\mathrm{A}}_3\overrightarrow{\mathrm{D}}_\delta \tag{22}$$

$$\overrightarrow{\mathrm{X}}(t+1) = (\overrightarrow{\mathrm{X}}_1 + \overrightarrow{\mathrm{X}}_2 + \overrightarrow{\mathrm{X}}_3)/3 \tag{23}$$

Grey wolf attack prey when it stops moving. In GWO vector A is a random value within an interval [-2a,2a] a decrease from 2 to 0 throughout the iteration. When $|A| < 1$ wolves attack prey.

D. Feature Extraction

For feature extraction Convolution Neural Network(CNN) is used.

1) Convolutional Neural Network: Convolutional neural network[24] is an effective machine learning technique from the deep learning and it is similar to ordinary Neural Networks. Convolutional neural network is a network with convolutional layers. Convolutional neural network is consists of three steps of neural layers to build its architectures: Convolutional, Pooling, and Fully-Connected.

Convolutional layers: The convolutional layer is the main part of the convolutional neural network. In the convolutional layer the output result is derived from the input by filtering in certain conditions. Convolutional layers consist of a rectangular grid or cubic block of neurons. It means input, output layers with filters can be a rectangular grid or cubic block of neurons.

The filter is supplied from uppermost left to downside right. In every location of the filter, the weighted volume of the pixels is determined by the equation

$$WT^X + b \tag{24}$$

and a new neuron is acquired. In the convolutional layer three kind of hyper parameters determine the volume of the output neurons: depth, stride, and zero-padding. A number of the filters with certain strides determines the depth For original RGB image depth equal to three.

Here, whereas the stride equal to 1, the filters runs a single pixel. In this case output neuron is 5x5xd. Whereas the stride equal to 2, the filters are skipped two pixels and output neuron size is 3x3xd. In the last case, when stride is 3 the filters are skipped 3 pixels. Output neuron size is 2x2xd.

Zero padding is the filling process in the input neuron where it zeros are around the edge. Zero-padding is mostly consumed for adjustment the size of the input neuron for research purposes. It is normally applied for when the input neuron size needs to maintain in the output neuron.The output neuron size is calculated by equation

$$Output = (W - K + 2P)/S + 1 \tag{25}$$

where W is the input neurons size, K is the filter (kernel) size, P is size of the padding, and S is the stride. In the convolutional neural network linear algebraic operations are also used. Suppose that matrix dimensions are m and n (m rows, n columns). 2D convolutional cube calculates the two-dimensional convolution with two input matrices. (MA, NA) is dimensions of the matrix A, and (MB, NB) is dimensions

for the matrix B. In case, the cube determines the complete output size, convolution equation is as below

$$C(i,j) = \sum_{m=0}^{M_A-1} \sum_{n=0}^{N_A-1} A(m,n) * B(i-m, j-n) \quad (26)$$

Max-pooling layer: After each convolutional layer, a pooling layer executes next operation. The pooling layers are utilized to decrease size of the input neuron. This layer acquires small rectangular blocks from the convolutional layer and samples it to provide a single output from that block. The max pooling method is common used. A formulation for a single type of the pooling layer, max-pooling is presented in equation

$$h_l^j(x,y) = max \in \mathcal{N}(x), y \in \mathcal{N}(y) h_j^{l-1}(\bar{x}, \bar{y}) \quad (27)$$

Fully Connected layers: Fully connected layer is the last layer, which is constructed from connection of all previous neurons. The fully connected layer normally promotes reduction of spatial information because it fully connected: from all input neurons until all output neurons.

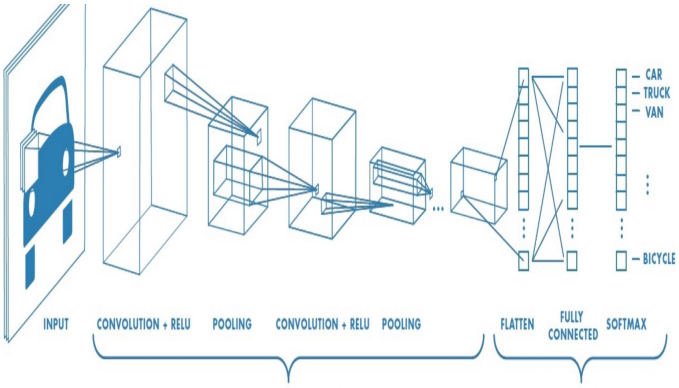

Fig. 2. Convolution Neural Network Architecture

E. Algorithms

1) Recurrent Neural Network: Recurrent neural networks (RNN) are used to capture the temporal dependency in the time series data[25]. As most of the healthcare data is a series of temporal recordings, hence RNNs have been widely adopted in the healthcare domain. We are usually provided with a series of observations $x_1 ... x_T$ and we train a classifier to generate hypotheses ŷ. The recurrent connections are added in feed-forward neural networks to make it RNN. The output of a neuron in a typical NN is as follows:

$$y_i^t = \sigma(W_i x^t + b_i) \quad (28)$$

Where W_i is the weight matrix, bi is the bias and represents the sigmoid function. While in the case of RNN, a neuron is fed with the output of the neuron at time t-1. The following equation shows the new activation function:

$$y_i^t = (W_i x^t + V_i x^{t-1} + b_i) \quad (29)$$

As RNN uses the previous outputs as recurrent connection, their current output depends upon the previous states. This property of RNN makes it very useful in sequence labeling tasks. The backpropagation through time can be used to train RNNs. It was demonstrated by that learning long-term dependencies is difficult using gradient descent. This is mainly because the backpropagating error can vanish which makes the network inefficient in learning long-range dependencies, or frequently explode which makes convergence impossible. LSTM networks were proposed to tackle the problem of vanishing gradients and were developed to model long-range dependencies efficiently. LSTMs can accomplish this by keeping an internal state that represents the memory cell of the LSTM neuron. This internal state can only be written and read through gates which control the information flowing through the cell state. The following diagram shows the recurrent neural network.

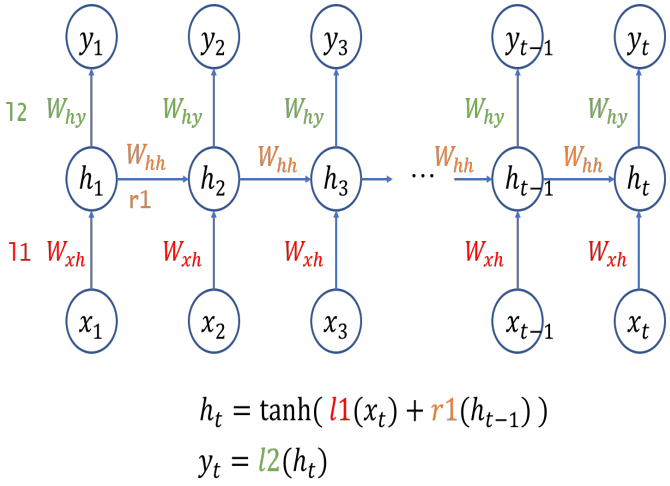

Fig. 3. An recurrent neural network

2) Long Short Term Memory (LSTM) Networks: Long Short Term Memory networks usually just called LSTMs are a special kind of RNN, capable of learning long-term dependencies. They were introduced by [26] and were refined and popularized by many people. They work remarkably well on a large variety of problems and are now widely used. To solve long-term dependency problem, LSTMs are explicitly designed. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn! All recurrent neural networks have the form of a chain of repeating modules of the neural network. In traditional RNNs, this repeating module will have a pretty simple structure, such as a single tank layer. LSTMs also have this chain-like structure, but the recurrent module has a different structure. Rather than having a single neural network layer, there are four layers which are interacting extraordinarily. The intermediate information is stored in a single hidden layer h and its state changes over time (h_{t-1}, h_t, h_{t+1}). On the final hidden state vector h_T, we used a fully connected layer followed by sigmoid function. For loss function, we used log

loss. The following equations can be used for calculation of the current hidden layer h_t.

$$f_t = \sigma(W_f X_t + R_f h_{t-1} + b_f) \tag{30}$$

$$i_t = \sigma(W_i X_t + R_i h_{t-1} + b_i) \tag{31}$$

$$o_t = \sigma(W_o X_t + R_o h_{t-1} + b_o) \tag{32}$$

$$\tilde{C}_t = \Phi(W_C X_t + R_C h_{t-1} + b_c) \tag{33}$$

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \tag{34}$$

$$h_t = o_t \phi(C_t) \tag{35}$$

f_t , i_t and o_t are forget gate, input gate and output gate respectively. To decide which historical information will be discarded from the cell state forget gates are used, the update of cell state is decided by the input gate, and the output gate decides the output of the cell state. Cell states are completely overridden in classical RNN, but LSTM has the potential to add or remove information to the cell state. The input weight of each gate, recurrent weight, and the bias are expressed as W* , R* , b* respectively where * can be f, i, o and c. Here σ, Φ stands for an element-wise application of the sigmoid (logistic) and tanh function respectively. For matrix multiplication. The candidate values are computed in equation (6), and equation(7) old state is multiplied by f_t, and this helps in forgetting the things we decided to forget. Then i_t*δC_t is added in it. This is the new candidate values, scaled by how much we decided to update each state value. The final output of an LSTM unit is given by equation 8. Here * represents the Hadamard (element-wise) multiplication operation. Figure 3 shows the LSTM network.

