TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

THESIS NO.: 072MSCS651

An Assessment for Predicting Stroke Patients using Bidirectional LSTM

by

Ajaya Puri

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE AND KNOWLEDGE ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

NOVEMBER, 2019

Copyright

Recommendation

The undersigned certify that they have read and recommended to the Department of Electronics and Computer Engineering for acceptance, a thesis entitled “An Assessment for Predicting Stroke Patients Using Bidirectional LSTM”, submitted by AjayaPuri in partial fulfillment of the requirement for the award of the degree of “Master of Science in Computer System and Knowledge Engineering.

.....

Supervisor

Prof. Dr. Shashidhar Ram Joshi (PhD)

.....

External Examiner

Mr. Bikash Bahadur Shrestha

.....

Committee Chairperson

Dr. Aman Shakya

Program Coordinator

Department of Electronics and Computer Engineering

Departmental Acceptance

The thesis entitled "**An Assessment For Predicting Stroke Patients Using Bidirectional LSTM**", submitted by Mr. Ajaya Puri in partial fulfillment of the requirement for the award of the degree of "**Master of Science in Computer System and Knowledge Engineering**" has been accepted as a bonafide record of work independently carried out by him in the department.

. . . . . .

Dr. Surendra Shrestha (PhD)
Head of the Department
Department of Electronics and Computer Engineering,
Pulchowk Campus,
Institute of Engineering,
Tribhuvan University,
Nepal.

Acknowledgement

I would like to express my sincere gratitude to the Electronics and Computer Engineering, Pulchowk Campus for accepting my thesis on "**An Assessment for Predicting Stroke Patients Using Bidirectional LSTM**". I would like to extend my sincere thanks for providing me with all the essential co-operation, valuable suggestions for choosing this thesis work.

I am grateful to my supervisor **Prof. Dr. Shashidhar Ram Joshi** for providing useful information and guidance regarding this thesis. He consistently allowed this research to be my own work, steered me in the right direction whenever he thought I needed it.

I would like to thank **Prof. Dr. Subarna Shakya** for his encouragement to this work.

I am grateful to **Dr. Divakar Raj Pant, Dr. Sanjeeb Panday, Dr. Basanta Joshi** and **Dr. Surendra Shrestha** for continuous positive feedback supports for the work.

I am also grateful to our program coordinator **Dr. Aman Shakya** for providing coordination and support related to this thesis.

I would also like to express my heartfelt thanks to respected teachers, my families and friends who have helped and supported me directly and indirectly during the thesis.

Abstract

Stroke is the medical condition when the supply of blood to the brain is either interrupted or reduced for very certain duration of time. When this happens, the brain does not get enough oxygen or nutrients, and brain cells start to die.

This thesis presents the development and evaluation of a machine learning model using deep learning techniques. The improved, memory based Bidirectional recurrent neural called Bidirectional Long short-term memory (BLSTM RNN) is used for the research work. The model thus developed predict whether a patient will experience stroke or not based on a time series input data computation. A 3-layer architecture having single BLSTM unit, Adam as model optimizer and dropout regularization of 0.42 achieves accuracy of 91%.

The model is developed by processing patient time series information which includes demographic and medical historical data. It includes age, gender, hypertension, heart diseases, and altogether ten biometric information. This work contributes for decision support for individuals and medical persons on their future stroke possibility.

Keywords:

Stroke Detection, Deep Learning, Recurrent Neural Network, BLSTM, LSTM, GRU

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	=	Artificial Neural Network
AUC	=	Area Under the Curve
BLSTM	=	Bidirectional Short-Term Memory
BP	=	Back propagation
DNN	=	Deep Neural Network
ELU	=	Exponential Rectified Linear Unit
FP	=	False Positive
FN	=	False Negative
GPU	=	Graphics Processing Unit
GRU	=	Gated Recurrent Units
LSTM	=	Long Short-Term Memory
ML	=	Machine Learning
ReLU	=	Rectified Linear Unit
ROC	=	Receiver Operating Characteristics
RNN	=	Recurrent Neural Network
SGD	=	Stochastic Gradient Descent
TP	=	True Positive
TN	=	True Negative

Chapter 1: Introduction

1.1 Background

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the human brain, process information. It is composed of a large number of highly interconnected processing elements (neurons) working in union to solve specific problems. Previously it was believed that machines are only for arithmetic operations but not for complex tasks that requires some intelligence like fingerprint recognition, image captioning, speech recognition, facial classification, object detection, image classification etc. This possibility is made through learning process or learning algorithm and availability of data. A huge dataset is required for training the model. Training is followed by testing to verify whether the system is working as desired or not.

Neural networks are sometimes described in terms of their depth, including how many layers they have between input and output, or the model's so-called hidden layers. It was believed that only 2 to 3 hidden layers are sufficient for Neural Network to work properly but later on it is observed that even more layers can represent high dimensional features of the input signals. Such neural networks are referred to as Deep Neural Networks (DNN). The common approach to ANNs is to use a single network and train it with labelled training data. This is called supervised learning as the model is explicitly told what output is correct for a given input. A drawback to supervised learning is that it requires data with labeled output, often created by human labor. Neural networks require thousands or even millions of elements for training, and supervised learning requires human labelling of each element. These labels may not always be available or even feasible. Unsupervised learning, on the other hand, is training without labels, which forces the learner to make its own generalizations about the data. Training a network to generate more natural looking images is certainly possible but would be a hard problem to solve with conventional supervised learning, requiring either a human to evaluate every generated target or creating a large dataset with examples of good and bad categories.

Large number of casualties have been reported with the stroke attack. It has been stated that in every 4 minutes one casualties occurs due to stroke attack and up to 80% of strokes can be prevented if we can apply a predictive model in its early age [1].

Healthcare domain is a recent research area for predictive and prescriptive analytics. Medical data comes from hospital in the form of patient records [2]. Such record contains data that categories the medical diagnosis and procedure done earlier. This is the baseline information for my research work. Such diagnosis and procedure done during the medical treatment bear the underlying knowledge of stroke risk.

Recently, a branch of Machine Learning (ML) techniques based on deep learning approaches, such as deep neural networks (DNN), has achieved impressive and sometimes, breakthrough, results across a variety of artificial intelligence tasks. The approach of deep learning is inspired by the ability of human brain to abstract high-level representations from low-level sensory stimuli; these multi-leveled representations can be casted mathematically as multi-layered neural networks, and only recently, it is being able to be trained via layer-wise back-propagation to obtain tractable optimization [3].

1.2 Statement of Problem

In traditional Machine learning techniques, most of the applied features need to be identified by a domain expert in order to reduce the complexity of the data and make patterns more visible to learning algorithms to work. The biggest advantage Deep Learning algorithms are that they try to learn high-level features from data in an incremental manner. This eliminates the need for domain expertise and hard-core feature extraction.

As, vanilla RNN suffer from vanishing and exploding problem, there has been done so-sophisticated enhancement on the algorithm. Namely, LSTM and GRU has shown their performance in different applied domains. Such researches are well published and described in the predictive model field like pattern recognition, sequence learning, stoke analysis, trend analysis etc. But there are very few works on medical predictive domains. After this research work, a conclusive and novel model will be proposed for stroke patient prediction using BLSTM network.

1.3 Research Objective

The objectives of this thesis are as follows.

1. To develop a stable model for stroke prediction problems.
2. To analyze the parameters like accuracy, training time, activation function, optimization function, objective function, training accelerator on different RNNs architectures.

Chapter 2: Literature Review

2.1 Related Work

Data driven approach is widely accepted model in predictive analytics. The success of machine learning algorithm in other active research domain pull scholar attraction day by day. A well and accurate machine learning algorithm can take a great role to increase the efficiency of disease prevention and improve patient outcomes through early detection and treatment. Huge number of success stories has been proposed and concluded in stroke prediction using both classical and advanced machine learning algorithm.

The Back-propagation algorithm was used to train the ANN architecture, and the same has been tested for the various categories of stroke disease. This research work demonstrates that the ANN based prediction of stroke disease improves the diagnosis accuracy of 89% [1].

Research work by Aditya and et al. [2] performed stroke analysis using classical machine learning algorithm for automatic feature extraction using Support Vector Machines, Margin-based Censored Regression and logistic regression. Research group use three different version on automatic feature extraction. Forward feature selection, L1 regularized logistic regression, Conservative mean selection. They came with the conclusion; the conservative mean feature selection performs very well in CHS dataset. How-ever, they are not sure in other datasets with highly correlated features as it evaluates the performance of each feature individually. To address this problem, they tested with an L1 regularized feature selection algorithm to prune the features before applying conservative mean feature selection for fine-tuning.

Hung and et al. [3] proposed model to utilize DNN on a large-scale EMCs to predict stroke with high UAR and accuracy. In this study, an encouraging AUC of 86% is achieved by both the DNN and GBDT algorithm while DNN requires lesser amount of training data. Their results show performances of DNN and GBDT are superior to that of LR and SVM, both in terms of predictive performance as well as predictive stability.

A. Sudha and et al. [4] proposed a model to predict stroke disease using classification algorithms like a Decision Tree, Naïve Bayes and neural networks for predicting the presence of stroke disease and Principal Component Analysis algorithm for dimensionality reduction.

The stroke dataset is collected from the medical institute. The dataset consists of patient information, patient history, Gene diagnosis disease database which contains the symptoms of stroke disease.

A model for the Prediction of Thrombo-Embolic Stroke in which Artificial Neural Network is proposed to aid existing diagnosis methods. The dataset is collected from 50 patients who have symptoms of stroke disease. All the fifty cases are analyzed after scrutiny with the help of the Physicians and 25 parameters are selected. Further, a backward stepwise method is applied to remove the insignificant inputs from the selected 25 parameters and 20 parameters are finalized.

T. Kansadub and et al. [5] proposed a model in which demographic data is used as a dataset for stroke risk prediction. Firstly, the demographic information was initiated and collected from Faculty of Physical Therapy, Mahidol University, Thailand. Identifying stroke is tedious and time-consuming for medical practitioners. Therefore, automated system for predicting symptoms of stroke from demographic data of patients is needed. The demographic data of patients consisting of gender, age, and education. Thus, according to the 250 stroke patients and 67,897 non-stroke patients, the re-sampling was executed. Consequently, the data of non-stroke were reduced; stroke consisted 250 and non-stroke 500. The nine randomized datasets were created and selected for the best one in terms of similarity to the original data by comparing the difference of normalized attribute values. After the data were ready, three classification algorithms: Decision Tree, Naive Bayes and Neural Network are used for predicting stroke, and the results were compared. In this study, aspect of accuracy, Decision Tree was the best method, but in the aspect of safety of life, Neural Network was the best method because of the highest in FP value and the lowest in FN value.

Two independent research team [6,7] evaluated the RNN in two domain, Energy Load Forecasting and acoustic model. Their work concludes GRU has better performance in time domain than LSTM and RNN. In terms of accuracy, GRU and LSTM beat the RNN. For energy load forecasting GRU has RMSE of 0.58 whereas LSTM has 0.61. On Acoustic Modeling, GRU offered 0.05 to 0.13 benefit in performance.

