Intracranial Hemorrhage Detection using CNN-LSTM Fusion Model

by

Kazi sabab Ahmed
18101509
Khandaker Sadab Shariar
18101306
Naimul Hasan Naim
18301192
MD. Nayimur Rahman Hazari
18101667

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
May 2022

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.

2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.

3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.

4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Kazi sabab Ahmed 18101509	Khandaker Sadab Shariar 18101306
Naimul Hasan Naim 18301192	MD. Nayimur Rahman Hazari 18101667

Approval

The thesis titled "Intracranial Hemorrhage Detection using CNN-LSTM Fusion Model" submitted by

1. Kazi sabab Ahmed (18101509)
2. Khandaker Sadab Shariar (18101306)
3. Naimul Hasan Naim (18301192)
4. MD. Nayimur Rahman Hazari (18101667)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 29, 2022.

Examining Committee:

Supervisor:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Syed Zamil Hasan Shoumo
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

____

Sadia Hamid Kazi
Chairperson
Department of Computer Science and Engineering
Brac University

Ethics Statement

This is hereby declaring that this thesis is based on the results we obtained from our work. Due acknowledgment has been made in the text to all other material used. This thesis, neither in whole nor in part, has been previously submitted by anyone to any other university or institute for the award of any degree.

Abstract

Intracranial Hemorrhage is a term used to describe bleeding between the brain tissue and the skull or within the brain tissue itself. It is life-threatening and needs immediate medical attention. As the first response, it is indispensable to detect the type of intracranial hemorrhage as soon as possible. Now, the manual detection methods require the help of an imaging expert and are certainly very time-consuming. Although there are several techniques for identifying them such as utilizing CT-scan images, magnetic resonance imaging (MRI), magnetic resonance angiogram (MRA), and ultrasound-based images, the results are still not adequate and have much room for improvement. In addition to these methods, researchers have also used imaging strategies based on Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) for this purpose. Therefore, this research aims to combine both these two fields and propose a model based on Deep Learning(DL) to detect intracranial hemorrhage. The goal of this paper is to automate the detection of intracranial hemorrhage and make the process more efficient and accurate. The model is expected to provide us with satisfactory results and can be used as an effective alternative to the existing methods.

Keywords: Deep Learning; Magnetic Resonance Imaging (MRI); Magnetic Resonance Angiogram (MRA); Intracranial Hemorrhage; Recurrent Neural Network (RNN); Convolutional Neural Network (CNN);

Acknowledgement

First and foremost, we give thanks to Allah for allowing us to finish our thesis without any serious setbacks. Second, we would like to express our gratitude to our supervisor, Dr. Md. Golam Rabiul Alam, and our co-supervisor, Syed Zamil Hasan Shoumo, for their invaluable assistance and guidance. They assisted us anytime we required assistance. Finally, without our parents' unwavering support, it may not be conceivable. We are currently on the verge of graduating thanks to their kind assistance and prayers.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$CBAS$	Confidence-guided Brain Anatomy Segmentation
CNN	Convolutional Neural Network
CT	Computed tomography
DL	Deep learning
DRL	Deep reinforcement learning
$DRL - LNS$	deep reinforcement learning-based lymph node segmentation
GAN	Generative Adversarial Network
ICH	Intracranial Hemorrhage
$LSTM$	Long Short Term Memory Network
ML	Machine learning
MRA	Magnetic Resonance Angiogram
MRI	Magnetic Resonance Imaging
PHH	Post Hemorrhagic Hydrocephalus
$RECIST$	Response Evaluation Cri-teria in Solid Tumors
RNN	Recurrent Neural Network

Chapter 1

Introduction

Since the brain cannot hold oxygen, it must rely on blood arteries to provide it. Whenever a hemorrhage occurs in the brain, oxygen starts getting blocked from reaching the brain tissues. Blood pooling in the brain due to an intracranial hemorrhage exerts pressure on the arteries, and brain cells start to die if they are deprived of oxygen for more than three to four minutes. This damage might result in severe mental/physical task-based disabilities such as confusion, dizziness, seizures, abnormality, loss of vision, slurred speech, coma, etc. The consequences and severity of a brain hemorrhage are determined by the origin, location of the bleed, and the amount of bleeding. Additionally, depending on its criticality it can easily turn into a life-threatening situation[1].

Intracranial hemorrhage (ICH) is mainly of two different types depending on the bleeding area extra-axial hemorrhage includes Epidural, Subdural, and Subarachnoid hemorrhage. This brain bleeding occurs within the skull but outside the brain tissue means it occupies the space between the bony skull and dura mater or arachnoid space. However, when the bleeding occurs inside the brain tissues like Lobe, Thalamus, Pons, and Cerebellum are known as Intracerebral, Intraventricular hemorrhage, or intra-axial hemorrhage.

In addition, Brain bleeds have a wide range of causes but common symptoms are hypertension, trauma, atherosclerosis, ruptured cerebral aneurysm, the presence of amyloid protein in the arterial walls or leakage into arteries, and many more. In the journal, JAMA one study shows that ICH is the deadliest stroke, which is also the most common cause of disability. Also, in a survey by Statista [2], worldwide in 2017-2018, the number of people who have died only as a result of an intracerebral hemorrhage was 2.97millions, among them, 1.62millons are men and 1.62millons are women. Similarly, in Bangladesh three months survey of a tertiary care hospital [3], they've taken during the research period, 430 individuals with clinical features of stroke were admitted through emergency and outpatient departments, with 152 (35.34%) of them having a spontaneous intracerebral hemorrhage.

Currently, the mortality of brain stroke in Western industrialized countries is significantly greater than in South Asian countries. However, most studies indicate that the death rate from ICH is greater in those aged 45 and more, but that's not always true. In a study [4], 200 patients were chosen (age range between 15 to 40).

Tobacco usage (20%), hypocholesterolemia (35%), hypertension (13%), and alcohol consumption (10%) were the most common risk factors. ICH was found in the lobar (55%) and basal ganglia/internal capsule (22%) areas, among other places (24%).

In an article by YaleNews [5], a survey was featured that, worldwide individual patients found that those who have had surgery had a statistically significantly higher probability of living after three, six, and twelve months than those who have not. Apart from ICH, an abnormality that can be seen in premature neonates is called post-hemorrhagic hydrocephalus (PHH). Early detection of severe PHH might avoid brain damage and improve the neonate's health significantly.

1.1 Health Consequences and Effective Use of Deep Learning for Brain Hemorrhage

Intracerebral hemorrhage (ICH), or brain hemorrhage, is a life-threatening disease. For treatment, we need immediate and accurate detection of the hemorrhage size, location, and classification. According to [6], The occurrence of the ICH worldwide is 246 people among one million people every year while four hundred thousand to sixty-seven thousand cases are reported every year in the USA alone. In low-to-middle-income nations, however, the number of ICH patients is twice as high as in high-income ones. ICH usually occurs among people with head injuries, concussions, alcohol intake, low cholesterol levels, anticoagulation, drug abuse issues, etc. As unfortunate as it is, the number of people with this kind of issue is beginning to increase, which dramatically increases the number of people with ICH. Moreover, the mortality rate within a month ranges from 35% to 52%. Furthermore, only 20 out of every 100 survivors are predicted to make a fully issueless recovery within six months. To add to our concern, half of this death are predicted to occur within the first 24 hours, emphasizing the critical need for early and correct discovery for successful treatment.

The ICH can be detected via computed tomography (CT) scan, magnetic resonance imaging (MRI), ultrasound pictures, or magnetic resonance angiography (MRA). The location, extent, and possibly the reason for the bleed are determined by these imaging studies [6]. The ICH can be detected via computed tomography (CT) scan, magnetic resonance imaging (MRI), ultrasound pictures, or magnetic resonance angiography (MRA). The location, extent, and possibly the reason for the bleed are determined by these imaging studies [6]. However, CT-scan images are preferred more for ICH detection due to faster detection and images providing pictures of tissues, organs, and skeletal structures.

Head CT scan images are utilized to diagnose neurologic deficiencies all around the world for emergencies[7]. There are two ways for ICH detection. When it comes to manual detection, the help of a highly trained expert is required to analyze these scans, and in some cases, a highly trained expert may make errors in identifying critical components. A unique challenge for head CT is to detect often small subtle abnormalities on a multislice cross-sectional (three-dimensional [3D]) imaging modality that is characterized by poor soft-tissue contrast, low signal-to-noise using current low radiation-dose protocols, and a high incidence of artifacts with perfect or near-perfect sensitivity and very high specificity. Moreover, the detection process may take a lot of time. This is why an optimal automatic detection method is required.

Usually, the separation of the skull and facial image from CT-scan pictures through a variety of image processing techniques is the initial step in bleeding detection[7] .Thresholding is one of these procedures, which involves identifying the skull and face bones before performing a sequence of close, open, and fill operations to gather only the brain's intracranial structure.

After the collection of data, different machine learning(ML) and artificial intelligence(AI) architectures are used for the detection of hemorrhage [7]. The objective of using ML and AI is to have a system that can reduce the detection time of hemorrhage detection without the help of an expert. This process of training a machine comes in a form of supervised, unsupervised, and reinforced learning.

Whether we train a machine using supervised, unsupervised, or deep learning we need to make sure the machine can give us a prediction with considerable accuracy. In recent years, many ML algorithms such as Long Short Term Memory Network(LSTM), Convolutional Neural Network (CNN), Recurrent Neural Network(RNN), Generative Adversarial Network (GAN), etc have been used for the detection of a brain hemorrhage. However, CNN is considered the most efficient among them when it comes to the detection of brain hemorrhage according to.[7] CNN can produce 0.997 accuracies and with the help of high-end devices, the task of hemorrhage detection can be assured.

After the detection, the data from the image of a hemorrhage-affected brain needs to be separated for effective segmentation. Previously, semi-automatic segmentation methods were used for the segmentation process[8]. Although this method greatly reduced process time compared to the manual segmentation process, it still needed the help of an expert and still needed time. But at present, there are many fully automated segmentation methods. These methods include quantitative and qualitative comparison using Unet, Unet6GAN, SegNet, semi-d-Unet using probability maps, graph cuts using CNN, data augmentation along with data balancing using CNN-LSTM network, and other deep learning and reinforced learning methods [9]. However, in most cases, the accuracy of the segmentation is at most 0.90-0.94, and none of the methods includes any significant recurrent neural network (RNN) methods.

As the cornerstone of artificial intelligence, machine learning is also the fundamental cause of computer intelligence, and it is frequently used in this field. Deep learning(DL) has risen to prominence in the field of artificial intelligence as computers' ability to analyze data is improving. The theory and applied research of DL is attracting an increasing number of researchers in recent years. Furthermore, picture identification and categorization are critical applications. The study compares deep learning to traditional machine learning methods, then goes on to describe the deep learning development process, investigate and analyze deep learning network structures such as deep belief networks, convolutional neural networks, and recursive neural networks, discusses the usage of deep learning in image identification, and recommends deep learning in image separation and classification[10]. The problems that emerge while using recognition and classification are discussed, as well as the solutions to these problems. Finally, prospective research prospects and the current state of in-depth learning research in picture identification and categorization are explored. This paper's information is critical to our investigation.