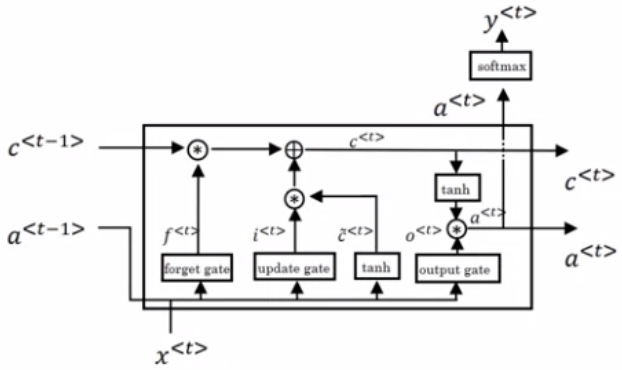

Fig. 4. A LSTM Network

3) Bidirectional Recurrent Neural Networks (BRNN): Bidirectional Recurrent Neural Networks also called BRNN is just like RNN but it trains simultaneously on both sides of the time series data[27]. This model gives the better result in both regression and classification problem. BRNN computes both forward ($\overrightarrow{\mathrm{h}}$) and backward ($\overleftarrow{\mathrm{h}}$) hidden sequence.

$$\overrightarrow{\mathrm{h}}_t = \mathcal{H}(W_{x\overrightarrow{\mathrm{h}}} x_t + W_{\overrightarrow{\mathrm{h}}\overrightarrow{\mathrm{h}}} \overrightarrow{\mathrm{h}}_{t-1} + b_{\overrightarrow{\mathrm{h}}}) \tag{36}$$

$$\overleftarrow{\mathrm{h}}_t = \mathcal{H}(W_{x\overleftarrow{\mathrm{h}}} x_t + W_{\overleftarrow{\mathrm{h}}\overleftarrow{\mathrm{h}}} \overleftarrow{\mathrm{h}}_{t+1} + b_{\overleftarrow{\mathrm{h}}}) \tag{37}$$

$$y_t = W_{\overrightarrow{\mathrm{h}}y} \overrightarrow{\mathrm{h}}_t + W_{\overleftarrow{\mathrm{h}}y} \overleftarrow{\mathrm{h}}_t + b_o \tag{38}$$

The long range context can be accessed in both directions by combining BRNNs with LSTM which gives bidirectional LSTM[28].

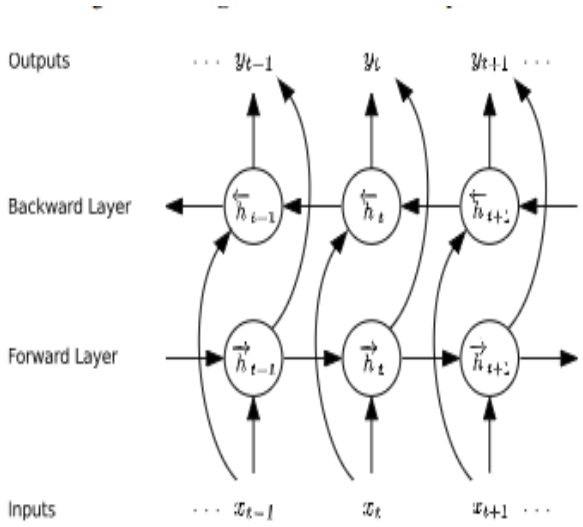

Fig. 5. Bidirectional Recurrent Neural Networks

F. Flow Diagram

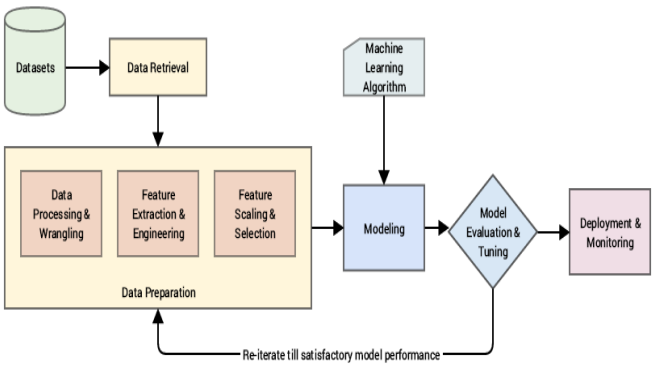

Fig. 6. Flow Diagram

IV. Results

A. Implementation Details

The entire dataset was split using stratified sampling and 10% of the dataset was used as a validation set and random search was used for hyperparameter optimization. The rest 80% of the dataset was used for training purpose and was tested on 10% of the dataset. The LSTM model was trained with 30 epochs using Adam optimizer. LSTM layer uses 100 memory cell with no dropout and 25 memory cell with 20% dropout and these architectures are found after validation performance. Also train this model with Bat Algorithm.
Another model train on this dataset was convolution long short term memory(CLSTM). It is a hybrid model in which convolution layers are used for feature extraction. The CLSTM model was trained with 30 epochs and the batch size of 1000. First, the convolution layer was used with 45 filters of size 5×5. Then the LSTM layer that uses 100 memory cell with 20% dropout. Also train this model with Grey Wolf Optimizer Algorithm.
In the training phase, the model was saved and also compute these performance metrics (AUC, F1, precision, and recall).
Several built-in modules of python will be utilized, namely:

- Numpy
- Pandas
- Scikit-Learn
- Keras
- Matplotlib
- NiaPy

The computations were performed on Ubuntu 16.04 LTS with following hardware specifications:

- **Processor** Intel i7 6th Gen. broadwalle processor.
- **RAM** 24 GB DDR4
- **Secondary Memory** 250 GB SSD
- **Graphics Card** Nvidia 1080Ti 11GB

B. Results Obtained

This section presents the readmission prediction results obtained from different predictive models. The prediction accuracy will be compared in terms of two factors AUC under ROC (Receiver Operating Characteristic) Curve[29], and overall accuracy. Both area under ROC curve and overall accuracy depends on the classifiers ability to rank patterns for positive class, but in the case of overall accuracy, it also depends on the ability to calculate the threshold in the ranking to classify the positive class. Applying different algorithms on the dataset, with default parameters, we obtained results as mentioned in Table 1. Also we compare deep learning results with baseline models in Table 2 to show how deep learning helps in improving the accuracy.