Chapter 3: Recurrent Neural Networks

3.1 Recurrent Neural Units

The deep learning has transformed the field of machine learning. One of the neural network architecture paradigms that has driven breakthroughs in this field is recurrent neural networks (RNNs). RNNs are a kind of neural network that takes sequential input and produces sequential output by sharing parameters between time steps. RNNs have led to breakthrough results in natural language processing, image captioning, and speech recognition. Though RNNs have proven useful in many sequence and series-based learning tasks, their application to raw time series prediction is still relatively unexplored.

This thesis thus applies established neural network techniques to the stroke prediction problem to produce novel results. Traditional RNNs consists of an input layer, an output layer, and a recurrent layer, as depicted in figure 3.1. They are comprised of a series of weight matrices and activation functions. Explicitly, the set of equations that maps a set of inputs, outputs.

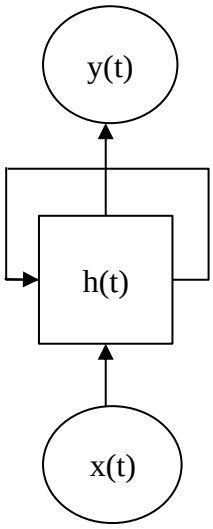

Figure 3. 1 Unfolded Recurrent Unit

Here, new state h_t is derived from previous state h_{t-1} as

$$h_t = f(h_{t-1} + x_t) \quad \text{(3.1)}$$

Where, f(.) is activation function. Hence, RNN is capable to make a prediction based on the hidden state in the previous timestamp and current input. The use case of RNN is nowadays limited as it suffers from vanishing and exploding gradient problem. Error being propagated in RNN at time t is defined as,

$$\frac{dE}{dW} = \sum_{t=1}^{T} \frac{dE_t}{dw} \quad \text{(3.2)}$$

The error for each time-step is computed through applying the chain rule differentiation as,

$$\frac{dE_t}{dW} = \sum_{k=1}^{t} \frac{dE_t}{dy_t} \cdot \frac{dy_t}{dh_t} \cdot \frac{dh_t}{dh_k} \cdot \frac{dh_k}{dW} \quad \text{(3.3)}$$

Here, term $\frac{dh_t}{dh_k}$ refers to the partial derivative of h_t with respect to all previous k time-steps. When such terms become very large then the overall multiplicative values go higher and resulting exploding problem. To minimize this problem, we can set threshold value, and when it reaches the point, we then clip it to some small value. This process is called gradient clipping. In the opposite case, when individual gradient become small, the resulting value goes to zero thus gradient vanishing. To minimize this problem, we use some random initialization of weight and use ReLU activation.

The parameter h_t in RNN serves two purposes.

1. Make an output prediction
2. A hidden state representing the data sequence processed so far.

3.2 Bidirectional Long Short-term Memory

The idea of BLSTMs comes from bidirectional RNN, which processes sequence data in both forward and backward directions with two separate hidden layers [7]. It has been proved that the bidirectional networks are substantially better than unidirectional ones in many fields like phoneme classification and speech recognition. Inspired from the success history of different domain, bidirectional LSTMs is used in stroke prediction problem.

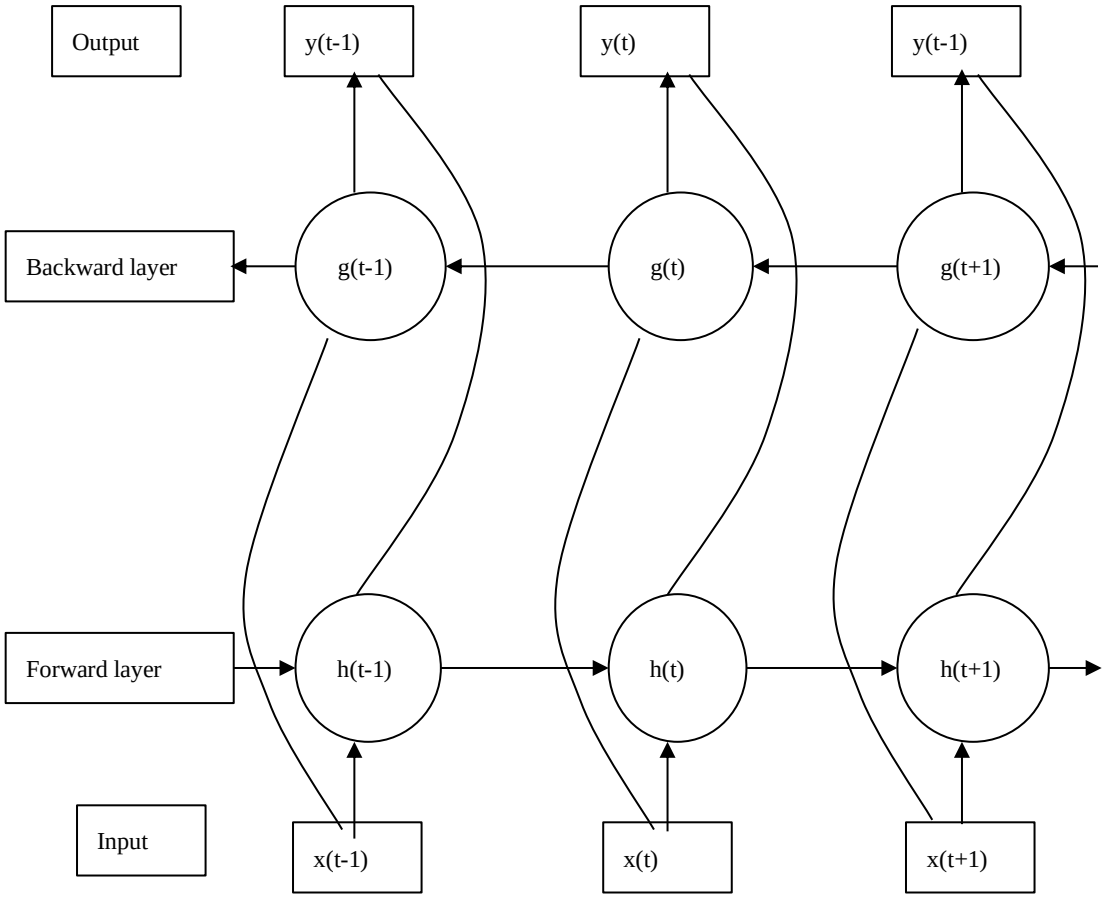

Figure 3. 2 Bidirectional Long Short-term Memory

The LSTM with input, forget and output gate is capability with long time lag, state reset and resolve temporal distance. But all states are accumulated forward. In real task, the system is usually requested to take backward information into consideration.

Mathematically,

$$h_t = \text{forward pass} = \sigma(W_{xh}.x_t + W_{hh}.h_{t-1} + b_h) \quad (3.4)$$

$$g_t = \text{backward pass} = \sigma(W_{xg}.x_t + W_{gg}.g_{t+1} + b_g) \quad (3.5)$$

$$y_t = W_{hy}.h_t + W_{gy}.g_t + b_y \quad (3.6)$$

Supposed the one piece of sequence is from time 0 to time T, the forward and backward pass algorithm is followed by:

Forward Pass feed all input data for the sequence into BDLSTM and calculate all predict output.

1. Do forward pass for forward states (from time 0 to time T), save all the cell output through time.
2. Do forward pass for backward states (from time T to 0), save all the cell output through time.
3. Do forward pass for output layer by adding two saving results.

Backward Pass Calculate the error function derivation for the sequence used in the forward pass

1. Do backward pass for output neuron
2. Do backward pass for forward states (from time T to time 0). Then do backward pass for backward states (from time 0 to T)

3.2.1 Activation Function

Neural network contains three element, input, node and output. Information is processed from one node to another node based on input values. The node will pass the information only when it gets triggered (cross the threshold value). Such controlling of passing

information from one node to another node is accomplished by activation function. Activation function holds two properties- Linearity and Differentiability. Linearity properties defines how input computation will map to output values. Differentiability properties define how its first derivative works. Differentiability also defines the gradient based learning phenomenon.

If an activation function is linear(say, $y = mx + c$), its first derivative will be always 1 thus, less opportunity for learning. Whereas, if activation function is nonlinear (say, $f(x)=1/(1+e^{-x})$), there will be a higher degree of learning opportunity as it provide same order of first differential value.

Hence, this function is divided into two groups,

1. Linear activation functions, thus gradient based learning is not possible.
2. Non-linear activation functions, thus gradient based learning is possible.

Four activation functions, Sigmoid, Tanh, ReLU and Leaky ReLU are described below.

a. Sigmoid

A sigmoid function is nonlinear function and produces a curve with an "S" shape. This function is well known for binary classifier. It is defined as,

$$f(x)=1/(1+e^{-x}) \quad \text{..... (3.7)}$$

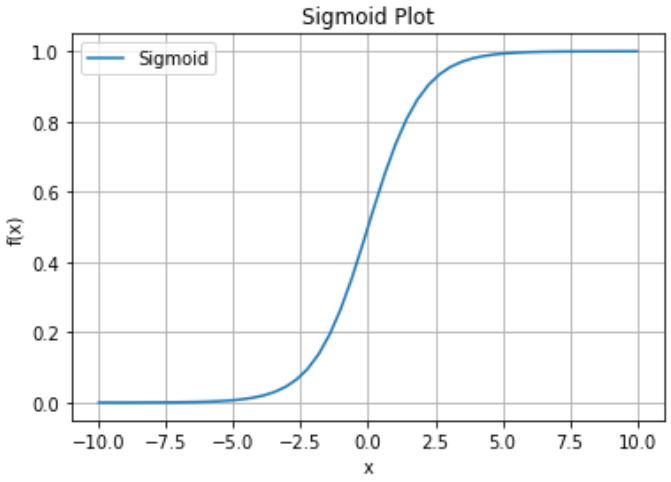

Figure 3. 3 Sigmoid Plot

Figure 3.3 shows, sigmoid activated output limits on (0,1). One serious problem with sigmoid function is it suffers from vanishing gradient, means, when output value of sigmoid become very high (almost 1) or very low (near 0), its first derivative will be very low (<<1). This causes vanishing gradients and poor learning for deep networks

b. Hyperbolic Tangent

The hyperbolic tangent (tanh) non-linearity is defined by the formula,

$$f(x) = (e^{-x} - e^{x})/(e^{-x} + e^{x}) \quad \text{..... (3.8)}$$

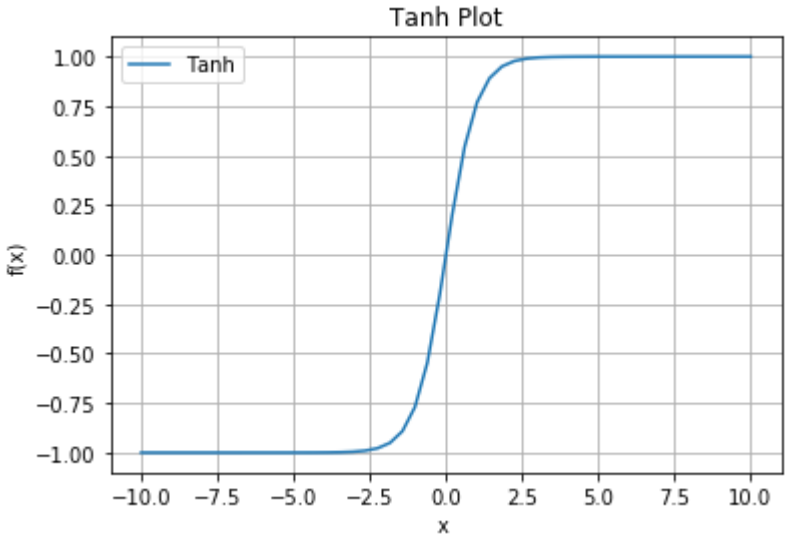

Figure 3. 4 Tanh Plot

The $\tanh(x)$ function can be viewed as a rescaled version of the sigmoid, and its Output falls in the interval of $(-1, 1)$, as illustrated in Figure 3.4.