1.2 Objective

The goal of this research is to understand the effectiveness of DL in medical imaging. Our goal is to propose a method combining CNN and RNN which can give us optimal accuracy. The acquired accuracy while compared to other detection methods will give us a clear idea about its effectiveness in Hemorrhage detection. The core objectives of our research are:

1. Understand hemorrhagic detection techniques.

2. To design a model for hemorrhage detection based on the combination of CNN-LSTM methods and apply it.

3. To reduce the time consumption of the hemorrhage detection process.

4. To evaluate the model.

5.Bring improvement in automatic detection of hemorrhage with DL.

6. To propose significant ideas for improving the model.

Chapter 2

Related Work

Neurons, which are vital for our nervous system and responsible for transmitting messages to our functionary organs, are found in large numbers in our brain. As a result, if these cells begin to die for whatever cause, our neurological system will be harmed. As a result, many of our functionary organs may stop working properly and in a worst-case scenario, this may even lead to death.

As stated previously, a person's brain can face injury or be stroked for a series of unwanted reasons. This can result in intracranial bleeding, which stops the flow of blood in that area of the brain. As a result, neuron cells start dying. We need urgent and effective therapy to put a stop to this. But for this, we need to detect the area of the hemorrhage quickly and accurately. To solve this problem many automatic systems containing DL and RL have been used. But the use of a method containing DL to achieve a perfect result has yet to be seen.

2.1 Deep Learning

DL is a sector of ML that uses both CNN and RNN to train a system with unstructured data. Neural networks have been utilized for many AI breakthroughs in problems such as computer vision, machine translation, and time series prediction, language processing. DL is a Kind of machine learning that is very specialized for these fields. A machine learning workflow begins with the manual extraction of essential characteristics from pictures. The characteristics are then utilized to build a classification model for the items in the image. Relevant characteristics are automatically retrieved from photos using a deep learning approach. Furthermore, deep learning accomplishes "end-to-end learning," in which a network is given raw data and a goal to complete, such as classification, and it automatically learns how to do so. In recent years, DL has seen various successes in the field of AI and ML. Most of this success has been seen in robotics. However, its usage in the medical sector is seen very often.

2.1.1 DL Architecture

According to [9], a D-Unet architecture has been proposed. We will follow the same architecture for the most part. But instead of training the data with D-Unet, we shall use DL for training the data. In every layer of the image processing path, two padded convolutional layers both followed by a max-pooling layer and a rectifier linear unit (ReLu) activation are utilized. To gather additional data from the images, the feature extraction ratio is doubled for every layer. These layers include skull stripping, tilt correction, smoothing at different scales, non-linear mirror, asymmetry map at different scales, and asymmetry map at different spaces to generate a probability map.

2.2 Existing Works

The aim of this part of our research is to study the previous relevant works which are related to the field of hemorrhage detection and segmentation including various automatic and semi-automatic methods. This part contains an analysis of different methods and techniques proposed in various significant research on hemorrhage detection and segmentation. Here, techniques and results of different methods have been studied to understand their challenges and effectiveness in solving the problem.

The authors of the paper[11], have used a unique joint segmentation technique to separate ischemia and hemorrhagic infarcts concurrently to evaluate post-treatment CIV more effectively. Deep learning and convex optimization techniques are used in the proposed methodology. Convolutional neural networks are used to combine acquired semantic information, local picture context, and a high-level user initialized prior into a multi-region time-implicit contour evolution scheme that may be improved globally through convex relaxation. The U-Net architecture (D-Unet) incorporates a well-known handmade feature, bilateral Density Difference between symmetric brain regions, to handle this difficult segmentation issue while using a Euclidean distance weighted loss function. To create final segmentation, the CNN learning information, local picture context, and expert input prior knowledge are combined in a convex optimization-based multi-region contour evolution. They obtained 0.934 accuracies for the two regions with favorable values after implementing the suggested Semi-D-Unet approach and comparing them to manually segmented pictures. Since we want to utilize the processing ability of the CNN in our architecture, we believe the training methods included in this paper can be very useful for our research.

The purpose of this report[8] is to segment the focal lesions of the brain while in the chronic stage by using high-resolution T1-weighted magnetic resonance (MR) images. While manual segmentation is possible with the help of an expert, which is extremely time reliant and unvarying and barely produces 90% accurate results. A lesion is segmented by distinguishing it from various surrounding compartments. With slight adjustments, the region-growing approach using pixel aggregation is used for segmentation. Inside each brain lesion, the starting point for the segmentation method was carefully selected. This method helps the segmentation to be faster than traditional methods. However, the current methods provided by this research are no longer useful compared to the current methods and do not provide us with a very accurate segmentation. To conclude, the methods used in this paper are outdated but it helps us understand different criteria for brain lesions segmentation through MR images.

In the paper[12],the authors proposed a deep learning system relying on U-nets for detecting and segmenting hemorrhagic strokes in CT-scan images of the brain. The proposed model incorporates symmetry constraints of the brain images by concatenating the flipped image with the original CT slice as the network's input. They also showed in their paper that the model is trained and tested on two different datasets, resulting in a competitive performance with human experts in terms of detecting location accuracy. The authors have also described in their paper, that U-Net is a type of CNN architecture, and it's been used to solve a variety of biological image segmentation issues with great success. Moreover, in this paperwork, the authors have described that adversarial training is also used in the proposed UNet model to improve the accuracy of the segmentation. Based on CT brain scans, comparison studies revealed that the suggested UNet-based model outperforms human experts in bleeding lesion diagnosis. Overall, the documentation is very detailed and informative. So, it can be concluded that the paperwork is instructive and beneficial for research.

The purpose of this research [13], is to detect and classify intracranial hemorrhage using a proposed model. The authors proposed Deep CNN and LSTM layers make up the suggested model. The weighted multi-label logarithmic loss based on their selected test dataset was 0.07528, which is roughly comparable to a classification accuracy of 92 to 93 percent based on the suggested model trained on balanced data. The resultant weighted multi-label logarithmic loss based on the test dataset, on the other hand, was 0.0813, which is roughly similar to a classification accuracy of 88 to 89 percent with imbalanced data. The researchers have produced a significant result that utilizes the combination of two different methods. However, this model was trained on a large number of datasets which can be a technical challenge. This research can be useful for training datasets.

The authors present a technique for detecting intracranial hemorrhage (ICH) in three-dimensional (3D) non-contrast computed tomography in their publication [14], (CT). They also demonstrated how to train a neural network from start to finish using only one GPU and construct high-resolution 3D NCCTs. For each imaging modality, the technique uses a 3D CNN and LSTM route, which are then combined to generate an image-level classification. The network can also predict the binary output classification of a single 3D picture, according to the authors. Moreover, they also mentioned in their paperwork that they use CNN, and in their paper, they show some schematic overview of the network architecture. In their papers, they attempted to provide a quick and accurate approach for 3D picture classification. Overall the paperwork is appreciated. But this project work needs more specifications. Although I believe that the paperwork is informative and useful for our thesis work.

The author, in this paper [15] expressed a process for anatomic segmentation using ultrasound by utilizing the algorithms of deep learning. Firstly, they showed how traditional CNN architectures fail due to enough positional information to localize. They also suggested utilizing a conditional generative adversarial network 7 (cGAN) to handle picture-to-image translation and image synthesis problems, in which the network is taught to understand the mapping between two domains. The Confidence-guided Brain Anatomy Segmentation (CBAS) network is a unique approach for estimating segmentation and related confidence maps of various sizes. In their research, they took 1629 images for their testing from 20 various subjects and got an accuracy of 89%.

The author of this paper [16], proposed an automatic method to prevent this severe brain injury. The main objective of this research is the prediction of the PHH outcome earlier so that the requirement of surgical intervention is known. The proposed methods consist of two parts. First of all, the segmenting of ventricles by cranial ultrasound images with the help of CNN and its grounds like fuzzy c means and with weight loss function. Method 1 does most of the tasks like cranial bounding box, detection cranial region initialization, bounding line detection, brain interhemispheric fissure detection, ventricle extraction, and many more after all of these the second method focuses on six parameters like birth weight, sex, head circumference at the time of birth, age in days when IVH was diagnose and IVH grade. This automatic process takes less than a half-minute time to segment and has an accuracy of 0.91. Compare to other CNN-based processes like U-net, SegNet, and De-convenient takes 2.5 minutes to process the segment and gives accuracy between 0.64 to 0.77 shown by thinking of the accuracy and the time this method should in this paper is highly reliable on the way to detect hydrocephalus syndrome.

The authors, in this paper [17] that Alex Net, a convolutional neural network model utilized lately in the biomedical area, can classify computed tomography (CT) pictures of brain hemorrhage very accurately. The convolutional neural network model (CNN), autoencoder structure, and heat map approach were used to increase classification accuracy. They clarify that the goal of the convolution layer is to circulate the entire picture with the chosen filter size. They employ a machine learning algorithm called SVM for regression and classification. Heat maps are a helpful tool that is frequently employed in photographs in the biomedical sector, in the study of biological systems images, and in the analysis of images of landforms. The accuracy they have got in this paper is 98% which is a great achievement however, the categorization findings were harmed by the poor resolution of images.

In this research [18], the authors have proposed a deep reinforcement learning-based lymph node segmentation (DRL-LNS) model. Based on Response Evaluation Criteria in Solid Tumors (RECIST) annotations, segmentation of RECIST-slices was implemented in an unsupervised way to produce pseudo ground truths, which were then used to train U-Net as a segmentation network. Next, a DRL model was trained, which helps the segmentation network to interact with the policy network to optimize the lymph node bounding boxes and segmentation results simultaneously. The proposed method was able to outperform U-Net-based models with a dice similarity coefficient (DSC) of 77.17%. Although this outperforms some well-known models, there is still room for improvement. Overall, this paper gives us a good idea about DRL architecture. Moreover, this paper deals with various image separation methods and some good utilization of image processing techniques, which should come in handy for our research.

In the paper [19], the authors offer a semantic segmentation approach for distinguishing six forms of intracranial hemorrhage and calculating blood loss. They stated in their article that to overcome this problem, they use a pre-trained U-Net model with fine-tuning, with the best final test accuracy reaching 94.1 percent of their work. They also demonstrated that they use a two-step training approach. The backbone must be frozen first, and the other layer must be trained first. The model is then unfrozen and retrained with a lower learning rate for additional fine-tuning. When compared to a model that was built from the ground up, the findings demonstrate that it is more accurate and takes less time to train. Because of the small dataset, this is particularly beneficial for medical images. In general, I appreciate the documentation. However, additional details about this project are necessary. Even though I feel the paperwork is instructive and helpful to our thesis research.

From the data above we can conclude that the majority of studies processed the data acquired from CT-scan images and utilized variously supervised and unsupervised learning techniques for bleeding identification. However, the majority of them are faced with the difficulty of employing high-end gear to generate the outcome in a shorter amount of time. Moreover, unsupervised learning provides great results at the cost of computational complications. In addition, some of the research only focuses on the detection of hemorrhage where without identification of the hemorrhage affected area the treatment is not possible. Moreover, none of the above research has utilized the use of DL to achieve a perfect result in the detection of hemorrhage. So, detection of intracranial hemorrhage utilizing the maximum ability of DL should be done.