1) Curves showing Receiver Operating Curve (ROC) and Loss Curve of different models:

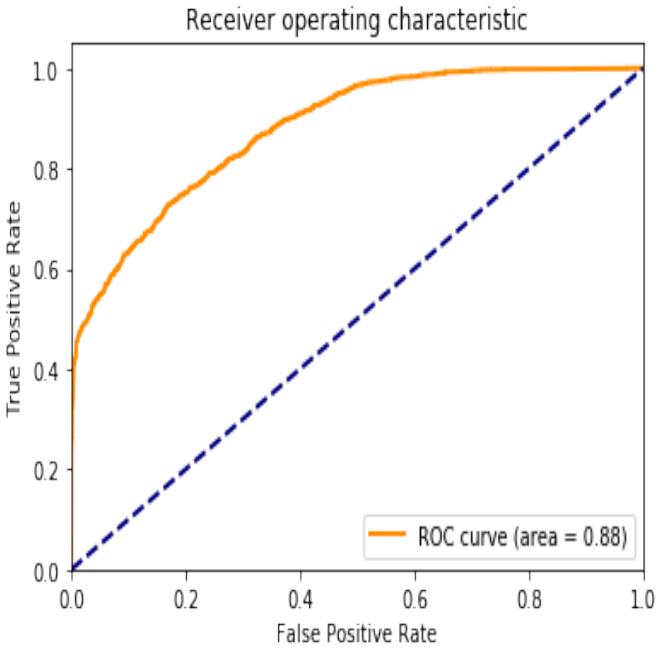

Fig. 7. Receiver Operating Characteristic Curve: Long Short Term Memory.

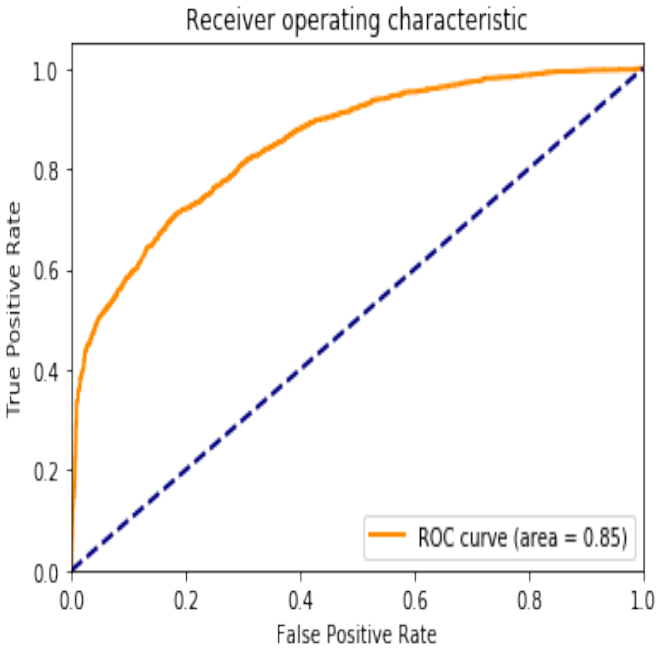

Fig. 8. Receiver Operating Characteristic Curve Bi-Directional Long Short Term Memory.

Algorithm type	**Algorithm**	**AUC**	**Accuracy**	**F1 Score Micro**	**F1 Score Macro**
Deep Learning without Nature Inspired Algorithms	**Long Short Term Memory**	0.88	78.84%	0.80	0.76
	Long Short Term Memory Auto-Encoders	0.73	70.00%	0.70	0.62
	Convolution Long Short Term Memory	0.75	70.98%	0.71	0.66
	Convolution Long Short Term Memory Auto-Encoders	0.87	80.32%	0.80	0.78
	Bi-Directional Long Short Term Memory Auto-Encoders	0.85	77.99%	0.78	0.75
Deep Learning with Nature Inspired Algorithms	**Long Short Term Memory with Bat Algorithm**	0.75	70.42%	0.70	0.63
	Long Short Term Memory with Grey Wolf optimizer Algorithm	0.85	81.57%	0.79	0.76

Table 1: Comparison among different Models implemented so far

Methods	Area Under ROC Curve	Accuracy
Logistic Regression	0.54	58.09%
Support Vector Machines	0.54	59.37%
Decision Tree	0.52	55.68%
Random Forest	0.53	60.49%