c. Rectified Linear Unit

The Rectified Linear Unit (ReLU) is defined as

$$y = 0, \text{when } x \leq 0$$
$$= x, \text{when } x > 0 \quad \ldots\ldots (3.9)$$

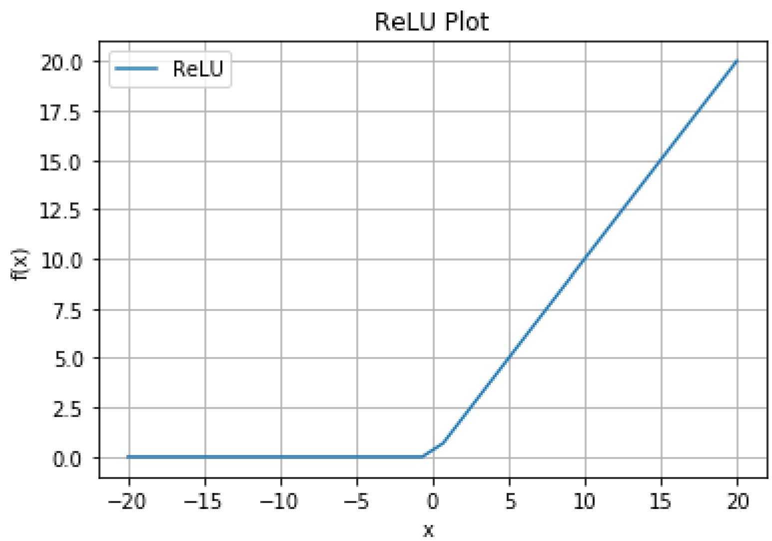

Figure 3. 5 ReLU Plot

The output from ReLU bounds to $(0, \infty)$. The first derivative is 0 for all negative inputs and 1 for all positive. So whenever input values fall in negative range, back propagation is zero, thus certain weights become dead.

d. Leaky ReLU

Leaky ReLUs are one attempt to fix the dying ReLU problem. Instead of the function being zero when x < 0, a Leaky ReLU have a small positive gradient for negative inputs. Leaky ReLU has the following mathematical form:

$$y = 0.01 * x, \text{when } x < 0$$

$$= x, \text{when } x \geq 0 \quad \text{..... (3.10)}$$

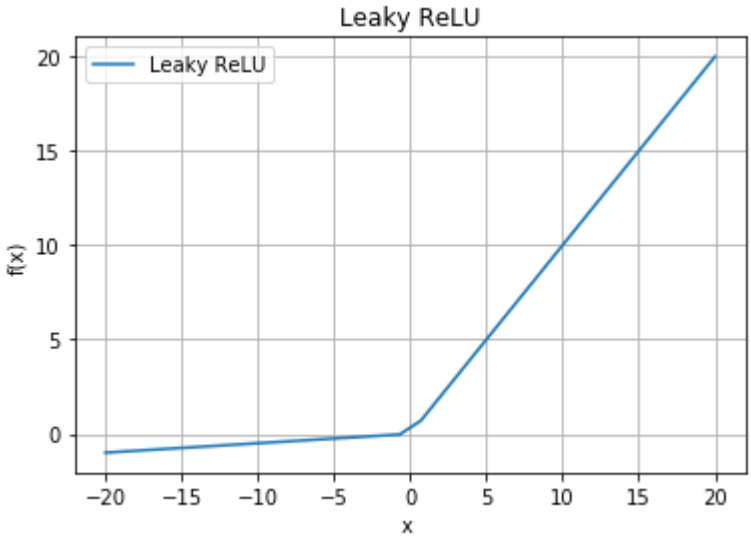

Figure 3. 6 Leaky ReLU Plot

3.2.2 Optimizer

3.2.2.1 Stochastic Gradient Descent

Generally speaking, neural networks employ some form of minibatch stochastic gradient descent (SGD) to learn an appropriately good setting of the weight matrices with respect to some loss function, L(x). The mini batch SGD update with respect to θ [8].

$$\theta_{t+1} = \theta_t - \alpha . \sum_{s_t \in X} \frac{\delta L(x)}{\delta \theta} \quad \text{..... (3.11)}$$

3.2.2.2 RMSProp

Minibatch SGD by itself is poorly suited for training Neural Network. The method is extremely sensitive to the choice of learning rate. If too large learning rate is chosen, the learning is unstable with parameters oscillating wildly around a local optimum. If the learning rate is set too small, and learning becomes intolerably slow. Furthermore, a single learning rate that was well suited early in optimization may end up non-ideal somewhere else in the parameter space. There are numerous adjustments to mini batch SGD that seek to remedy these issues.

RMSprop [11], update consists of the following two steps:

$$E[g^2]_t = 0.9 * E[g^2]_{t-1} + 0.1\, g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(E[g^2]_t + \varepsilon)}} g_t \qquad \text{……..…….....} \quad (3.12)$$

Where,

g_t is the minibatch gradient calculated at iteration respect to parameter $\eta, \theta_t, E[g^2]_t$ is an approximation to the running average of squared gradients, and η is user-set hyperparameter that adjusts the rate of learning.

In order to keep the parameter, update roughly consistent, the algorithm scales the learned gradient by the square root of a running average of squared gradients $\sqrt{E[g^2]_t}$

The running average as defined here gives much higher weight to recently calculated gradients which are more likely to have a magnitude similar to the one at the current time step, and thus better ensures that the parameter is updated by a consistent amount. By ensuring the size of the actual parameter update is consistent, learning will be more stable without the need for a cripplingly slow learning rate, and thus more rapid.

3.2.2.3 Adaptive Moment Estimation (Adam)

Adam, a method for efficient stochastic optimization that only requires first-order gradients with little memory requirements. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients [9]. It combines the advantages of two recently popular methods: AdaGrad (Duchi et al., 2011), which works well with sparse gradients, and RMSProp (Tieleman & Hinton, 2012), which works well in on-line and non-stationary settings. Good default settings for the tested machine learning problems are $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, e = 10e - 8$

Algorithm run as, Initialize m_0, v_0 and $t = 0, \alpha, \beta_1, \beta_2,$

$$g_t = df/dt$$

$$m_t = [\beta_1 m_{t-1} + (1 - \beta_1).g_t]/[1 - \beta_1]$$

$$v_t = [\beta_2 v_{t-1} + (1 - \beta_2).g_t]/[1 - \beta_2]$$

And finally, weight updates as

$$w_t = w_{t-1} - \alpha.m_t/(\epsilon + \sqrt{v_t}) \quad \text{……..……..... (3.13)}$$

3.2.2.4 Adaptive Moment Estimation (Nadam)

Nadam [10], is an acronym for Nesterov and Adam optimizer. Nesterov factor has been added in Adam optimizer. The momentum, m contains the gradient update for the current timestamp $g_t$in addition to the momentum vector update for the next timestamp $t + 1$, which needs to be applied before taking the gradient at the next timestamp.

3.2.3 Loss Functions

A loss function is minimized as the objective to measure the compatibility between a prediction and the ground truth labels.

3.2.3.1 Quadratic Cost

Quadratic cost is also known as the Mean Squared Error (MSE). It is defined by the formula

$$\text{Loss}\,(x, y) = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i|^2 \quad \text{…….……..... (3.14)}$$

Where x represents n predictions of neural networks, and y represents the real classes of the input data.

3.2.3.1 Cross Entropy

The cross entropy of a distribution P with respect to a distribution Q measures how many bits are needed on average to encode data from P with the code that is optimal for Q. It is defined as,

$$\text{Loss}\ (x, y)\ = -\ \sum_{i=1}^{n} y_i * \log \frac{\exp(x_i)}{\sum_{j=0}^{n} \exp(x_i)} \quad \text{..... (3.15)}$$

Where x is a vector of n predictions of neural networks, and y is a binary vector full of 0s and 1s representing the real classes of the input data.

3.2.4 Batch Normalization

Batch normalization potentially helps neural network to converge faster and get higher overall accuracy. Batch Normalization shifts inputs to zero-mean and unit variance, which makes the inputs of each trainable layer comparable across features. It allows singular functions such as TanH and Sigmoid to not get stuck in the saturation mode, where the gradient is almost 0. In practice, networks that use batch normalization are more robust to bad initialization.

3.2.5 Dropout Regularization

Large neural network models consist of many parameters and perform extremely complex manipulation. Model will often overfit on the training set and lose generalizability and accuracy on the test set. In this work, dropout regularization with a dropout probability of $p = 0.42$ has been employed to address the problem of over fitting. Dropout regularization independently sets each weight in consideration to zero with probability p in response to this, the network cannot rely on a few weights per-example to predict an outcome, lest those weights get pruned in a training step. The model is thus forced to employ many weights to process and predict each example, reducing over fitting. We deliberately avoid dropout on weights between time steps, as doing so effectively eliminates long-range memory.

Chapter 4: Research Methodology

4.1 System Design and Development

4.1.1 Functional block diagram

The Overall research methodology involves the following steps. The workflow follows from data collection, data preprocessing, model implementation and results and analysis. The methods and process followed in each step are described in preceding sections.

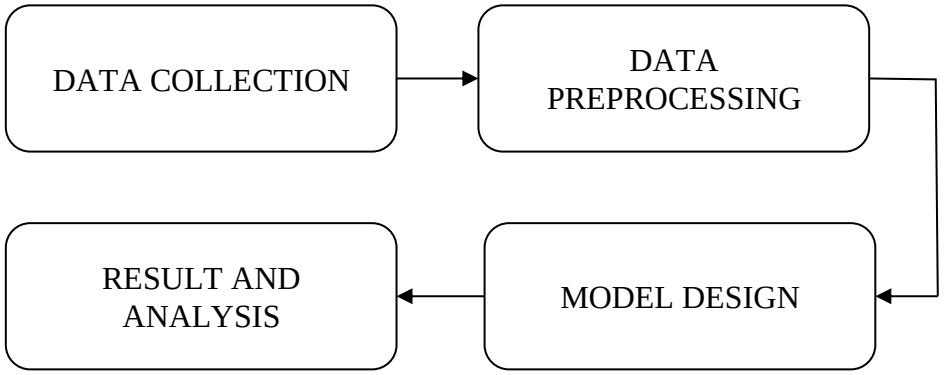

Figure 4. 1 Functional block diagram

4.1.2 Flow Chart

Research workflow diagram is shown in figure 4.2. The flow diagram incudes raw data feeding from CSV, pre-processing of data, Model validation, and result and graphing.