Chapter 3

Background Study

First Neural Networks Architecture was proposed by McCulloch and Pitts in 1943.

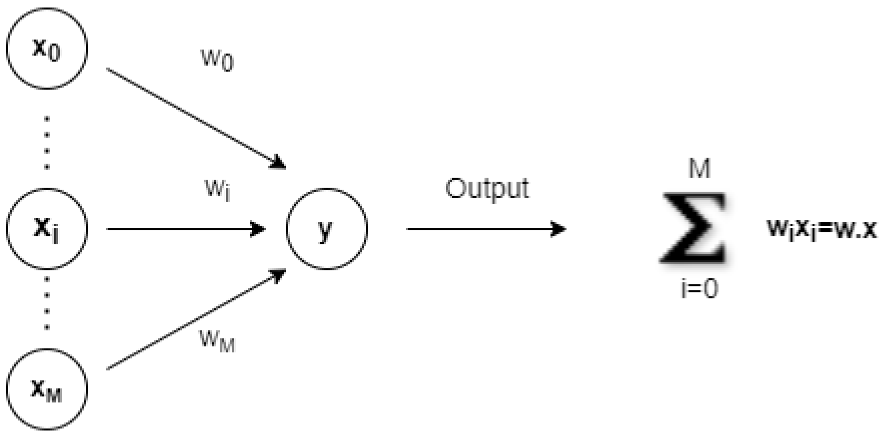


Figure 3.1: The First Neural Network Model

Here, x0 to xM, 0 to M, and y denote the input, weights, output, or activation nodes respectively. For the network shown, in order to activate the function y the conditions are if yin≥ [threshold]f(yin)=1 ; else f(yin)=0. If the activation node gives an output value of 1, that means the node is activated and vice versa.

3.1 Common Activation Functions

In neural networks, the activation function is critical since it determines whether or not a neuron should be stimulated. Activation functions come in two varieties. They are known as linear and non-linear. The linear activation function f(x)=mx+c is a straight-line function. With a gradient decent problem, this activation function operates like linear regression and provides a constant output. Non-linear functions are common in neural networks because they can quickly adjust to different data and outputs. There are various activation functions, but the most prominent is sigmoid, Tanh, and ReLu.

3.2 Sigmoid Activation Function

This non-linear function takes a value as an input and normalizes the value between 0 and 1. If the value is large enough, then the process converts it to 1, and if the value is smaller, the sigmoid function converts it to 0. It can be represented by f(x)=11+e-x

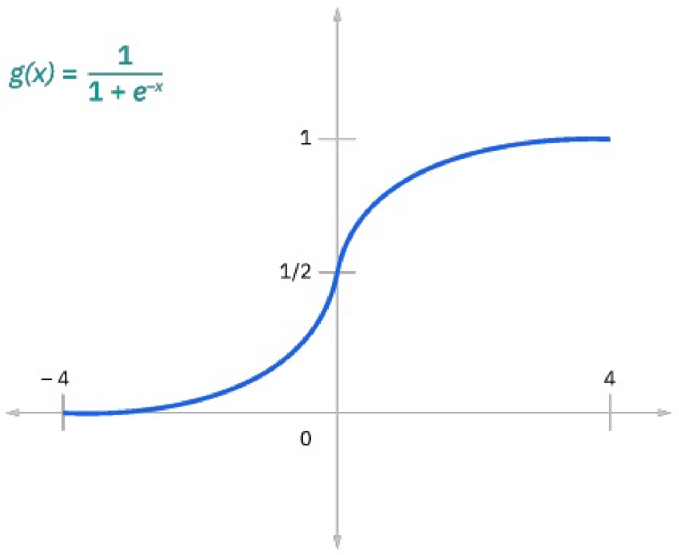


Figure 3.2: Sigmoid Activation Function

Sigmoid functions are straightforward for training small datasets and are commonly used when the output is needed to match the probability, which ranges between 0 and 1. Some of the flaws of sigmoid functions are that the problem of vanishing gradient arises during the backpropagation. Additionally, if the gradient reaches 0, no learning takes place.

3.3 Tanh Activation Function

Unlike the sigmoid function, the Tanh function's output is 0 centered. The sigmoid and Tanh functions share the same diagram, but they range from -1 to +1. Thus it helps the next layer to learn more about the model easily. Tanh function can be represented by, $f(x) = \frac{e^{-x}-e^{-x}}{e^{-x}+e^{-x}}$

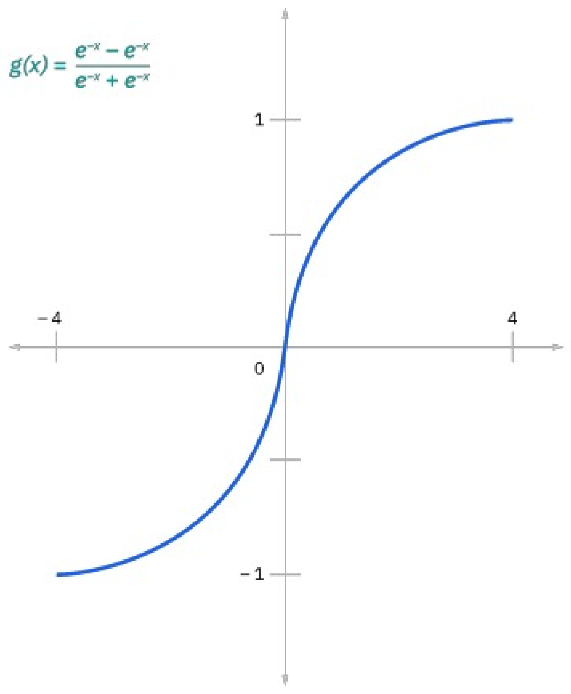


Figure 3.3: Tanh Activation Function

Tanh function also suffers from a vanishing gradient problem, but it moves in both directions as zero centered.

3.4 ReLu Activation Function

The Rectified Linear Unit or ReLu function is more efficient than both Tanh and sigmoid processes because of its derivative process that allows backpropagation in a well-mannered way. ReLu function can be represented by, f(x)= max(0,x)

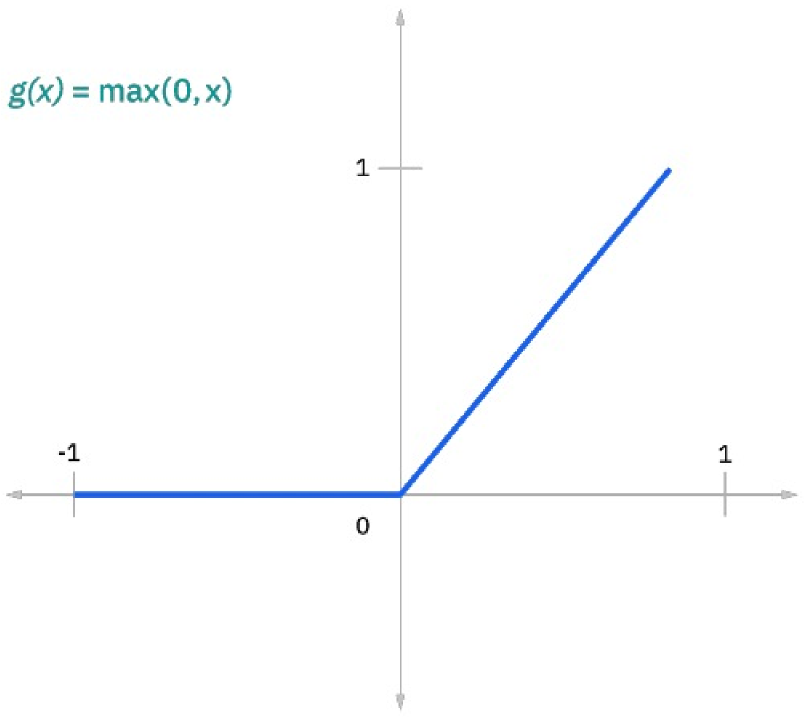


Figure 3.4: ReLu Activation Function

It omits the gradient decent problem by converging to the global minimum of the loss function.

3.5 Convoluted Neural Networks

Convolutional neural networks are extremely effective in image processing. A digital image is made up of pixels, each of which has a value that defines what color and brightness it should be. When we perceive a picture, our brain analyzes a large quantity of information. Each of these neurons is coupled to the others in order to span the full visual field. Each neuron in the visual system responds to stimuli that occur within a certain receptive field; thus, CNN functions in a similar manner. Simpler patterns (lines, curves, and so on) are spotted first, ahead of more complicated patterns, thanks to the layering system (faces, objects, etc.). CNN may be used to offer a vision to computers. A CNN is made up of three layers: 1) the convolution layer 2) the max pool layer, followed by a completely connected layer. A CNN is often defined by its convolution layer. This layer carries the majority of the computational load. The layer works by taking the dot product of two matrices, one of which is called the filter and the other the receptive field. Even though the filter does not take up as much area as the image, it contains more information. There are three sorts of channels in an RGB picture. The filter depth encompasses all three channels, although the breadth and height are modest. The filter processes the image, including its height and breadth, to create a visual representation of that specific receptive field.

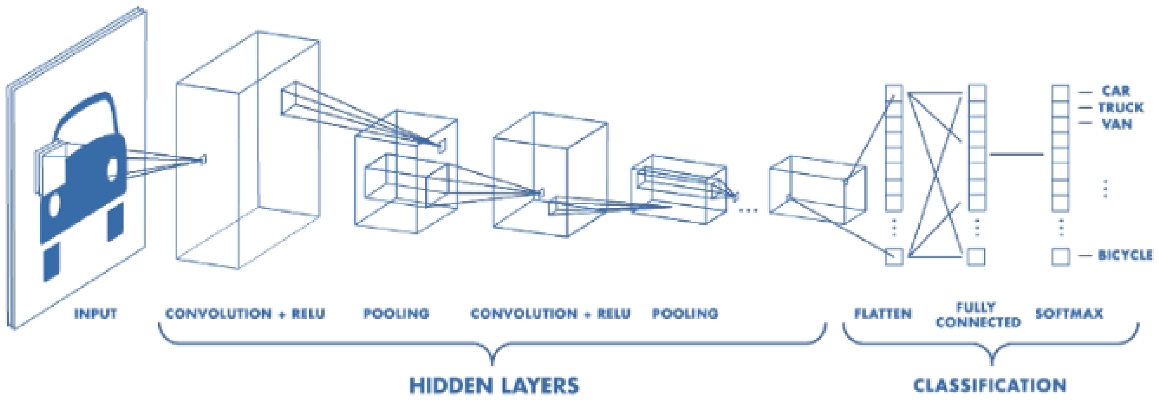


Figure 3.5: An overview of Convoluted Neural Networks

A feature map is a representation of a two-dimensional representation of picture data that includes certain features. The amount by which the filter slips is called astride. The pooling layer is in charge of replacing network output at particular locations chosen by computing the summary statistic of surrounding outputs. The pooling approach processes each slice of the representation individually. Pooling functions include the rectangular neighborhood average, the rectangle neighborhood L2 norm, and a weighted average based on the distance from the center pixel. Max pooling is the most prevalent CNN approach, which takes the maximum of each layer in the feature map. In all circumstances, pooling gives some translation invariance, which implies that an item may be recognized no matter where it appears on the screen. Neurons in the fully connected layer, like those in standard FCNN, are completely interconnected to all neurons in the preceding and subsequent layers. As a result, it may be calculated using matrix multiplication and a bias effect. The FC layer aids in the mapping of input and output representations.