Table 2: Comparison with Baseline Models

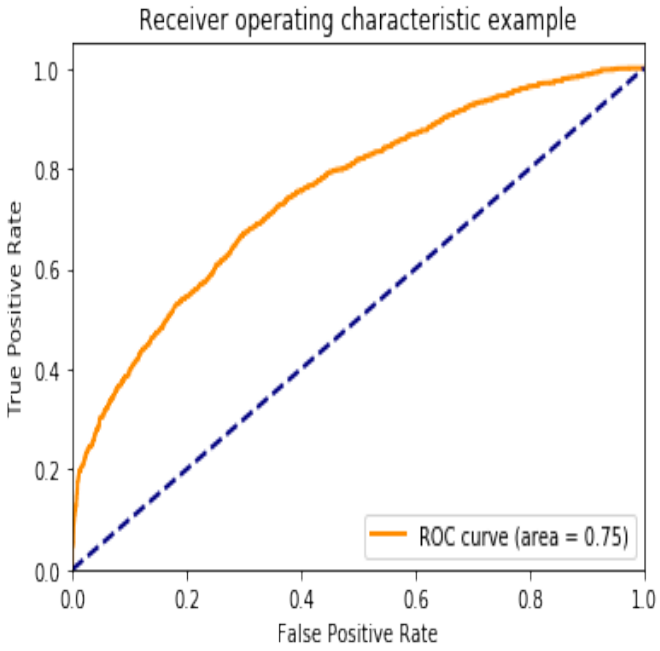

Fig. 9. Receiver Operating Characteristic Curve: Convolution Long Short Term Memory.

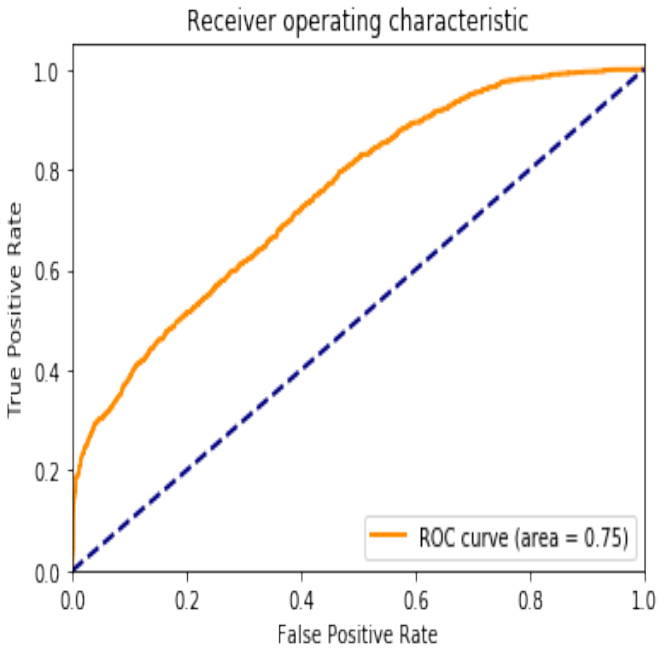

Fig. 10. Receiver Operating Characteristic Curve: Long Short Term Memory with Bat Algorithm.

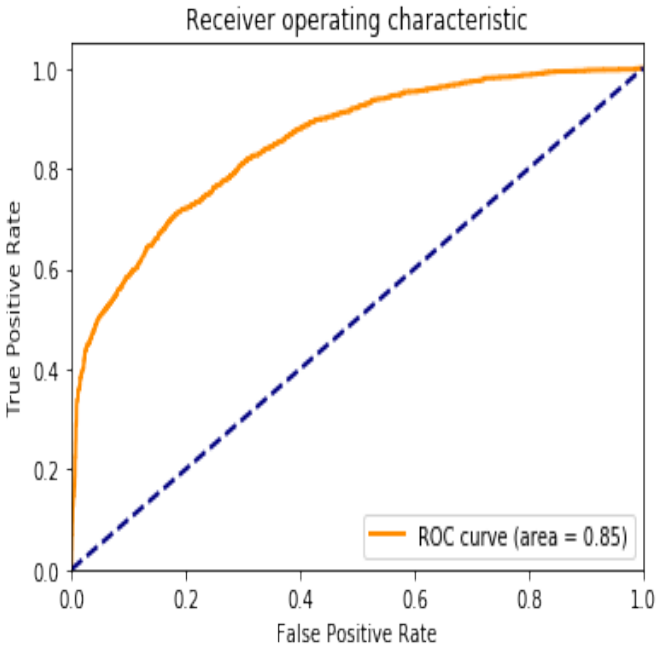

Fig. 11. Receiver Operating Characteristic Curve: Long Short Term Memory with Grey Wolf Optimizer Algorithm.