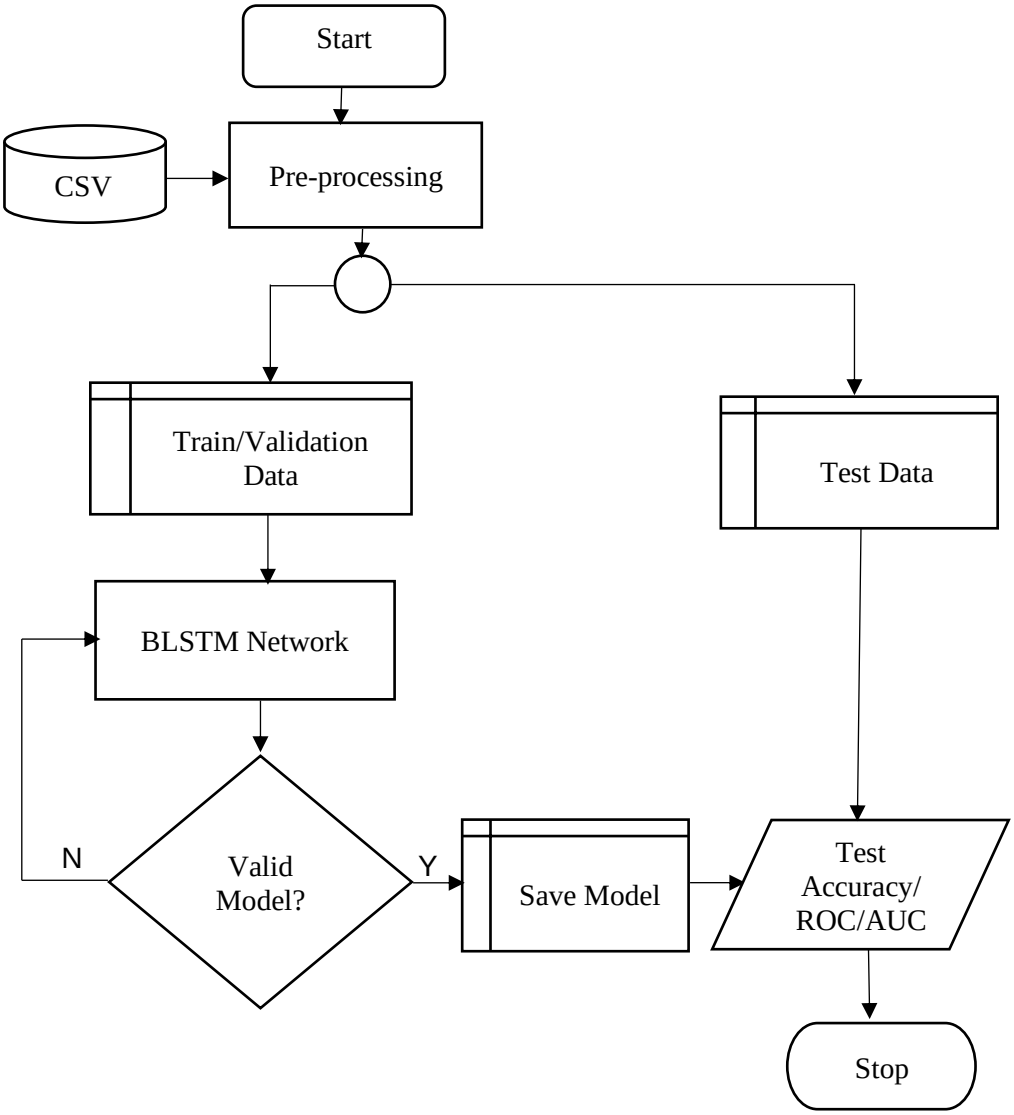

Figure 4. 2 Flow Chart

4.2 Datasets

In machine learning and Deep learning algorithms for classifying and recognizing, data plays a vital role. Having too many data is always recommended to avoid the underlying problem associated with machine learning. Training with low volume of data is always susceptible to under fitting the model we will use.

Overall experiment has been carried out in openly available data set, Kaggle Open Stroke Data Set. Data has been divided into two sections, training and testing set. Each set has data record of counts 42000 and 18000 accordingly. Training set has 10 features set as,

Table 4. 1 Feature Set

1	id
2	gender
3	age
4	hypertension
5	heart_disease
6	ever_married
7	work_type
8	residence_type
9	avg_glucose_level
10	bmi
11	smoking_status

Two row extracted from dataset.

[5374,Male,23,0,0,No,Private,Rural,93.74,31.2,never smoked,0]

[56669,Male,81,0,0,Yes,Private,Urban,186.21,29,formerly smoked,1]

4.3 Data Preparation

Data preparation refers to the process of transformations on source data before feeding it to the algorithm. We should ensure data preprocessing for achieving better results from the applied model in Machine Learning the format of the data has to be in a proper manner. Data process goes through following phase.

1. Feature Representation

   The neural network takes numeric data as input, performs complex computation and generate numeric output. The dataset in this research work is not what is expected for the neural network input feeds. Thus, proper data representation is done here. Let's take two rows from the original dataset.

   [5374,Male,23,0,0,No,Private,Rural,93.74,31.2,never smoked,0]

   [56669,Male,81,0,0,Yes,Private,Urban,186.21,29,formerly smoked,1]

   After feature representation, this looks like,

   [5374,1,23,0,0,0,1,0,93.74,31.2,0,0]

   [56669,1,81,0,0,1,1,1,186.21,29,1,1]

2. Data cleaning

   The purpose of data cleaning is to fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies. The dataset here used is comprised of attributes with some missing in attribute, it is required to remove such data from dataset before model is trained. Take one row extracted from dataset after feature representation.

Table 4. 2 Raw data

id	gender	age	hypertension	heart_disease	ever_married	work_type	residence_type	avg_glucose_level	bmi	smoking_status	stroke
31091	1	34	0	1	1	1	1	106.23		1	0

The row with BMI missing data should be removed during data cleaning process. If this row is fed for neural network, it will start to learn from noisy data as well.

3. Data Normalization

   Normalizing refers to rescaling each observation (row) to have a length of 1 (called a unit norm in linear algebra). One experiment is done to demonstration the learning behaviors without data normalization later in below result and analysis section. The single row after data-normalization looks like

   [3.92900000e+04 5.00000000e-01 1.21093750e-01 0.00000000e+00
   0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
   2.72061004e-01 1.13142857e-01 1.00000000e+00 0.00000000e+00]

   Here, min-max normalization technique is used to preserve the originality of dataset.

4.4 Model Design

The proposed model for the research work used bidirectional long short-term memory recurrent network. BLSTM model in the research work has a series of layers. Architectural contains three layers.

1. Embedding Layer
2. BLSTM Layer
3. Dense Output Layer

Layer (type)	Output Shape	Param #
embedding_5 (Embedding)	(None, None, 256)	256000
bidirectional_5 (Bidirectional)	(None, 256)	394240
dropout_5 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 1)	257

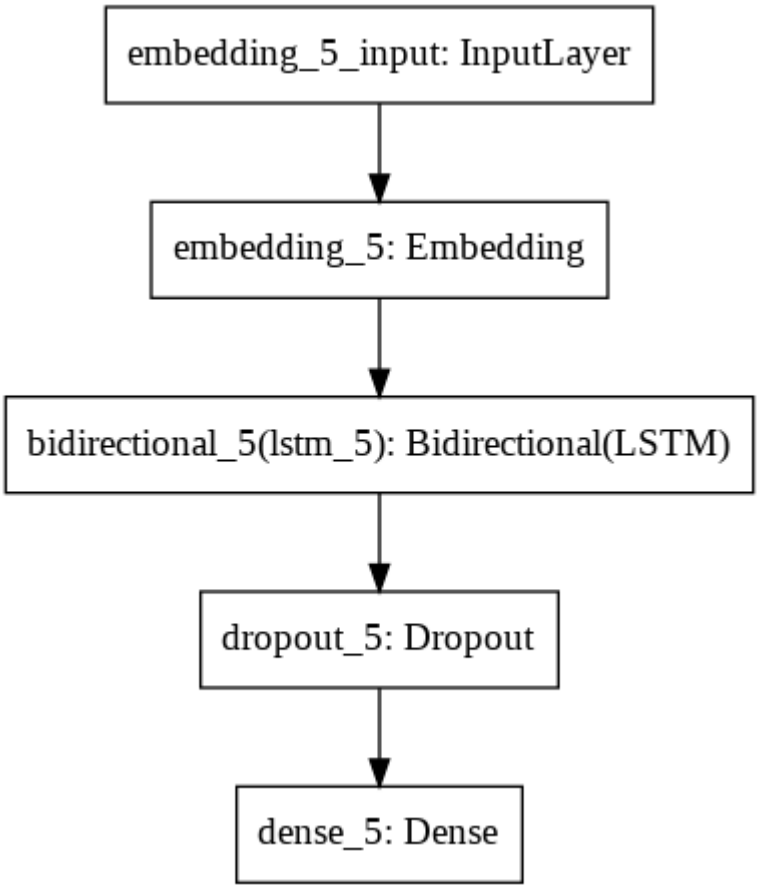

Figure 4. 3 BLSTM Design in Keras

Each layer works as,

1. Embedding layer

   The Embedding layer is used to create input vectors for incoming data. It sits between the input and the BLSTM layer, i.e. the output of the Embedding layer is the input to the LSTM layer.

2. BLSTM Layer

   This is a single unit of BLSTM with configuration of 256 output vector. This output goes to densely connected layer which is our final layer.

3. Dense Output Layer

   This is our final layer which takes input vector of 256 size and result as output data.

4.5 Tools and Environments

4.5.1 Python, Keras and Tensorflow

Python is an interpreter, high-level, general-purpose programming language. This language has been proved as first research language as of large number of in-built and open sourced libraries. General syntax of this language is based on indentation with high user-friendly keywords.

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation for deep learning. It supports both convolutional networks and recurrent networks, as well as combinations of the two and seamlessly run on CPU and GPU.

4.5.2 Environments

The system configuration of the overall research work is carried under following test bed.

Table 4. 3 Development Environment

Platform	Google Colab
Operating System	Ubuntu 18.04.2 LTS
Memory	12 GB
GPU	NVIDIA® Tesla® K80

Chapter 5: Result and Analysis

The different version of recurrent neural network is trained in 8:2 data set ratio. Series of experiments conducted under different optimizer and activation function. In all experiment dropout of 0.42 is implemented.