3.6 Recurrent Neural Network

A recurrent Neural Network or RNN is a neural network where the output from the previous step is fed as input of the current stage. This neural network is widely used in the sequential network, speech recognition, natural language processing, sequence classification, etc.

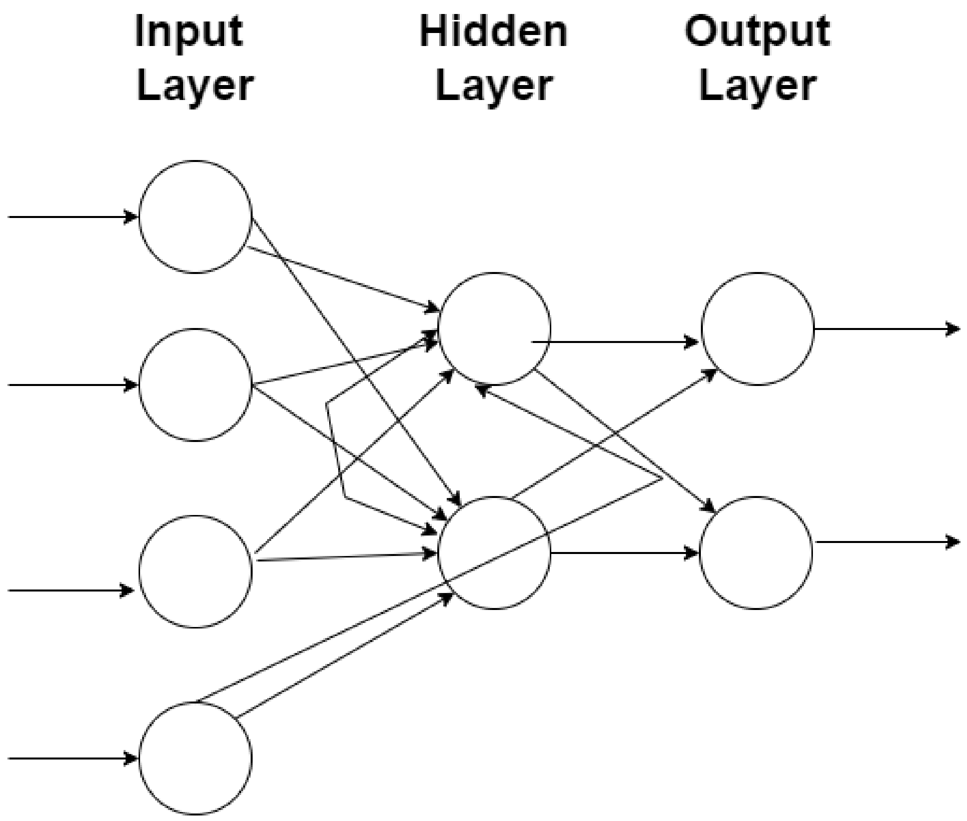


Figure 3.6: RNN Basic Model

The function RNN is f:h',y=f(h,x). Here, h',y,f,h, and x are called a current hypothesis, current output, RNN function, previous hypothesis, and current input respectively. Also, h' and h are vectors with the same dimension. Additionally, RNN reduces complexity as it only needs one function f to perform the feed-forward nature.

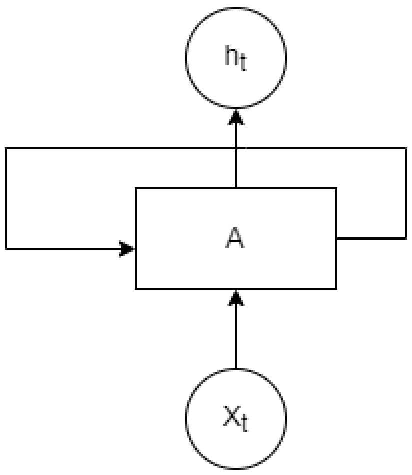


Figure 3.7: RNN Feedforward Architecture

3.7 Types of RNN

There are several types of RNN, but deep RNN, bidirectional RNN, pyramid RNN, and naive RNN are widely used. Deep RNN uses previous function output as a new input of a newly created RNN. Number theory's base conversion can be an example of deep RNN. The function this,

h',y=f1(h,x);
g',z=f2(g,y);
i',m=f3(i,z); ... as follows.

Two opposite directional hidden layers connect bidirectional RNN with the same output. An example of bidirectional RNN would be data sequence and palindrome checking. The function this, y,h=f1(x,h); z,g'=f2(g,x); p=f3(y,z) Naive RNN is that RNN that accepts variable input and variable output in a feedforward manner. The function this, f:h',y=f(h,x)

Deep RNN uses previous function output as a new input of a newly created RNN. Number theory's base conversion can be an example of deep RNN. The function this, h',y=f1(h,x); g',z=f2(g,y); i',m=f3(i,z); ... as follows.

wo opposite directional hidden layers connect bidirectional RNN with the same output. An example of bidirectional RNN would be data sequence and palindrome checking.

The function this, y,h=f1(x,h); z,g'=f2(g,x); p=f3

Naive RNN is that RNN that accepts variable input and variable output in a feedforward manner. The function this, f:h',y=f(h,x)

3.8 Long Short Term Memory

LSTM is the advanced version of RNN which was proposed in the sense of covering the lakes of RNN.Following, like RNN it does not have any vanishing gradient problem, can hold a longer memory and has a cell state. Three sigmoid functions forget gate, output gate, and input gate are in LSTM.

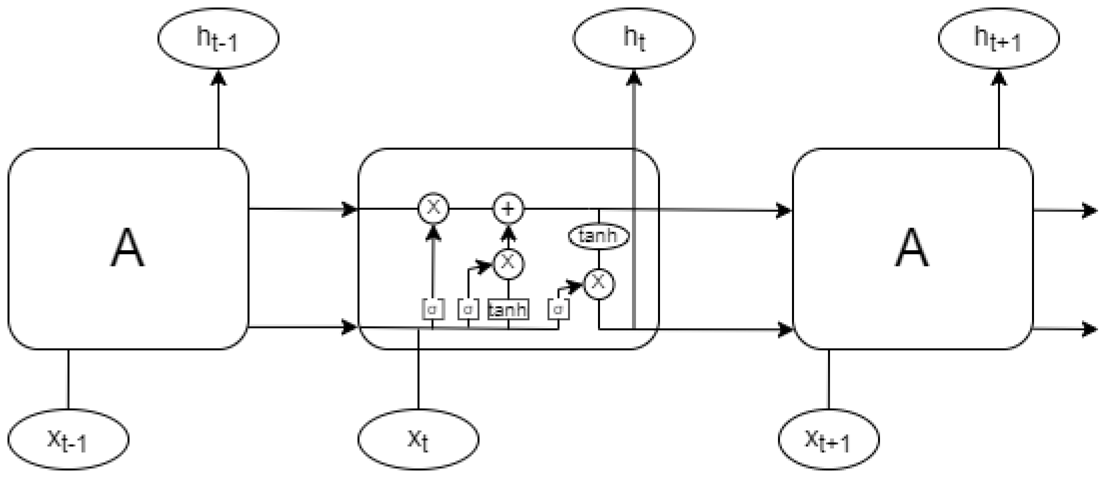


Figure 3.8: LSTM Architecture

Forget gate is the first sigmoid function in the architecture. The information of the current input and previous out put is passed through it. When any value is closer to zero it's forgotten and closer to one means kept. The function is, ft=(Wf*[ht-1,xt]+bf).

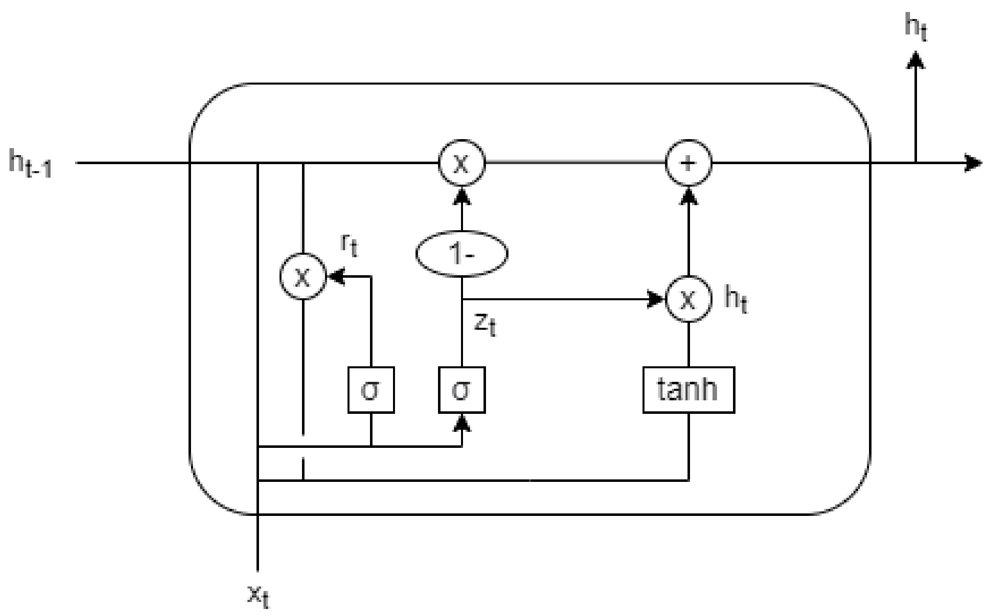
Input gate decides what information should be added in the current step. The function is, it=(Wi*[ht-1,xt]+bi).

Update gate determines how much the past knowledge needs to be passed into the current node. Updating the cell state function is, Ct= ft* Ct-1+ it * C't.

Lastly, the function of output gate is, ot=(Wo*[ht-1,xt]+bo) ; ht=ot*tanh(Ct).

3.9 Gated recurrent units

Gated Recurring Units (GRU) were first proposed by Kyunghyun Cho in 2014. With fewer parameters than LSTM, additionally GRU has a reset gate. The following figure 8 represents the design proposed by Cho along with the functions.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 3.9: GRU Architecture

From Figure 8 we can see GRU latkes Xt as the only input. Unlike LSTM, GRU has two gates. Reset gate and Update gate. Moreover, unlike LSTM, GRU doesn't rely on the previous cell state. GRU is considered more effective than RNN as they explicitly attempt to address the vanishing and exploding gradient problem, which arises during RNN.

Chapter 4

Methodology

We started off our work by looking for a suitable dataset optimized for the detection of brain Hemorrhage. Even though we found quite a few datasets for brain Hemorrhage detection not all of them were suitable for our purpose.

The following subsections serve the purpose of describing our research methods in detail.