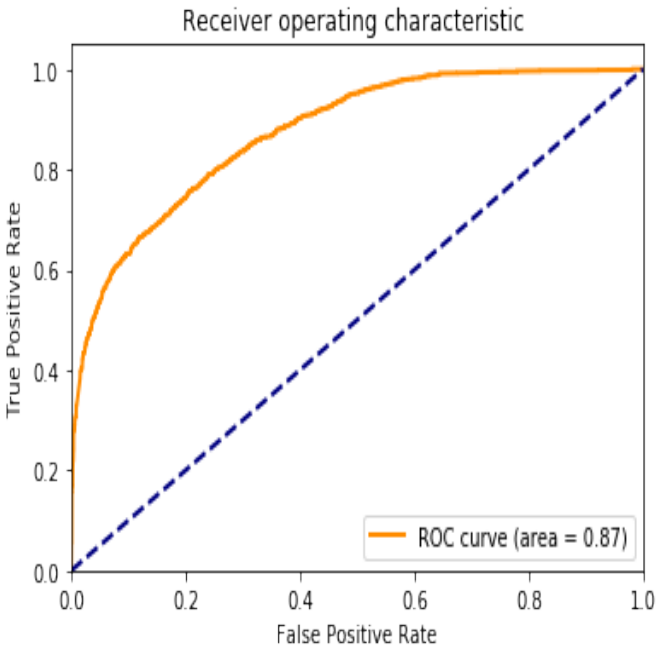

Fig. 12. Receiver Operating Characteristic Curve Convolution Long Short Term Memory Auto Encoders.

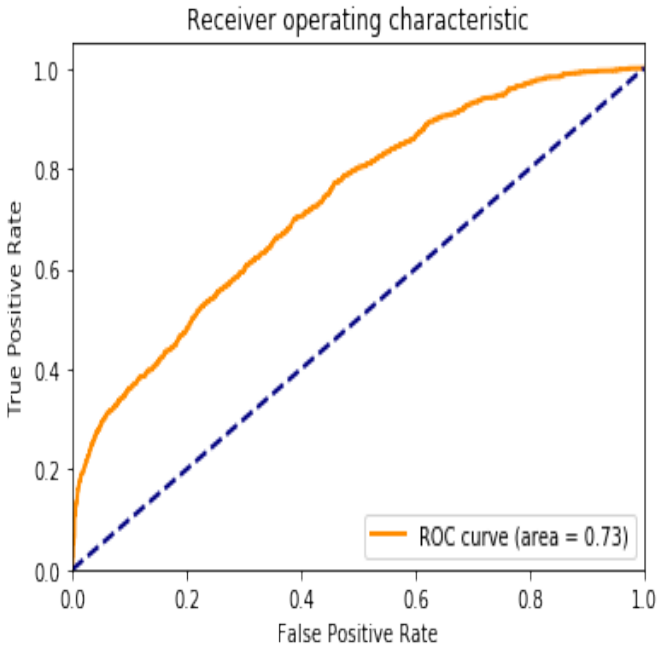

Fig. 13. Receiver Operating Characteristic Curve: Long Short Term Memory Auto-Encoders.

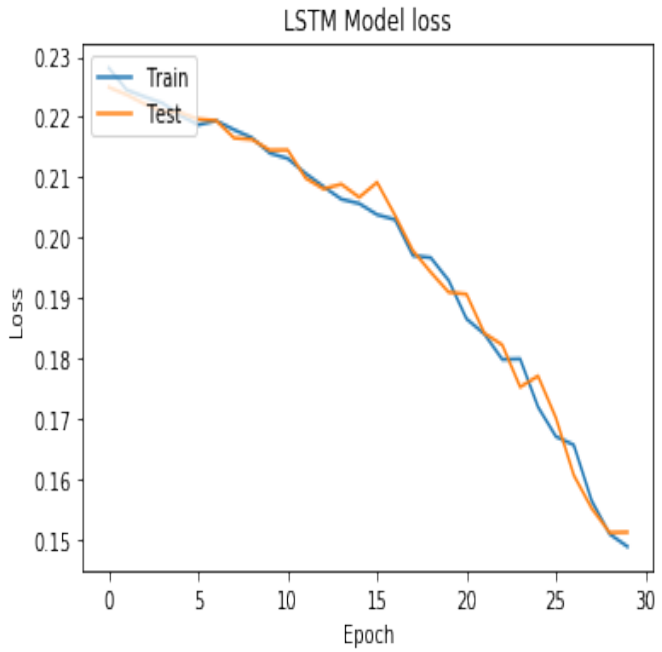

Fig. 14. Loss Curve: Long Short Term Memory.

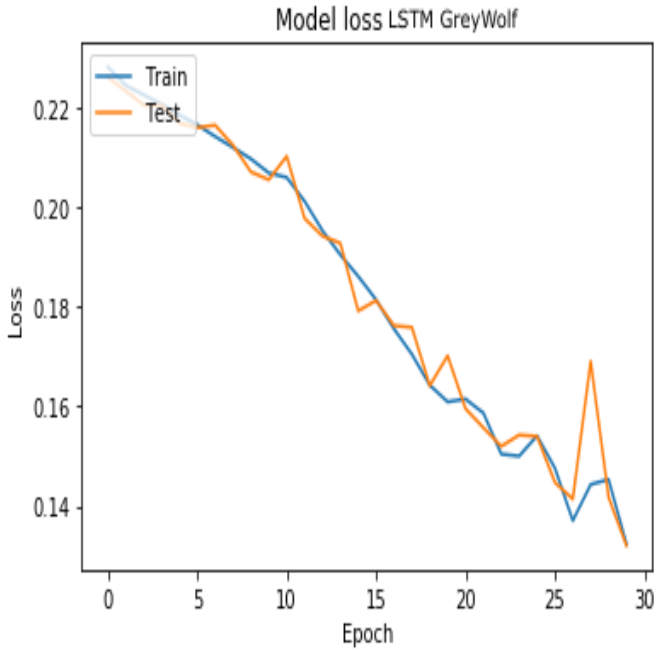

Fig. 15. Loss Curve: Long Short Term Memory Grey Wolf.