5.1 Experiment on stochastic gradient descent (SGD) optimizer

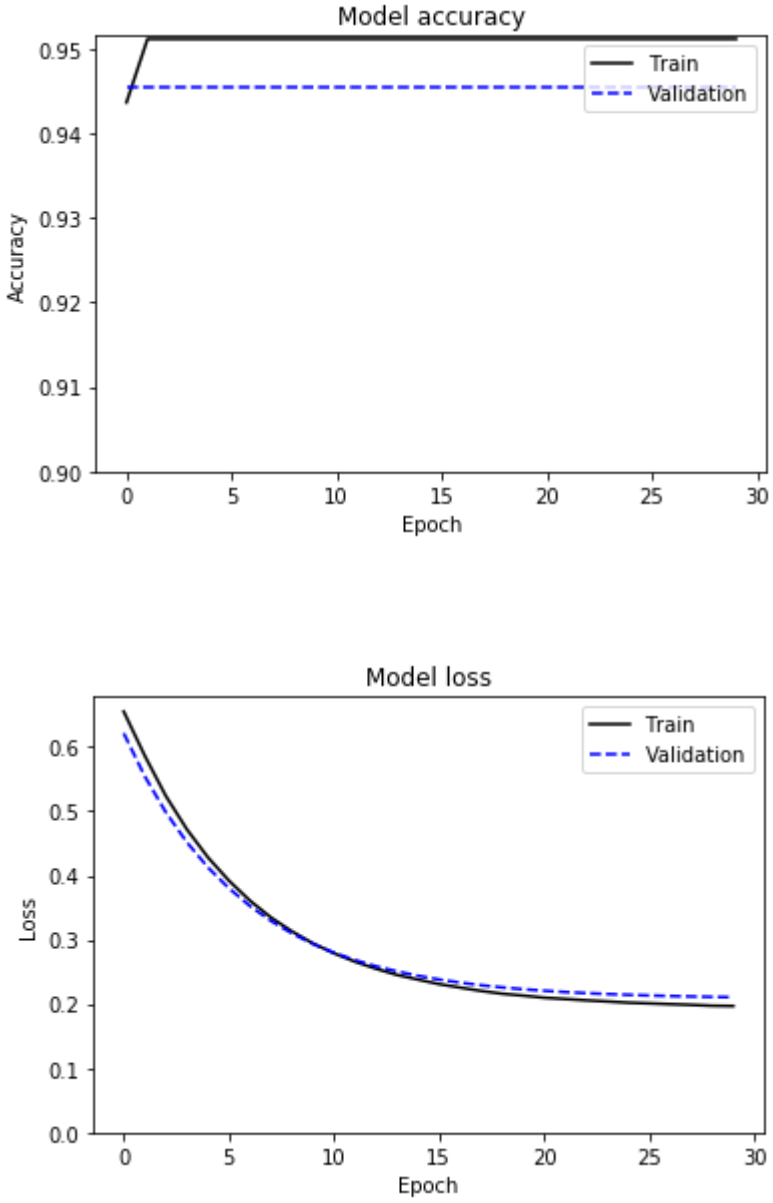

Figure 5. 1 Experiment on SGD Optimizer

The BLSTM network is trained with SGD, stochastic gradient descent optimizer. The loss curve shows it has reached optimal state in around 30 epoch. The learning rate for the optimizer is set on 0.001 with momentum 0.95.

5.2 Experiment on RMSProp Optimizer

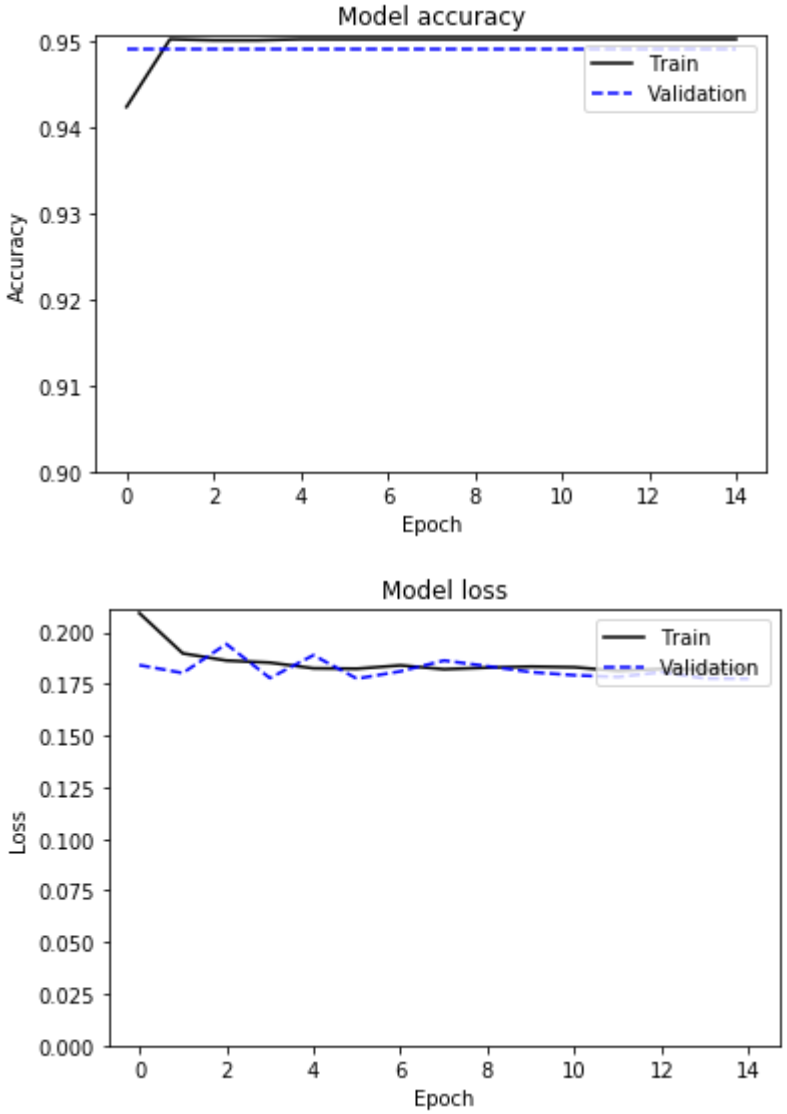

Figure 5. 2 Experiment on RMSProp Optimizer

The experiment is done with RMSprop Optimizer for runtime parameter, learning rate 0.001. The model reached optimal state in around 15 epochs.

5.3 Experiment on Adam Optimizer

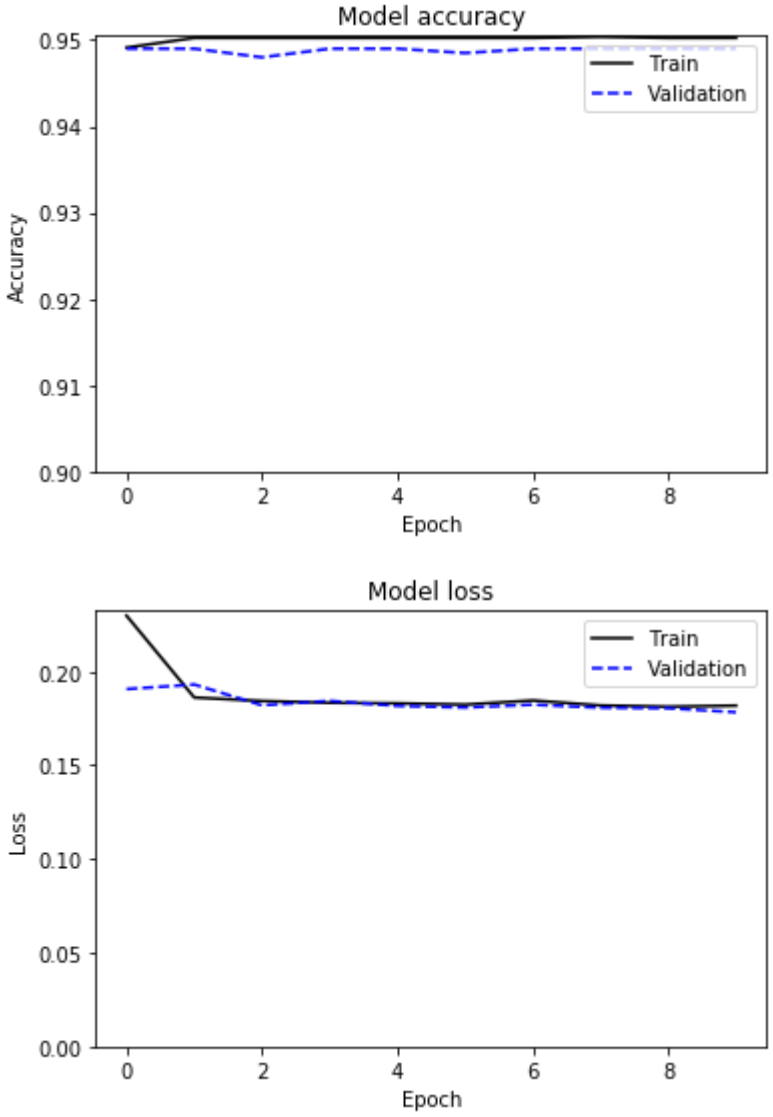

Figure 5. 3 Experiment on Adam Optimizer

The experiment with Adam optimizer under runtime parameter as Learning rate=0.001, beta_1=0.9 and beta_2=0.999 reached the optimum state around 10 epochs.

5.4 Experiment on Nadam Optimizer

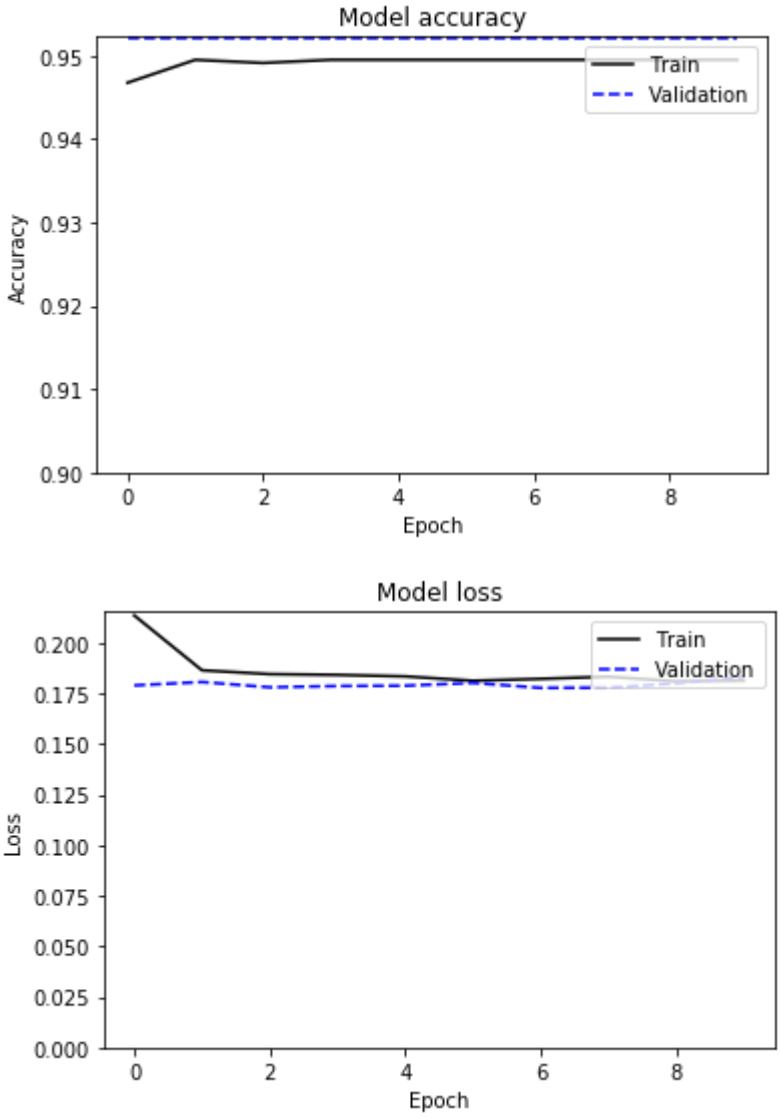

Figure 5. 4 Experiment on Nadam Optimizer

The network is now fed with Nadam as optimizer under parameter, learning rate=0.002, β_1=0.9, β_2=0.999. The model gets optimized under 10 epochs similar to Adam optimizer.