4.1 Data Collection

Our first aim was to obtain a dataset of high-resolution CT scan pictures of brains that we could use in our design specification. The image set also needed to be categorized as Hemmorrhaige-type or Normal-type for our research purpose. We did come across quite a few datasets consisting of CT scan images categorized as Hemmorrhaige or non-Hemmorige. However, not all of them had a sufficient amount of data and, not all of them had high-resolution images. After, searching quite a few datasets we finally came across CT - 500. It had both high-resolution CT scan images alongside being categorized as Hemmorrhaige or non-Hemmorige.

CQ500 [19] is a database given by the Center for Advanced Research in Imaging, Neurosciences, and Genomics (CARING) in New Delhi, India. This database is made up of head CT scans obtained by many radiologists in New Delhi's central district. To obtain the pictures, radiology facilities employ tomographs that have anywhere from 16 to 128 incisions. The data was extracted from local PACS servers and anonymized following HIPAA internal rules. The information was gathered in two sections (B1 and B2). Block B1 was constructed by choosing all CT scans performed at the radiological center for 30 days starting November 20, 2017, and Block B2 was compiled from the remaining scans.

The following exclusion criteria were used for each of the studies that were chosen:

• There should be no postoperative problems such as burr holes, shunts, or clips in the patients.

• They should have at least one CT study without axial cut contrast and a soft kernel reconstruction that contemplates the entire brain.

• Patients should not be less than 7 years old. If age information is not available, it will be estimated through bone degradation and cranial sutures.

The dataset[20] is an updated version of the previous dataset and contains 6794 files which include both hemorrhagic and normal brain ct scan images. The following images (figure: 4.1), (figure: 4.2), and (figure: 4.3) show the amount of hemorrhagic and normal brain CT images available.

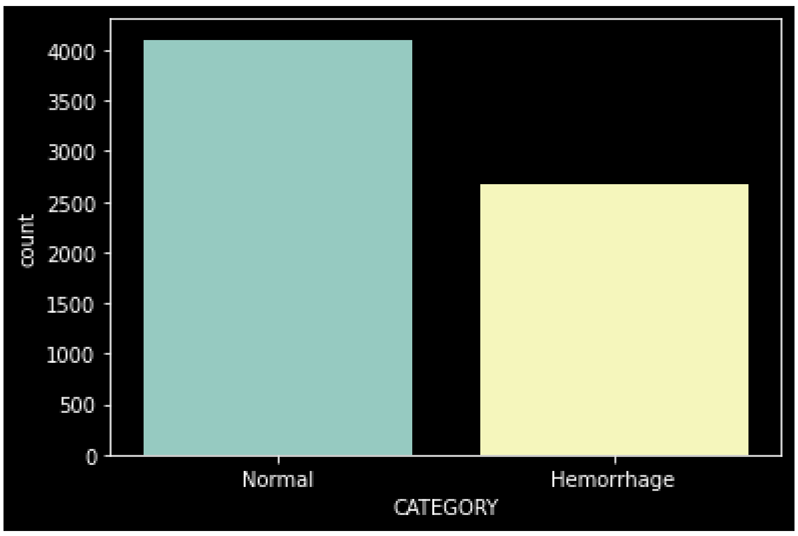


Figure 4.1: Image comparison

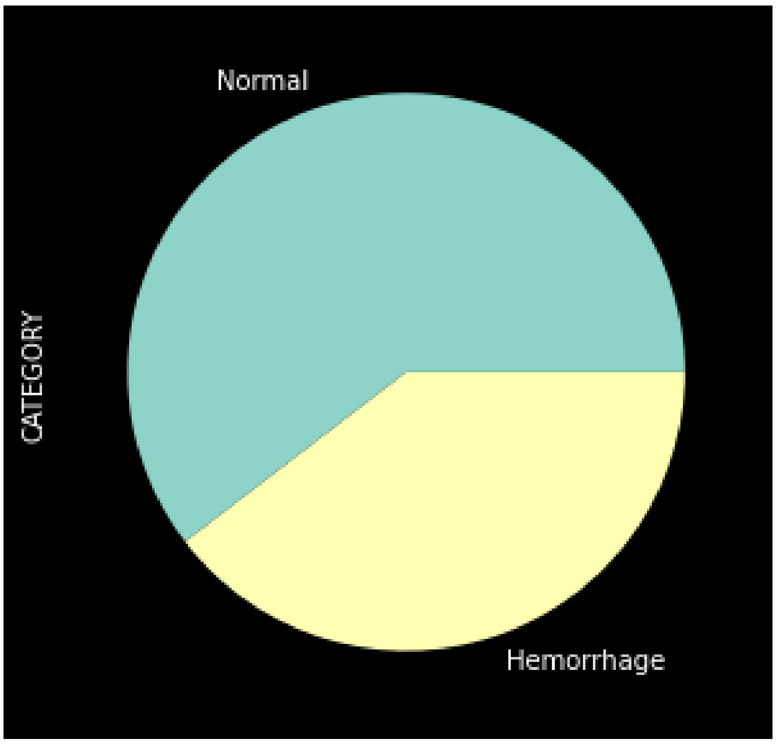


Figure 4.2: Image comparison

The number of Normal-type images is 4105 while the number of Hemorrhage-type is 2690. In other words, a considerable amount of CT scan images of Hemorrhage type are available in the dataset for us to study and research. The following images (figure: 4.4), and (figure: 4.5) show the two types of images in the dataset.