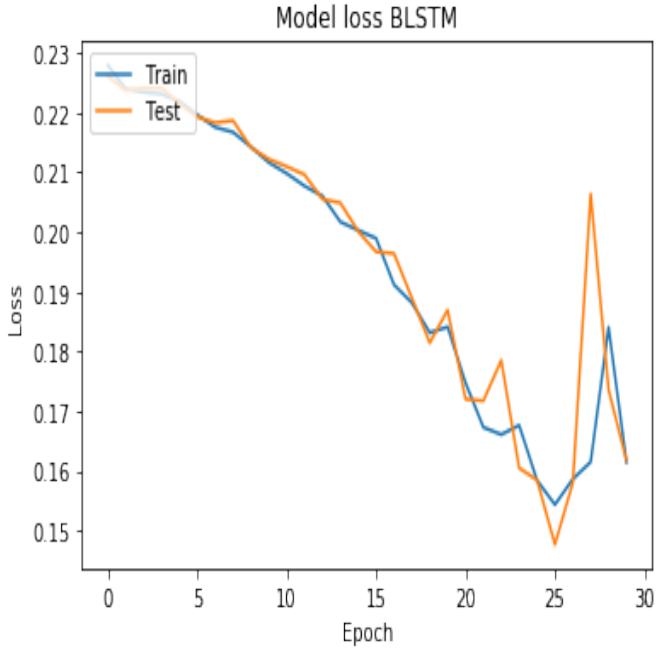

Fig. 16. Loss Curve: Bidirectional Long Short Term Memory.

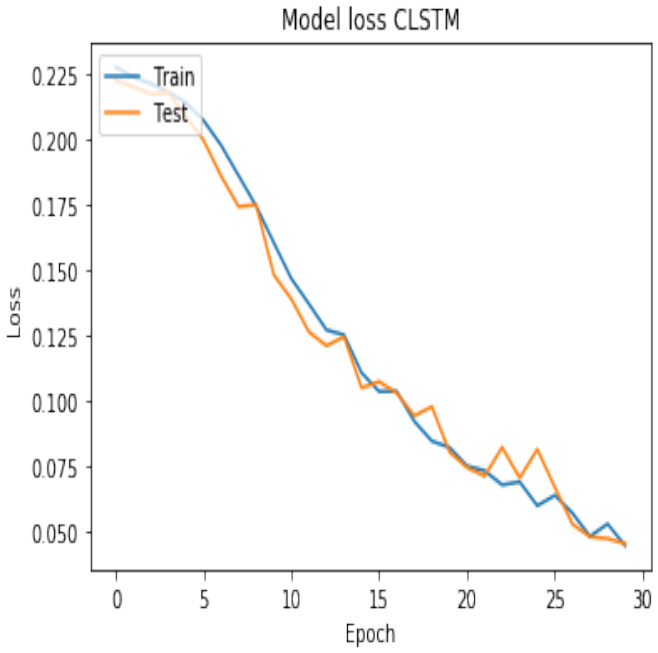

Fig. 17. Loss Curve: Convolution Long Short Term Memory.

V. Conclusion

Hospital Readmissions is very stressful thing for both patients and hospitals. In our research, we built a powerful model to predict the number of patients readmitted to a hospital. We compared different deep learning models with and without Nature inspired algorithms to predict and concluded that Long short term memory with grey wolf optimizer outperformed the rest of the machine learning models in the prediction quality. We also found that the performance of LSTM and Convolution LSTM was remarkable on this data. This framework can be implemented in todays health system to target high risks patients, reduce rate of readmission and deliver better health care.

References

[1] Strack, B., DeShazo, J. P., Gennings, C., Olmo, J. L., Ventura, S., Cios, K. J. and Clore, J. N.: 2014, Impact of hba1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records, BioMed research international 2014.

[2] Bengio, Y. et al.: 2009, Learning deep architectures for ai, Foundations and trends R in Machine Learning 2(1), 1127.

[3] Lipton, Z. C.: 2015, A critical review of recurrent neural networks for sequence learning, CoRR abs/1506.00019.

[4] Hochreiter, S. and Schmidhuber, J.: 1997, Long short-term memory, Neural computation 9(8), 17351780.

[5] Ghassemi, M. M., Richter, S. E., Eche, I. M., Chen, T. W., Danziger, J. and Celi, L. A.: 2014, A data-driven approach to optimized medication dosing: a focus on heparin, Intensive care medicine 40(9), 13321339.