5.5 Learning with high volume dataset

The experiment now run for 42K training dataset. The experiment is done with the Adam optimizer with learning rate 0.001. It shows with higher volume of dataset, model reached its optimized state in 3rd epoch with validation accuracy of 98%.

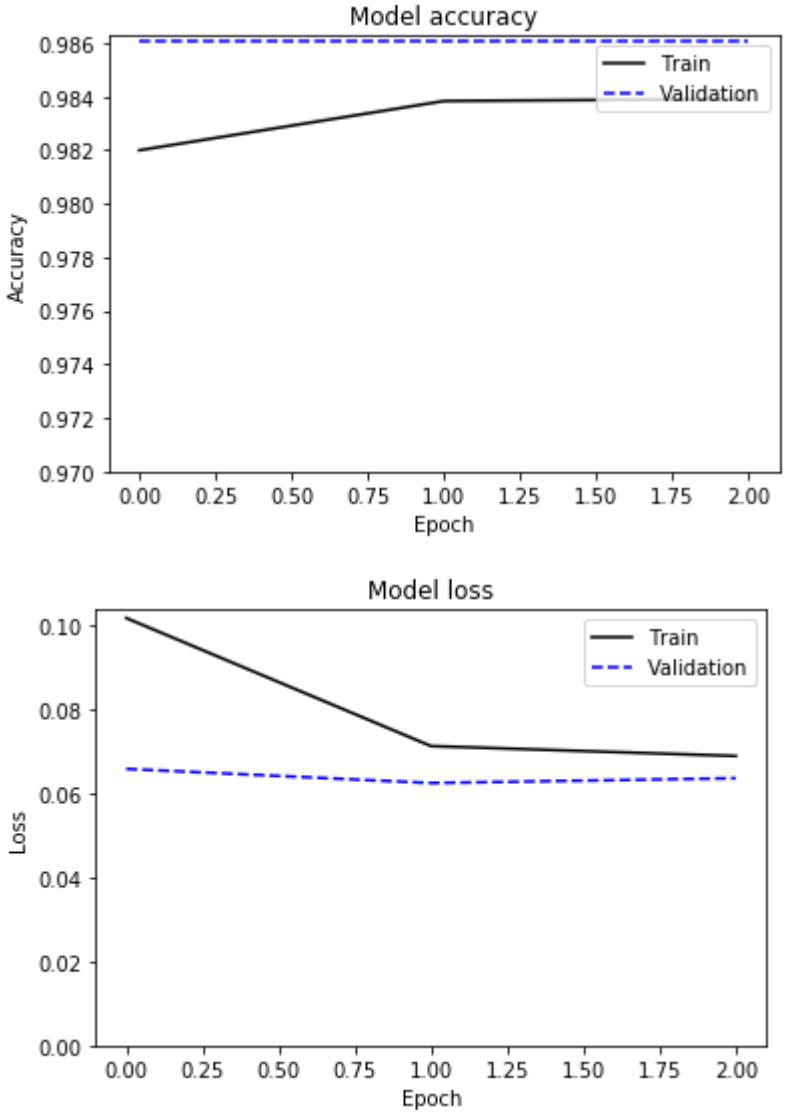

Figure 5. 5 Learning with high volume dataset

5.6 Learning without data normalization

The experiment conducted in un-normalized dataset with Adam optimizer. The validation vales on loss curve starts to rise after 4 epochs as shown in figure 5.6.1, thus data normalization needs to be done for the model to learn faster and smoother. The loss graph for normalized dataset is shown in figure 5.6.2. Data normalization makes sure different features range in same values (0,1), hence model gradient reach global minima quickly.

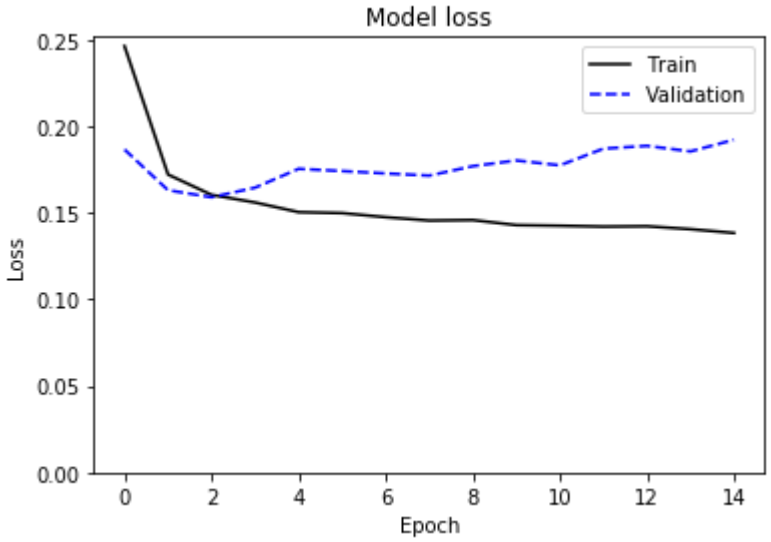

Figure 5.6. 1 Learning without data normalization

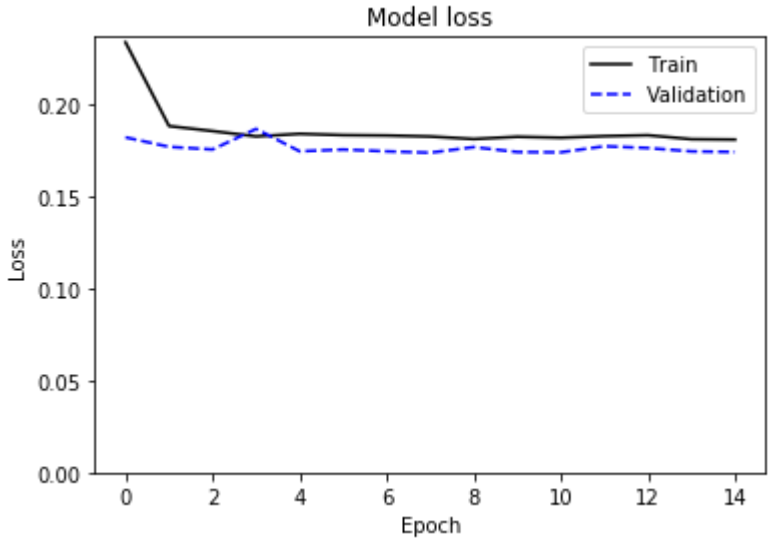

Figure 5.6. 2 Learning with data normalization

5.7 Dropout Analysis

This analysis figures out the drop ratio of network during model training. Setting too low drop-out will cause model to overfit on training data and too high cause the low learning ability. This experiment concludes our model has reached optimal state in drop-out ratio in the range 0.4 to 0.45. Following graph illustrate their loss behavior on training and validation data.