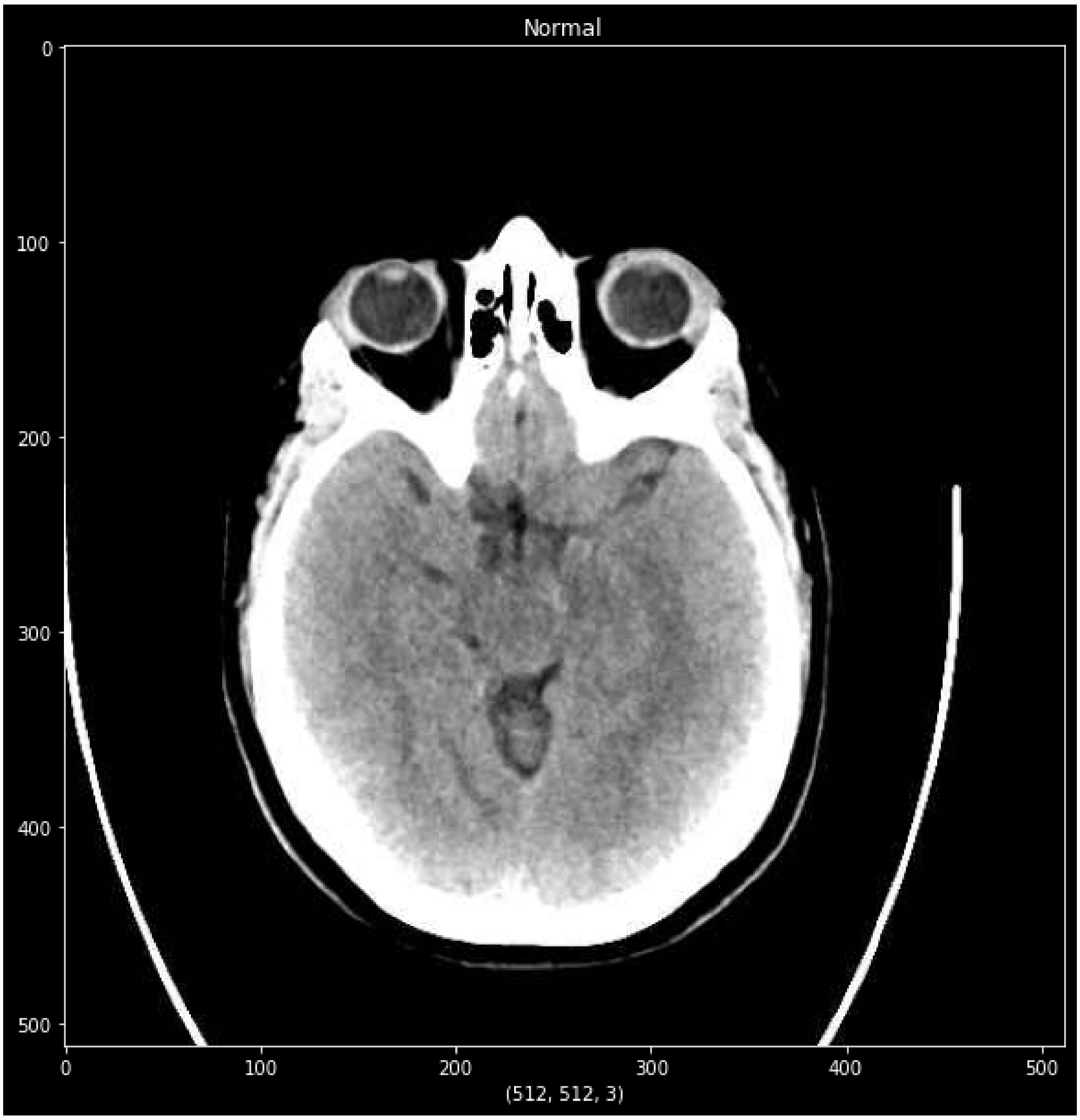


Figure 4.3: Normal-type

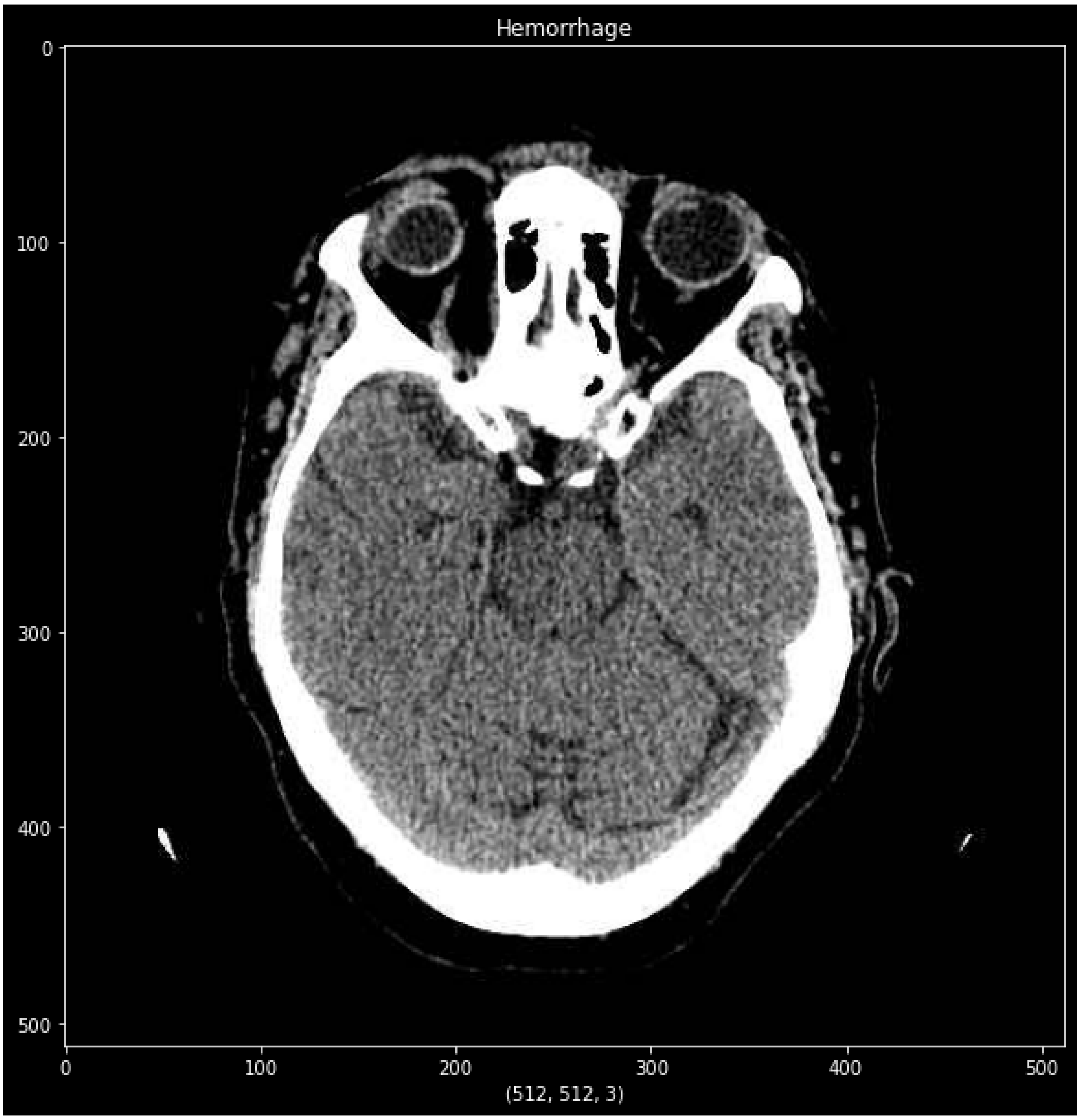


Figure 4.4: Hemorrhage-type

The images of the dataset are categorized as normal type and Hemorrhage type. The following image(figure: 4.5) shows how the dataset is categorized.

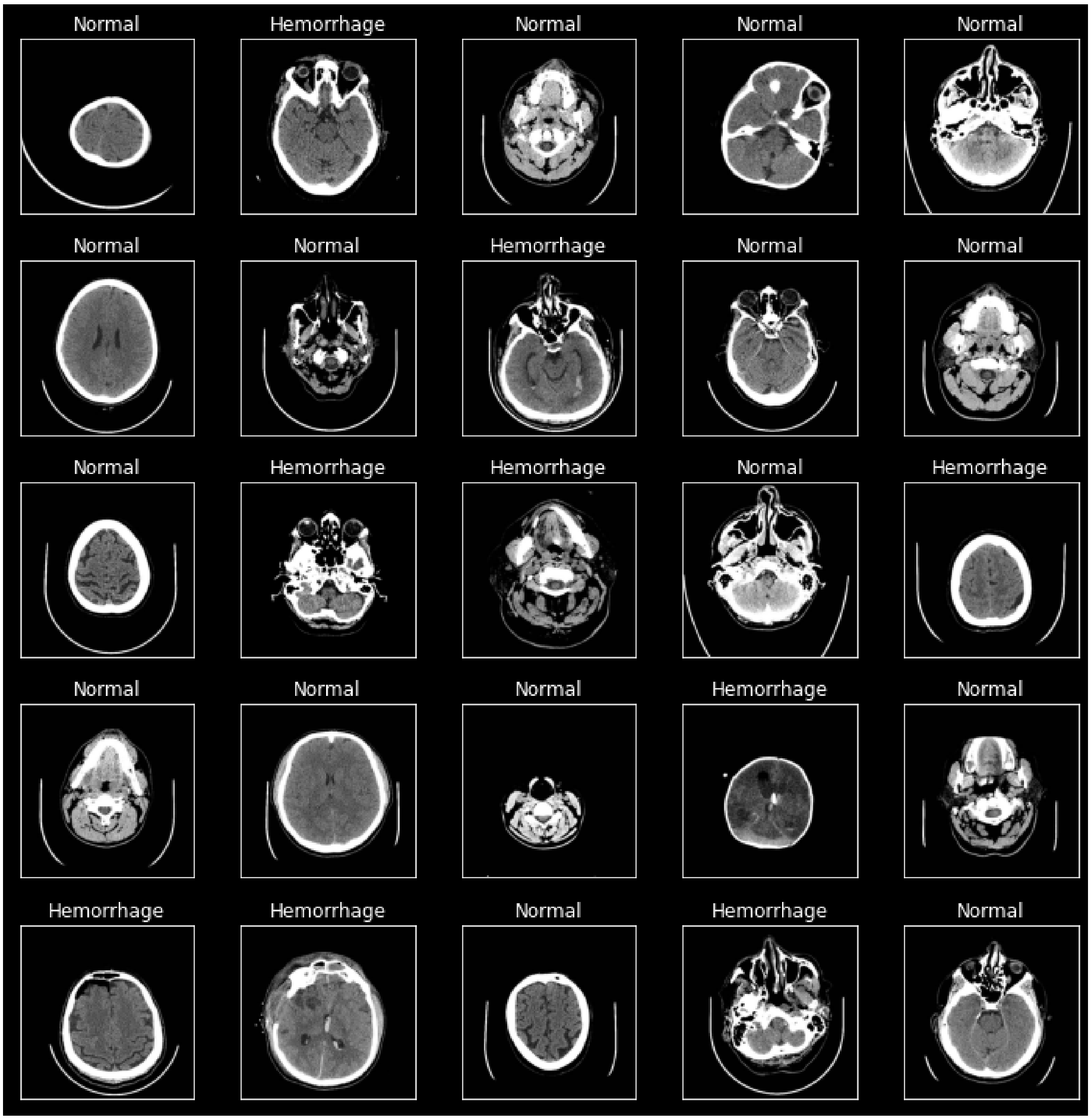


Figure 4.5: Image Catagory

4.2 Data Preprocessing

We have decided to use the panda library to convert the dataset into a table format which will be containing 2 columns and 2 categories (Hemorrhagic and Normal). We have categorized the Hemprrage Type images as 0, and Normal as 1.

We decided to remove the background image of all slices as it will not be of any use for classification and will only increase computational complexity. Moreover, we have converted all the images to JPG format as it is easier for PCs to read images in this format, and this type of image holds less space. This will reduce space complexity and, reduce the load on our device while processing the images.

First, we have fixed the file path for the dataset. Then we are assigning all images in the file path to a list of the dataset using the list method of python.

4.2.1 Data Splitting

Our dataset is made up of only one type of image data which is CT Scan images of the brain while categorized as Normal-type images and Hemorrhage-type images. First, our dataset was split into train and test datasets. We split our dataset into a ratio of 9:1, where 6115 images were selected for training the data and, 680 images were selected for testing the data. Moreover, we Have decided to shuffle the dataset by setting the shuffle parameter to True for training quality. We have also decided to use the same data as random states.

4.2.2 Generating Images

We wanted to avoid the situation where our model faced the overfitting orientation problem. We have used diversification so that our model does not shift to an overfitting orientation.

We rescaled our image as 1./255, used a 20% zoom range, 20% shear range, 40 as rotation range, and, horizontally flipped the image to generate the images that we want to process. We used diversification for the train data only. We did not use the diversification for train data as we can use it as it is and, it will give us a better idea about how much it can be used in real-life data. The following image (figure: 4.7) shows what the generated images look like:

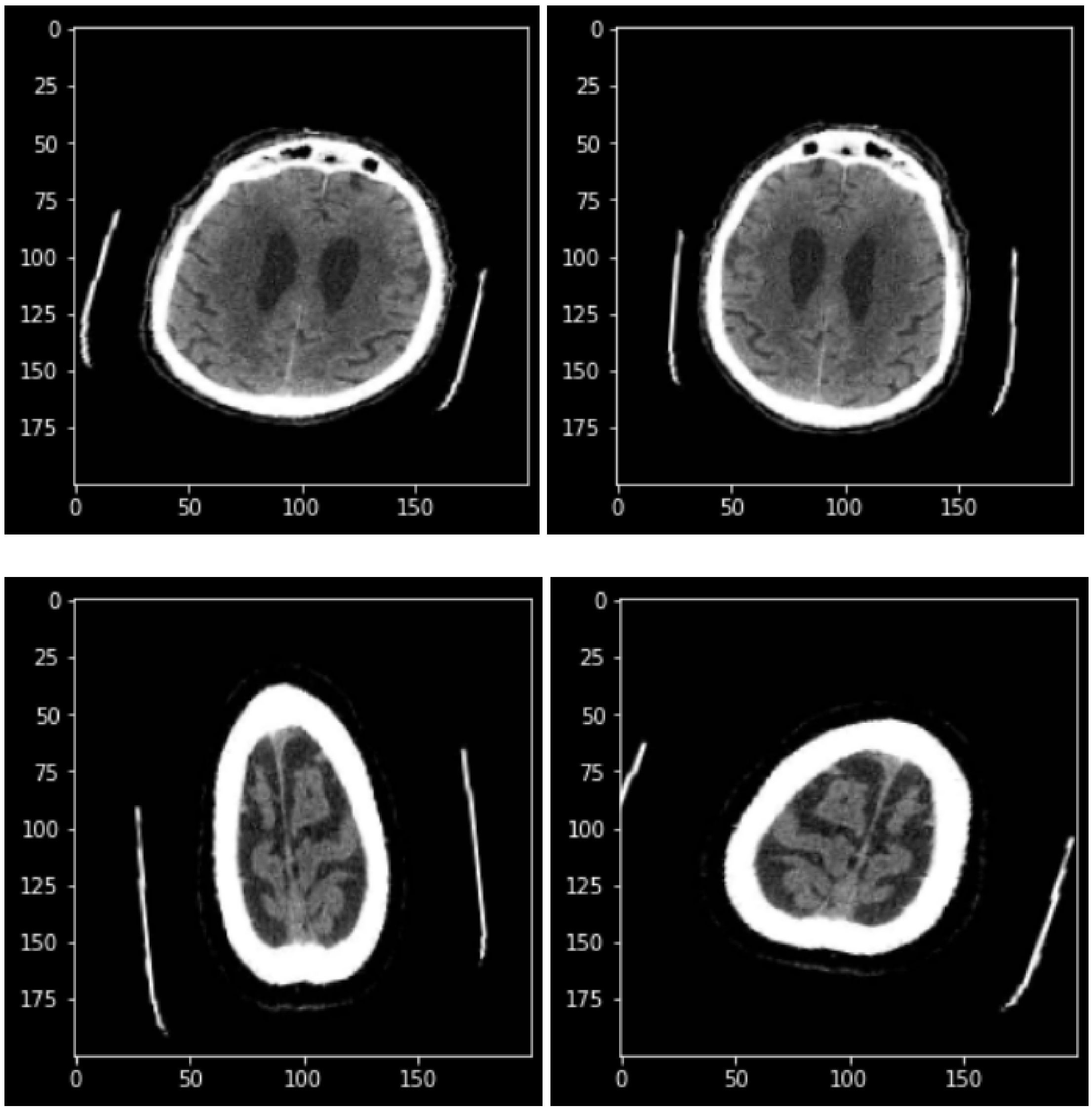


Figure 4.6: Generated Images for processing

4.2.3 Image Preprocessing

After loading our images, we needed to convert the images to pixelated format. Python and Tensorflow functions can not process raw images for computational purposes. We have used TensorFlow functions to decode the image data. In most cases, tensorflow.image.decode image can be used for generic decoding. However, for JPEG decoding, we used tensorflow.image.decode jpeg.Keyword argument is used to change the pixel format of the decoded image. The number of integer values per pixel is represented by the channel parameter. By default, channels are set to 0, causing the decoding function to utilize the raw data interpretation. In our case, we are using the default value. If we set the channel value to 1, we can use a grayscale image; if we set it to 3, we can use an RGB image. We are going to use grayscale value in our case as our images don't need RGB detection. Because we can utilize the precise pixel value in our dataset because the model would get biased toward higher pixel values, we don't normalize the pixel value by decoding. We are using class mode as "categorical": "categorical". This class mode is a 2D NumPy array of one-hot encoded labels. Supports multi-label output. This class mode is more suitable for our Data Frame. The resulting input dataset would be made of one attribute compromising two Categories; Hemorrhagic and Normal and comprising 6794 images with the size of 256*256 pixels.