[6] Pirracchio, R., Petersen, M. L., Carone, M., Rigon, M. R., Chevret, S. and van der Laan, M. J.: 2015, Mortality prediction in intensive care units with the super icu learner algorithm (sicula): a population-based study, The Lancet Respiratory Medicine 3(1), 4252.

[7] Che, Z., Purushotham, S., Khemani, R. G. and Liu, Y.: 2015, Distilling knowledge from deep networks with applications to healthcare domain, CoRR abs/1512.03542.

[8] Liang, M. and Hu, X.: 2015, Recurrent convolutional neural network for object recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 33673375.

[9] Ioffe, S. and Szegedy, C.: 2015, Batch normalization: Accelerating deep network training by reducing internal covariate shift, Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, ICML15, JMLR.org, pp. 448456. URL: http://dl.acm.org/citation.cfm?id=3045118.3045167

[10] Che, Z., Purushotham, S., Khemani, R. and Liu, Y.: 2016, Interpretable deep models for icu outcome prediction, AMIA Annual Symposium Proceedings, Vol. 2016, American Medical Informatics Association, p. 371.

[11] Choi, E., Bahadori, M. T., Schuetz, A., Stewart, W. F. and Sun, J.: 2016, Doctor ai: Predicting clinical events via recurrent neural networks, Machine Learning for Healthcare Conference, pp. 301318.

[12] Choi, E., Schuetz, A., Stewart, W. F. and Sun, J.: 2016, Using recurrent neural network models for early detection of heart failure onset, Journal of the American Medical Informatics Association 24(2), 361370.

[13] Hanson, J., Yang, Y., Paliwal, K. and Zhou, Y.: 2016, Improving protein disorder prediction by deep bidirectional long short-term memory recurrent neural networks, Bioinformatics 33(5), 685692.

[14] Che, Z., Sauver, J. S., Liu, H. and Liu, Y.: 2017, Deep learning solutions for classifying patients on opioid use, AMIA Annual Symposium Proceedings, Vol. 2017, American Medical Informatics Association, p. 525.

[15] Dey, R. and Salemt, F. M.: 2017, Gate-variants of gated recurrent unit (gru) neural networks, Circuits and Systems (MWSCAS), 2017 IEEE 60th International Midwest Symposium on, IEEE, pp. 15971600.

[16] Greff, K., Srivastava, R. K., Koutnk, J., Steunebrink, B. R. and Schmidhuber, J.: 2017, Lstm: A search space odyssey, IEEE transactions on neural networks and learning systems 28(10), 22222232.

[17] Rubin, J., Abreu, R., Ganguli, A., Nelaturi, S., Matei, I. and Sricharan, K.: 2017, Recognizing abnormal heart sounds using deep learning, KHD@IJCAI.

[18] Franco, J., Formiga, F., Trullas, J.-C., Bautista, P. S., Conde, A., Manzano, L., Quiros, R., Franco, A. G., Ezquerro, A. M., Montero-Perez-Barquero, M. et al.: 2017, Impact of prealbumin on mortality and hospital readmission in patients with acute heart failure, European journal of internal medicine 43, 3641.

[19] Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H. and Adeli, H.: 2018, Deep convolutional neural network for the automated detection and diagnosis of seizure using eeg signals, Computers in biology and medicine 100, 270278

[20] Johnson, A. E., Pollard, T. J., Shen, L., Li-wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A. and Mark, R. G.: 2016, Mimic-iii, a freely accessible critical care database, Scientific data 3, 160035.

[21] Jain, Y. Kumar, and Santosh Kumar Bhandare. "Min max normalization based data perturbation method for privacy protection." International Journal of Computer & Communication Technology 2.8 (2011): 45-50.

[22] Yang, X.-S. and Hossein Gandomi, A.: 2012, Bat algorithm: a novel approach for global engineering optimization, Engineering Computations 29(5), 464483.

[23] Lipton, Z. C.: 2015, A critical review of recurrent neural networks for sequence learning, CoRR abs/1506.00019.

[24] Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences." arXiv preprint arXiv:1404.2188 (2014).

[25] Che, Z., Purushotham, S., Cho, K., Sontag, D. and Liu, Y.: 2018, Recurrent neural networks for multivariate time series with missing values, Scientific reports 8(1), 6085.

[26] Hochreiter, S. and Schmidhuber, J.: 1997, Long short-term memory, Neural computation 9(8), 17351780.

[27] Schuster, M. and Paliwal, K. K.: 1997, Bidirectional recurrent neural networks, IEEE Transactions on Signal Processing 45(11), 26732681.

[28] Graves, A. and Schmidhuber, J.: 2005, Framewise phoneme classification with bidirectional lstm and other neural network architectures, Neural Networks 18(5- 6), 602610.

[29] Rubin, J., Abreu, R., Ganguli, A., Nelaturi, S., Matei, I. and Sricharan, K.: 2017, Recognizing abnormal heart sounds using deep learning, KHD@IJCAI.