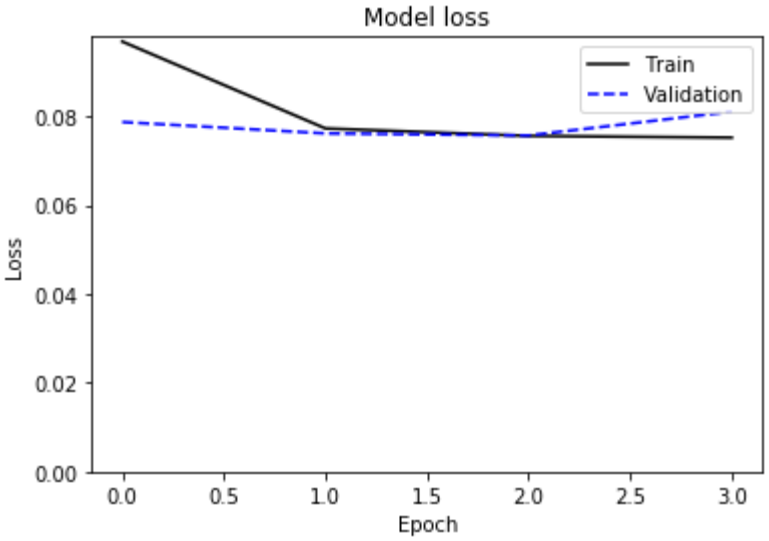

Figure 5.7. 1 Dropout ratio=0.2

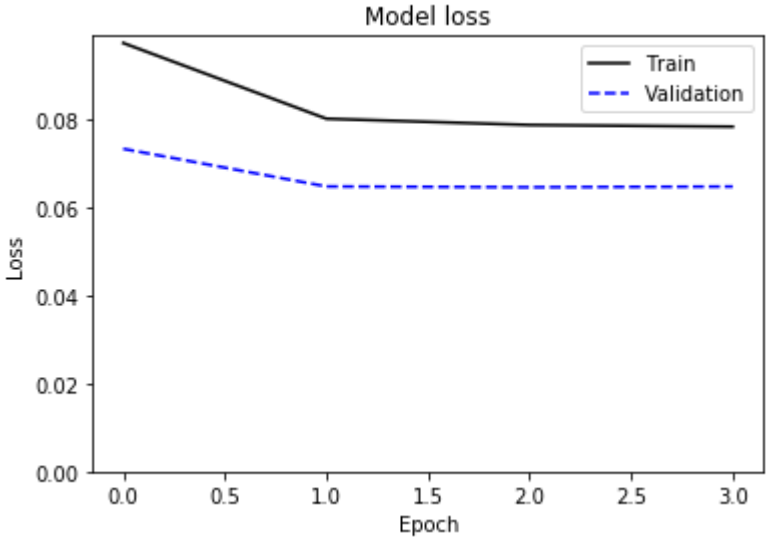

Figure 5.7. 2 Dropout ratio=0.3

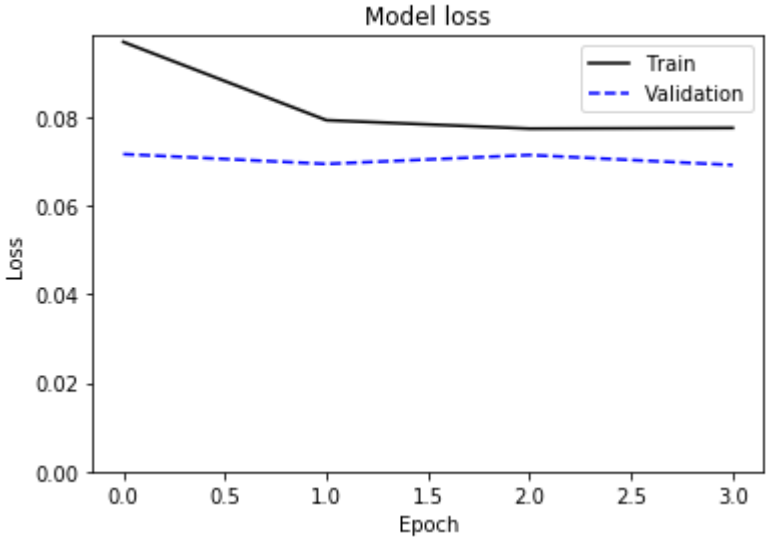

Figure 5.7. 3 Dropout ratio=0.40

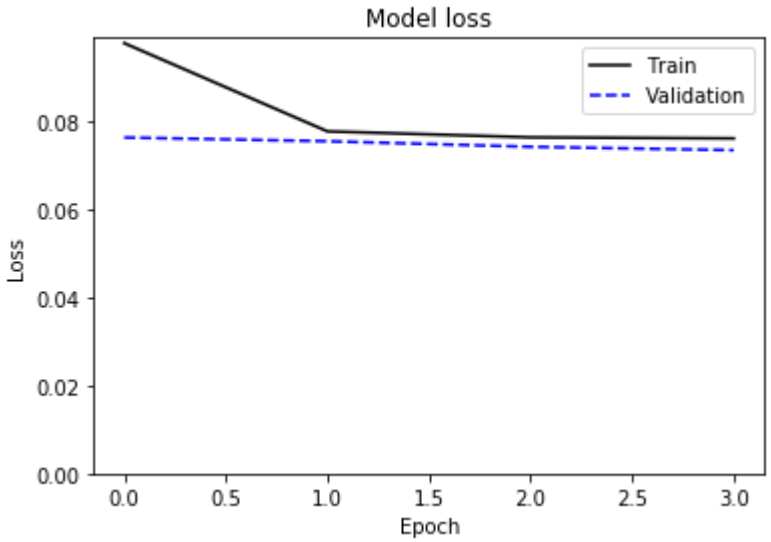

Figure 5.7. 4 Dropout ratio=0.42

5.8 Activation Function Analysis

Number of experiments are carried out with different activation function. The following section list out the respective results.

5.8.1 Observation with SoftMax function

The experiment with SoftMax has not gain any learning curves. The experiment is run for 5 epochs, but model's performance is not improving.

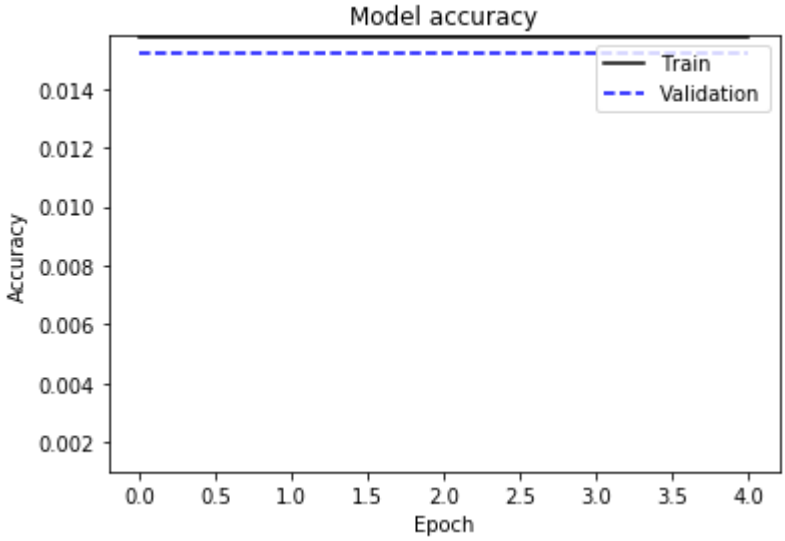

Figure 5.8. 1 Experiment on SoftMax function

5.8.2 Observation with sigmoid function

Under the sigmoid function model has reached 98% of training accuracy.

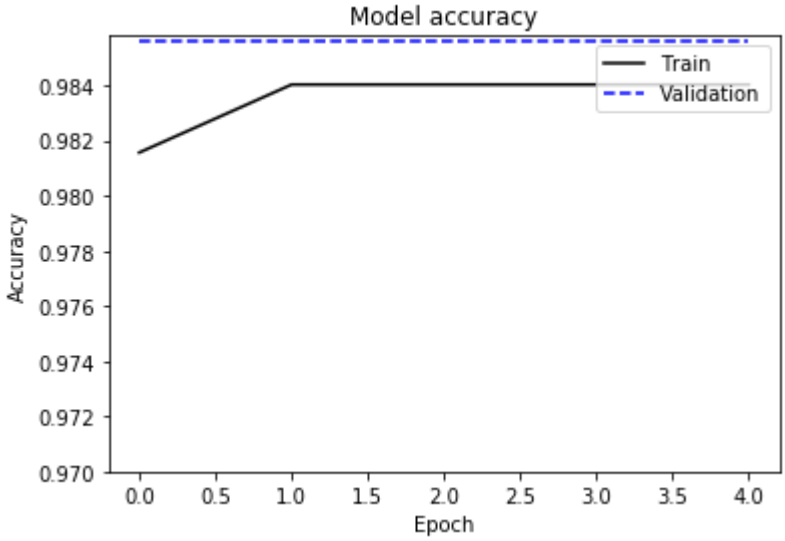

Figure 5.8. 2 Experiment on sigmoid function

5.8.3 Observation with exponential function

Model has pretty well performance on exponential functions as well. Training accuracy is a little bit low from sigmoid activation function.

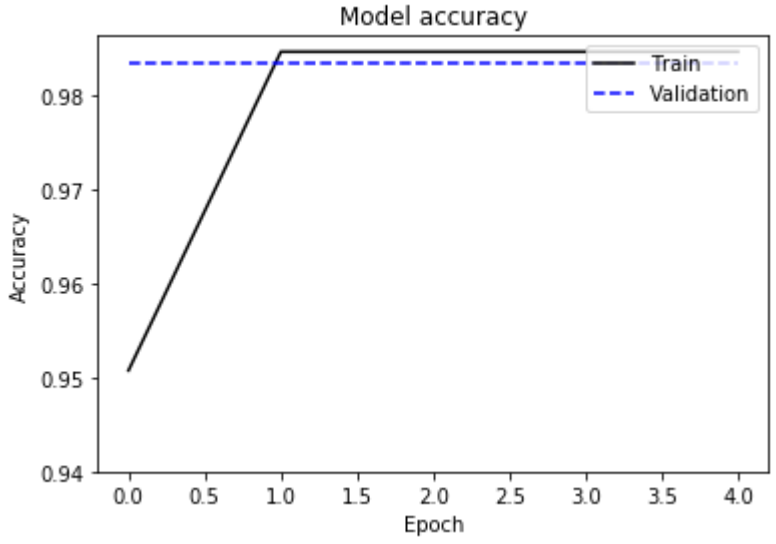

Figure 5.8. 3 Experiment on exponential function

By analyzing experiment 5.8.1, 5.8.2 and 5.8.3, it has been noticed that sigmoid activation has best performance over exponential and SoftMax.

5.9 Validation data ratio analysis

Four experiment is done to perform validation data analysis. Experiment is conducted on 6:4, 7:3, 8:2 and 9:1 data ratio. Figure below shows the respective loss graph. On the experiment it has concluded that optimum model is achieved at validation ratio 8:2.