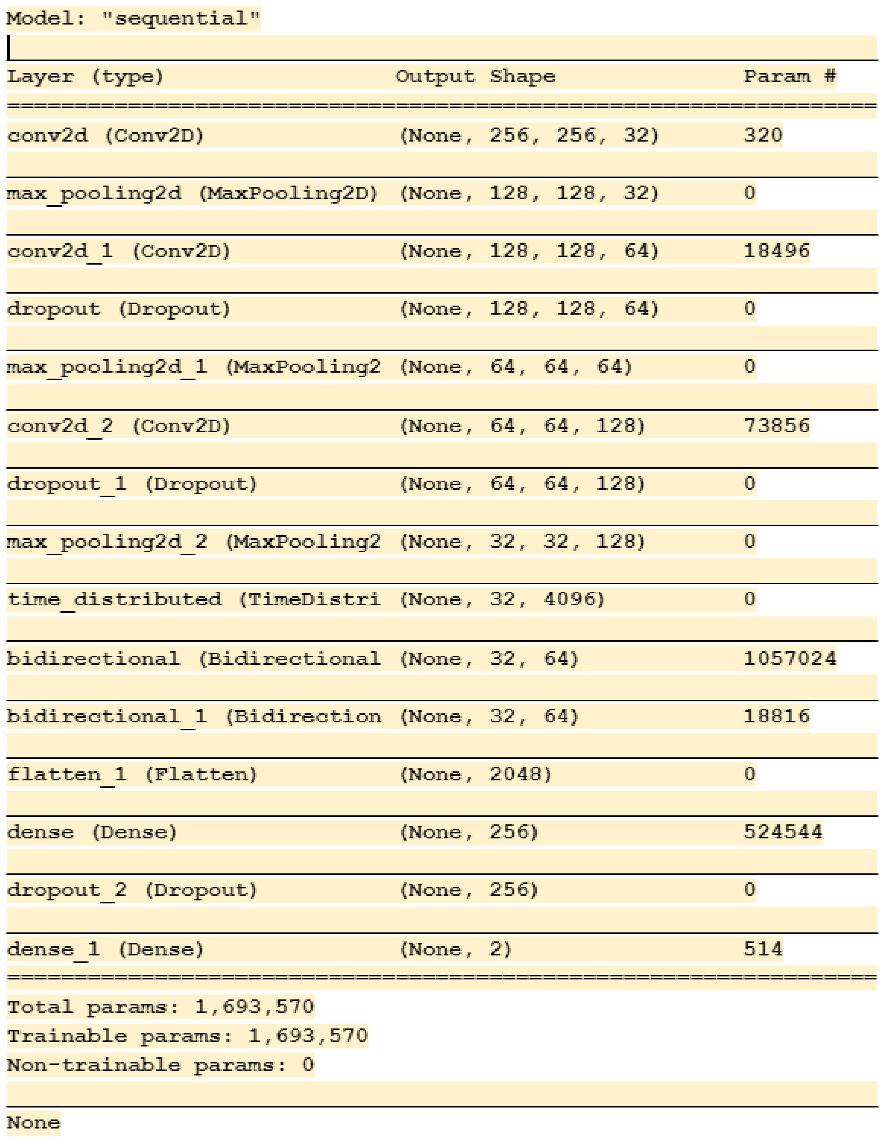
4.2.4 Padding

When an image is processed by the CNN kernel in convolutional neural networks, the padding determines how many pixels are added to it. Padding works by increasing the processing area of a CNN model. A neural network filter called the kernel examines each pixel in a picture and turns the data into a smaller or bigger format. Padding is provided to the picture frame to aid the kernel in processing the image by providing an extra area for the kernel to cover. Padding a picture that has been CNN-processed enables for more exact image interpretation. In the Cov2D layer, if padding is defined as 'valid', it means no padding. On the other hand, if padding is defined as 'Same' it means padding with zeros evenly on the up/down/left/right of the given input data. In our case, we want to take advantage of the Padding function so we defined padding as 'same'.

4.3 Model Implementation

We've spoken about our dataset and how we prepared it up to this point. We will discuss our suggested model and its implementation in this section. In short, our suggested model is a combined model made of CNN and LSTM layers. We are using the previously mentioned CQ-500 model for training our model. We have made the necessary adjustments to run our model in this dataset. The size of the images mentioned in the dataset plays an important role in our model since it determines the form of the Embedding layer and the final output layer.

The following Figure 4.7 shows the summary model.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 256, 256, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 64)      18496
_________________________________________________________________
dropout (Dropout)            (None, 128, 128, 64)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 128)       73856
_________________________________________________________________
dropout_1 (Dropout)          (None, 64, 64, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 32, 32, 128)       0
_________________________________________________________________
time_distributed (TimeDistri (None, 32, 4096)          0
_________________________________________________________________
bidirectional (Bidirectional (None, 32, 64)            1057024
_________________________________________________________________
bidirectional_1 (Bidirection (None, 32, 64)            18816
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 256)               524544
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 514
=================================================================
Total params: 1,693,570
Trainable params: 1,693,570
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 4.7: Model Structure

The following figure 4.8 shows a more detailed architecture of our model.

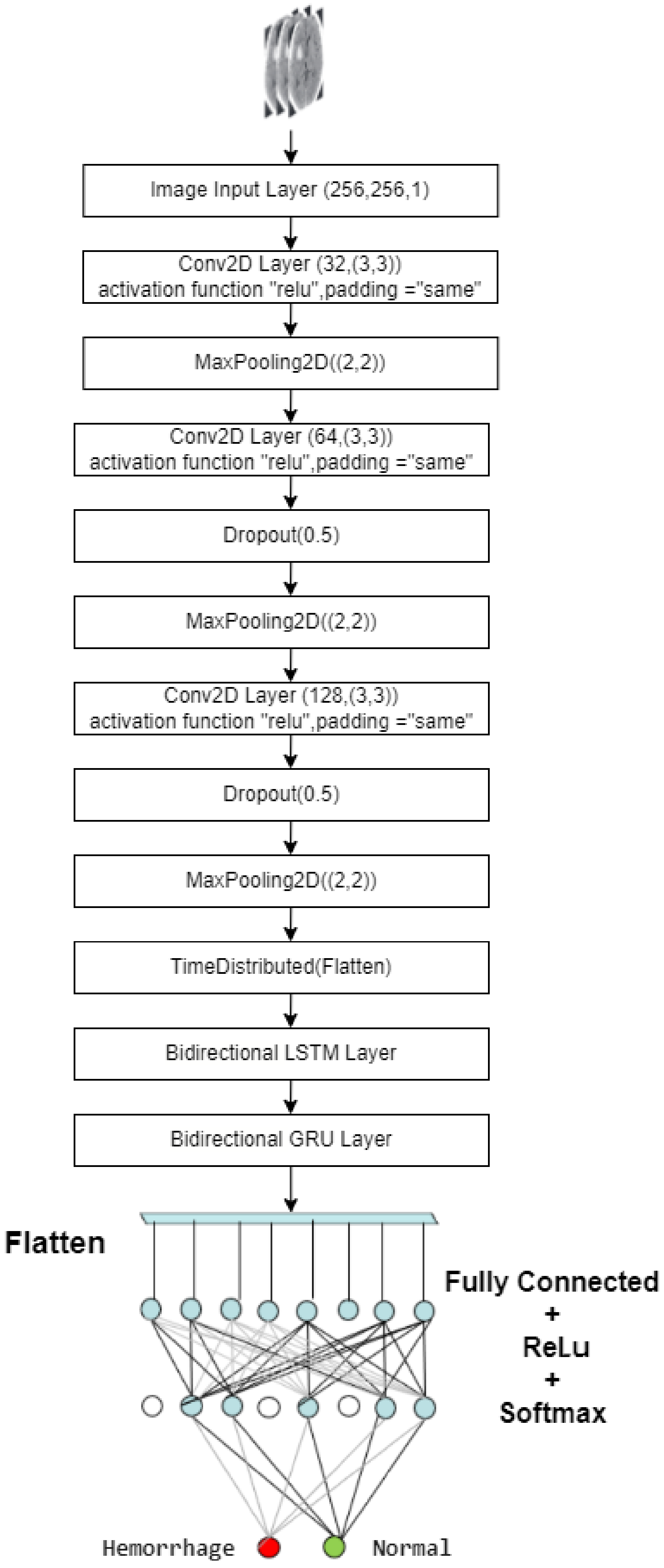


Figure 4.8: A Detailed Architecture of Our Proposed Model

4.3.1 Input Layer

For our image dataset, we have shaped our image dataset and fixed the input dataset as (256, 256, 1). We did not need to make any other changes in our model in order to work with this shape. In our model, each layer is stacked in a plain format and, each layer will take exactly one input tensor and one tensor output. In such a situation, the sequential model is more fitted. So, we have transformed our model into a sequential model by using the keras.Sequential() from the TensorFlow library.

4.3.2 Conv2D Layer

We are using four convolutional layers. We explicitly specified our input shape in the first Conv2D layer so that the CNN architecture may start from there. In our model, Layers closer to the original input data learn fewer convolutional filters, whereas layers closer to the output predictions learn more filters. We raised the number of filters we learned as our output spatial volume dropped. This is a frequent design technique for CNN architectures. When it came to determining the right number of filters, we went with powers of two in all the Conv2D layers. Depending on the complexity of our dataset and the depth of our neural network, we had to fine-tune the precise value. The width and height of the 2D convolution window are specified by a 2-tuple called kernel size. We have specified kernel size as (3*3) for all the available Conv2D layers as they are most commonly used. Moreover, we have used the ReLU activation function for all the layers as ReLU helps to keep the computing required to run the neural network from becoming exponentially. On the other hand, we have tried using Softmax too. But Relu gave us better results overall.