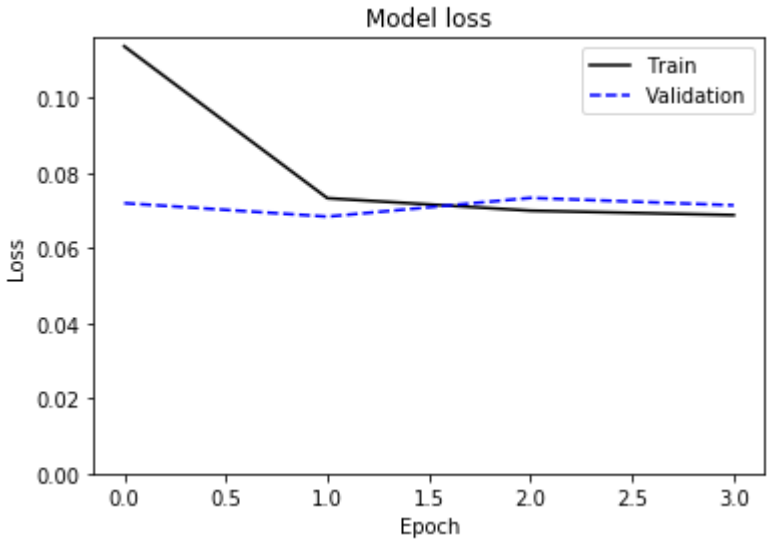

Figure 5.9. 1 Validation ratio 6:4

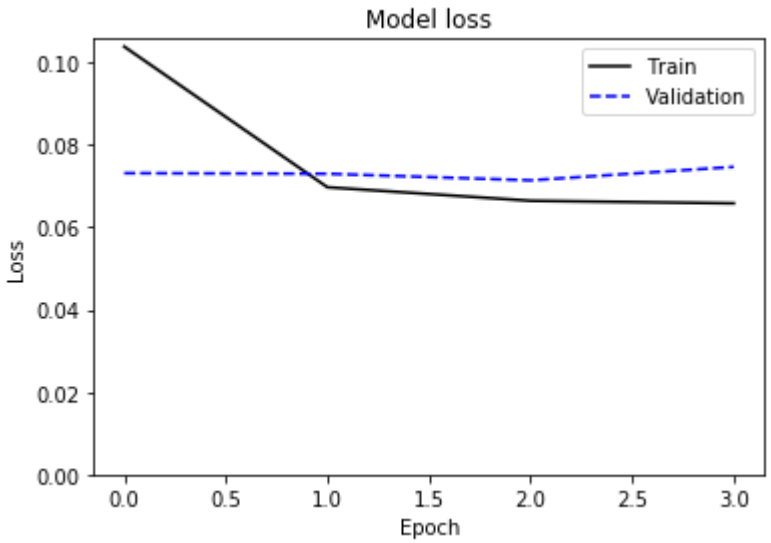

Figure 5.9. 2 Validation ratio 7:3

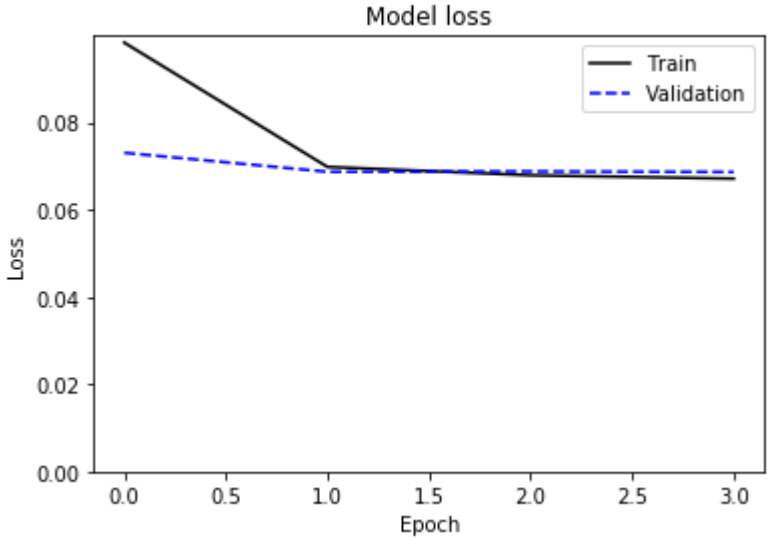

Figure 5.9. 3 Validation ratio 8:2

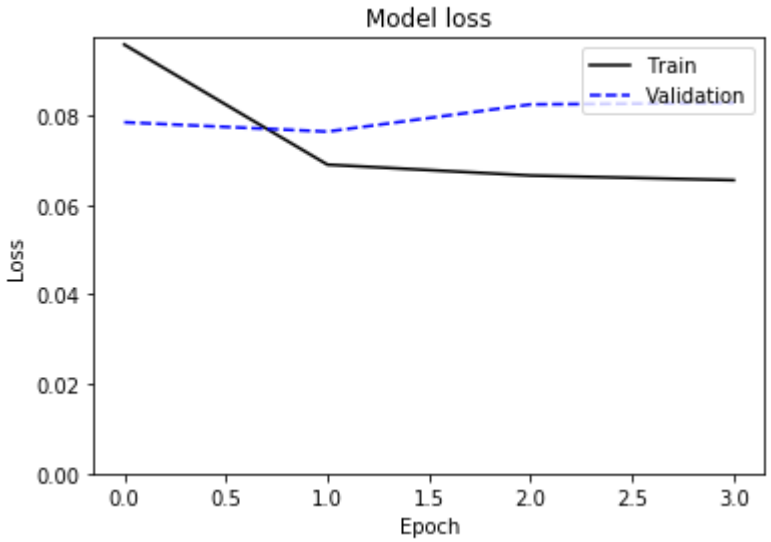

Figure 5.9. 4 Validation ratio 9:1

Looking into validation ratio loss curve of 6:4 in figure 5.9.1, 7:3 in figure 5.9.2 and 9:1 in figure 5.9.4, validation loss is higher and is not steady as seen for validation ratio 8:2 in figure 5.9.3.

5.10 Comparative Analysis

5.10.1 10K Training Set

The experiment has been conducted in 10K dataset. Overall performance on different algorithms are summarized below.

Table 5.10. 1 Performance on 10K dataset

Algorithm	BLSTM		LSTM		GRU	
Optimizer	Train Accuracy	Test Accuracy	Train Accuracy	Test Accuracy	Train Accuracy	Test Accuracy
Adam	0.9353	0.80266	0.93237	0.8	0.930807	0.8
RMSProp	0.9249	0.79801	0.93059	0.7986	0.93255	0.799
Nadam	0.9346	0.80024	0.931487	0.8101	0.93284	0.8003
SGD	0.9144	0.7921	0.931254	0.7913	0.9083	0.7944

The table above shows, under the dataset training accuracy has reached about 93% and testing in a range of 81%. The testing is performed with different optimizer and Adam showed it as best of all. Following ROC curve shows the marginal region for true positive rate and false positive rate.

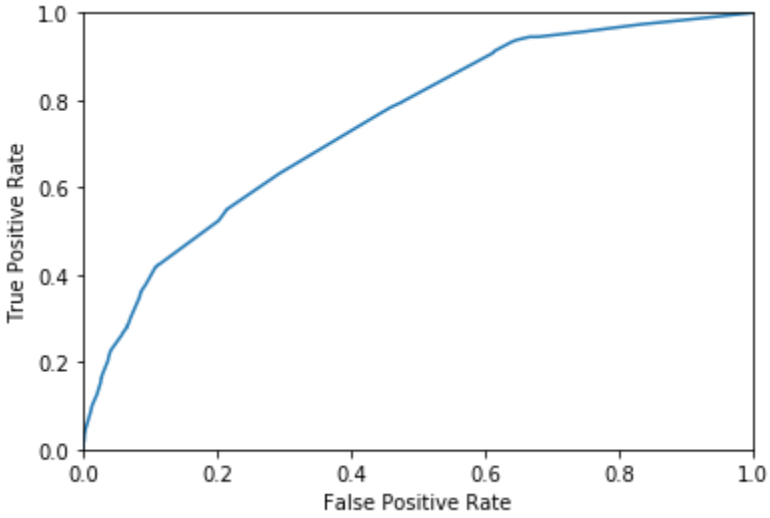

Figure 5.10. 1 ROC analysis for BLSTM with Adam Optimizer

The curve nature shows model has classified the problem pretty well. It has reached nearly 80% of the area under the curve.

5.10.2 42K Training Set

The experiment now conducted for 42K dataset. All of the training algorithm has reached its saturation state under 4 epochs. Table below summarizes the overall performance.

Table 5.10. 2 Performance on 42K dataset

Algorithm	BLSTM		LSTM		GRU	
Optimizer	Train Accuracy	Test Accuracy	Train Accuracy	Test Accuracy	Train Accuracy	Test Accuracy
Adam	0.98439	0.91233	0.98287	0.91173	0.98369	0.91224
RMSProp	0.98296	0.91102	0.983105	0.9105	0.98366	0.9102
Nadam	0.983118	0.91152	0.98395	0.9101	0.98399	0.9113
SGD	0.97955	0.90511	0.976177	0.9011	0.972289	0.9061

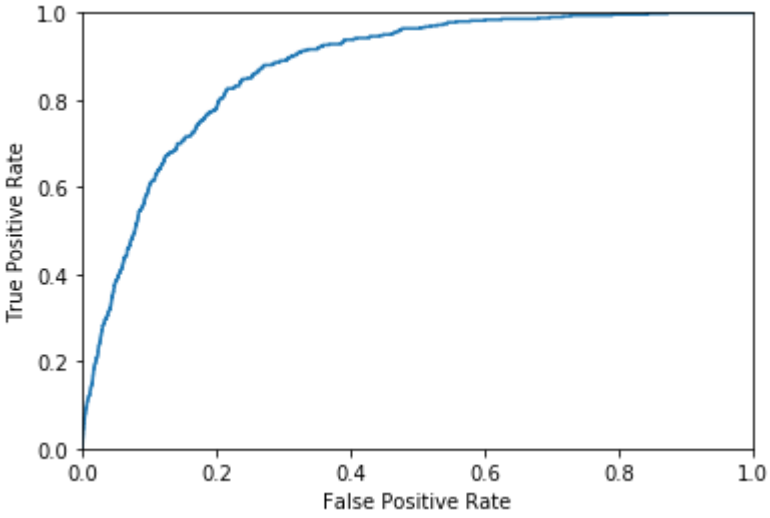

Figure 5.10. 2 ROC analysis for BLSTM with Adam Optimizer

On larger dataset, BLSTM with Adam as optimizer has won the competition with training accuracy of 98%. The test accuracy also reached 91%. ROC curve for the BLSTM with Adam

as optimizer is shown in figure 5.9.2. The ROC curve has supported the classifier is able to reach nearly 90% of the area under the curve. The AUC is little smaller or larger than test accuracy score as AUC is based on multiple threshold calculations.

5.10.3 Confusion matrix analysis

Confusion matrix analysis gives the measurement of how truly or badly classified the predictive problem. This research work is validated with confusion matrix on test data for its accuracy output. For this unseen test data with 143 positive class and 357 negative class are fed to models' output. Table 5.10.3 summarize the model accuracy. Model has reached true positive rate of 0.8 and true negative of 0.95. The slight variation of accuracy is due to imbalanced training dataset, where major of dataset are in negative class. Out of 42K dataset only 783 sample are in positive class.

Table 5.10. 3 Confusion matrix

	Stroke (Predicted)	Non-Stroke (Predicted)
Stroke (Actual)	114	29
Non-Stroke (Actual)	16	341

Chapter 6: Conclusion

This research conducted for series of experiments using different optimizer and three kinds of recurrent neural network. List of experiment shows that, BLSTM networks has best performance over all other RNN variant network. In 42K dataset, BLSTM reach 98% of validation accuracy.

This work points that model has reached optimum state trained with Adam as optimizer in 0.001 learning rate with sigmoid activation function in final layer. Data set partition of 8:2 from training to validation and use of 0.42 drop-out gives the best performance. This work suggests increasing training dataset to achieve higher accuracy.

This research work shows, it is the best model design among task done by other researchers. Researcher Hung and et al. [2] designed model using DNN which achieved the accuracy of 86%. The work done by M. Singh and P. Choudhary [1] was used ANN architecture and achieved the accuracy of 89%. Result comparison shows this work gained better performance over those ones.

Chapter 7: Further Enhancement

This research work is based on very few feature vectors, all together it has included eleven features. Biologically these feature set could not be sufficient to determine the actual stroke condition. It has been advised that this model could solve the real time problem if those more correlated feature vectors is available in training data set.

Relatively, 42K dataset is not enough for the real time problems using machine learning model. Future research with high volume of dataset lead high accuracy even.

Looking into the current data ratio only 2% fall under stroke patient case. This thesis work has not addressed any solution to mitigate the imbalanced data. So, machine learning model could achieve more if the work can stir on fact.

This research is done on single BLSTM with 128 timestamps. Next research could be done on stacked BLSTM networks.

References

1. [M. S. Singh and P. Choudhary, "Stroke prediction using artificial intelligence," *2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON)*, Bangkok, 2017, pp. 158-161]

2. [Khosla, Aditya & Cao, Yu & Lin, Cliff & Chiu, Hsu-Kuang & Hu, Junling & Lee, Honglak. (2010). An integrated machine learning approach to stroke prediction. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 183-192. 10.1145/1835804.1835830]

3. [C. Hung, W. Chen, P. Lai, C. Lin and C. Lee, "Comparing deep neural network and other machine learning algorithms for stroke prediction in a large-scale population-based electronic medical claims database," *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Seogwipo, 2017, pp. 3110-3113. doi: 10.1109/EMBC.2017.8037515]

4. [Sudha, A. & Gayathri, P. & Jaisankar, N. (2012). Effective Analysis and Predictive Model of Stroke Disease using Classification Methods. *International Journal of Computer Applications*. 43. 26-31. 10.5120/6172-8599.]

5. [T. Kansadub, S. Thammaboosadee, S. Kiattisin, and C. Jalayondeja, "Classification of brain cancer using artificial neural network," *The 2015 Biomedical Engineering International Conference* (BMEiCON2015), 2015]

6. [S. Kumar, L. Hussain, S. Banarjee and M. Reza, "Energy Load Forecasting using Deep Learning Approach-LSTM and GRU in Spark Cluster," *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT), Kolkata, 2018*, pp. 1-4. doi: 10.1109/EAIT.2018.8470406]

7. [Zhiyong Cui, et al., "Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction" *International Workshop on Urban Computing (UrbComp) 2017*, Held in conjunction with the ACM SIGKDD 2017]

8.[Ruder. S, "An overview of gradient descent optimization algorithms," arXiv:1609.04747v2 [cs.LG], p. 14, Jun 2017]

9. [Kingma, D.P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. CoRR, abs/1412.6980.]

10. [Timothy Dozat(2015), Incorporating Nesterov Momentum into Adam]

Bibliography

1. https://www.medicalnewstoday.com/articles/7624.php
2. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
3. https://jhui.github.io/2017/03/15/RNN-LSTM-GRU/
4. https://www.kaggle.com/asaumya/healthcare-dataset-stroke-data
5. https://tensorflow.org/
6. https://keras.io/
7. http://mlwiki.org/index.php/ROC_Analysis