4.3.3 Max_pooling2d Layer

Max pooling takes the highest component from the region of the feature map covered by that of the filter. As a result, the max-pooling layer's outcome is a feature map with the most important characteristics of the previous feature map. We have used the Max pooling layer after all the Conv2D layers. Max pooling is used in our model to reduce the spatial dimensions of the output volume. We have kept both the pool size as 2 and stride as 2 in our model. The usage of Max pooling after the usage of the Conv2D layer is always apparent.

4.3.4 Dropout Layer

The Dropout layer's goal is to ensure that our network generalizes our model and avoids overfitting. This layer's goal is to eliminate the possibility of current-layer neurons arbitrarily disconnecting from neurons in the next layer, requiring the network to rely on existing connections. One disadvantage of LSTMs and Conv2d is that they can easily overfit training data, reducing their ability to forecast. Input and recurrent connections to LSTM and Conv2d units are probabilistically excluded from activation and weight updates when using the Dropout technique to train a network. Overfitting is decreased as a consequence, and model performance is enhanced. During the training of our dataset, we first encountered problems with overfitting. The overfitting problem was handled by adding the Dropout layer after the last two Conv2D layers and after the LSTM layer. We can select a dropout value that is appropriate for us. It's a hyperparameter we need to figure out. The most common dropout value is 0.5-0.9. For our model, we used 0.5.

4.3.5 Time_Distributed Layer

LSTMs are strong, yet they are difficult to use and set up. Before implementing the LSTM layer, we apply the time dispersed layer to reduce the complexity. We have used the Time_Distributed flatten layer to create a model for our many-to-many architecture. Because each of the "many" outputs must have the identical output function applied to each timestep, this is the case. We have used the Time_Distributed Flatten layer to apply the Flatten function to each output across time.

4.3.6 Bidirectional LSTM Layer

We used the Bidirectional LSTM layer after flattening the data in the TimeDistributed layer. We have used only one LSTM layer for our model. The input and out dimension of this layer is the same as any other layer which is 256 input space and 256 output space. We have set the return to True, dropout value to 0.5, and recurrent dropout to 0.5.

4.3.7 Bidirectional GRU Layer

We used the Bidirectional GRU layer after flattening the data in the Bidirectional LSTM layer. We have used only one GRU layer for our model. The input and out dimension of this layer is the same as any other layer which is 256 input space and 256 output space. We have set the return to True, dropout value to 0.5, and recurrent dropout to 0.5.

4.3.8 Flatten Layer

When dealing with multi-dimensional inputs such as picture datasets, the Keras flatten class is crucial. The Keras.layers.flatten function converts multi-dimensional input tensors to a single dimension, allowing us to model our input layer, create our neural network model, and then effectively transmit those inputs to each and every neuron in the model.

4.3.9 Dense Layer

We have used the Dense layer twice in our model to finalize our model. The dense layers were utilized to change the dimensionality of the output from the previous layer, allowing the model to clearly establish the connection between the values of the data it is working with. We have used the activation function ‘Relu’ for the first layer and ‘Softmax’ for the second layer.

4.3.10 Optimizer

We have explored two optimizers SGD (stochastic gradient descent) and RMSprop optimizer. SGD: A class that implements the stochastic gradient descent optimizer we can pass a learning rate and momentum value in the constructor.

RMSprop: A gradient-based optimization approach for neural network training. Gradients have a tendency to vanish when input passes through extremely sophisticated algorithms like neural networks also known as the vanishing gradients problem. RMSprop solves the problem by using a moving average of squared gradients to normalize the gradient.

For our model, we use RMSpropoptimizer as it was generalizing the data better than SGD.

4.3.11 Batch Size

In Deep Learning algorithms, the size of the batch is critical. The amount of training samples utilized in one iteration is referred to as batch size in machine learning. There are three different batch sizes available. The procedure will use stochastic gradient descent if the batch size is one. The algorithm will use mini-batch gradient descent if the batch size is more than 1 but less than the length of the training dataset. We have defined batch size as 1.

Chapter 5

Performance Evaluation

The next parts will exhibit the relevant functional plottings, such as accuracy, validation accuracy, loss, validation loss, and so on, to explain our conclusions from the experiment.

5.1 Experimental Setup

Our experiment used specialization networks processing grid topology, subdivided into three stages of CNN layers with RELU activation, several max-pooling layers, and a flatten layer to classify the feature maps. On the other hand, we also used high-resolution CT scan image data as inputs that require high computational processing. So our main machine has a TensorFlow CUDA core GPU specification: GeForce RTX3060 with a memory space of six gigabytes and Ryzen 5 5600H processing unit. In another device setup, we have a graphics processing four gigabytes of GTX 1050 ti with core i5 as processing. We have used Windows 11 for both of our operating systems.

5.1.1 Language

Implementing any kind of neural network, artificial intelligence, or machine learning project we need that kind of language that is both reliable and adaptable and has some of the best computational tools. Python itself provides all of these which facilitate any programmer. For this project, we are using python version 3.8.

5.1.2 Platform

The platform we have used for our experiment is The Jupyter notebook, an open-source web application. A user can create an online repository, including links to research materials, datasets, code, and methodologies. While working on this experiment, our first change was how we should implement our code parts in a sequential and shareable manner with the help of its JSON format. With the jupyter notebook platform, we can quickly make our experiment sharable and well documented. Furthermore, the pandas' library helps by writing fewer codes and excellent data representation, and it has an extensive set of features. We have used the latest version for our experiment purposes, which is v3.2.1.

5.1.3 Library

Seaborn is a visualization library built upon a matplot library. It performs the required semantic mapping and statistical accumulation for producing an informative plot. Additionally, it uses the matplot library to draw the plots, and for the measurement of statistical values, it uses bootstrapping to compute intervals. For statistical analyses, it includes inclusive computational approaches like kernel density estimation as seaborn integrates with the matplot library, so it automatically adds informative axis labels and supports different exploratory analyses with real-time interaction in GUI applications. In our implementation, we used seaborn plating for the different categories like counting the normal brain ct scan images with hemorrhage brain ct images and sns.

To complete the image's loading and processing, we imported the Pillow module and called it PIL(python imaging library). It provides an efficient internal representation with powerful image processing capabilities. A couple of substitutes for the PIL library are Numpy and Scipy, which deals only with images directly by manipulating pixels. Pillow library is highly recommended for image processing as this doesn't require advanced image processing expertise. There are other advances, faster, and more powerful libraries than pillow modules like Mahotas, scikit-image, and OpenCV. Still, the pillow's main advantage is that it is widely used among the community, and as it is an open-source library, it can be used for exploratory work while dealing with images.

Keras is a free and open-source high-level API that supports multiple backend neural network computation libraries. Keras is only an extension for making it easier to read and write machine learning programs. Unfortunately, Keras can't deal with the low-level computation. We need some low-level APIs like theano, CNTK, or TensorFlow. In our experiment, we used TensorFlow combined with Keras, where the TensorFlow framework supports both high-level and low-level APIs. We used TensorFlow and Keras framework multiple times to import many different libraries such as ImageDataGenerator, Sequential, RMSprop, Adam, Optimizer, etc. While the training process ImageDataGenerator class helps us augment our image in real-time with a particular aspect by flipping, shifting, rotating, etc. It has some subdivided types: Random Rotations, Shifts, Flips, Brightness, and Zoom. After that, sequential API allows us to design layer-by-layer models, which helped generate Keras layers in a sequential ConvNet architecture. Then for training the neural network model, we use RMSprop, which is gradient optimization. As we know, most neural networks deal with vanishing gradients and exploding problems. To normalize the gradient, RMSprop uses an average of the square gradient. Similarly, we imported Adam to minimize the stochastic gradient descent for training the model.

5.2 Result And Analysis For The Dataset

To avoid overfitting in our training model, we implemented a cross-checking validation technique that can make predictions about whether our training model can perform well for our future new datasets. That is why we implemented the validation accuracy and loss function along with the loss accuracy function.

For our experiment, we ran our model into our proposed dataset and, after training our model for nearly 7 hours and 41 epochs, found out that our training accuracy is around 0.9594 and validation accuracy is about 0.9456 (Figure 5.1). In other words, Our model can perform 94.56% accurately in other datasets whether it performs 94.56% on our chosen dataset. We can see that until almost 20 epochs the accuracy increases gradually is a rapid pace. After 30 epoch training, the model reaches a point where the model can no longer learn to improve the training at a very significant amount. Moreover, after 36 epochs, the model reaches a point where it can hardly improve anymore. In our experiment, there is a point where mode validation accuracy reached up to 94.47 %. This result is considered satisfactory and shows significant improvement while training

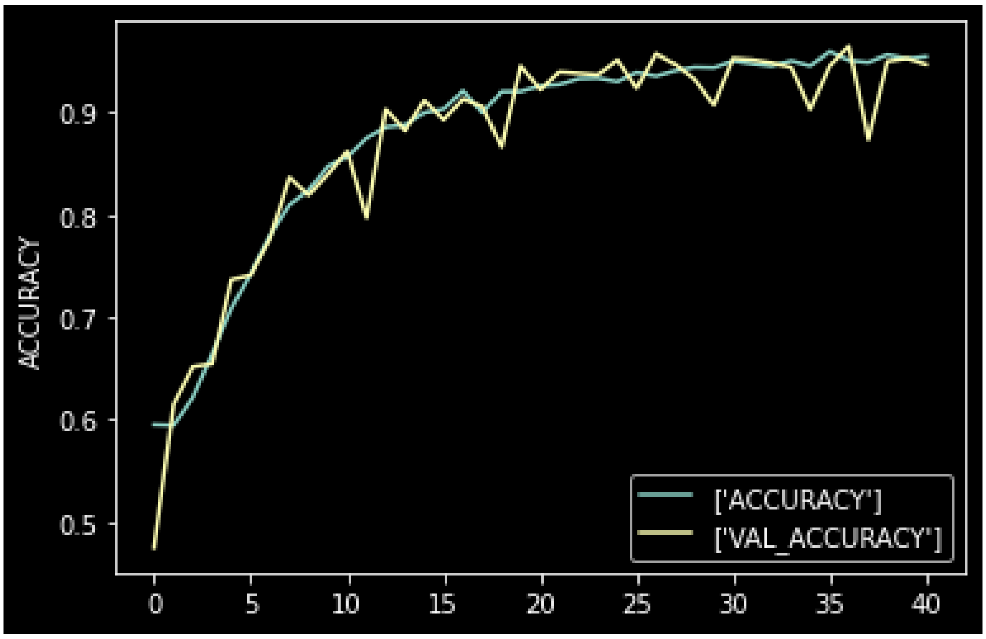


Figure 5.1: Accuracy vs Validation accuracy plot

From Figure 5.2, we can see that the loss function and validation loss function decrease. The loss function refers to the minimization of the error by continuously updating the weights. By calculating the error of each input by looking at what output it predicts for the information and taking the difference of the output value.

We start our loss and validation function with 0.7054 and 0.6961 respectively. After 36 epochs, our loss is around 0.1329, and validation loss is around 0.1572, which is suitable for any training or new dataset. As our validation loss lesser than training loss that means we don't have any overfitting.

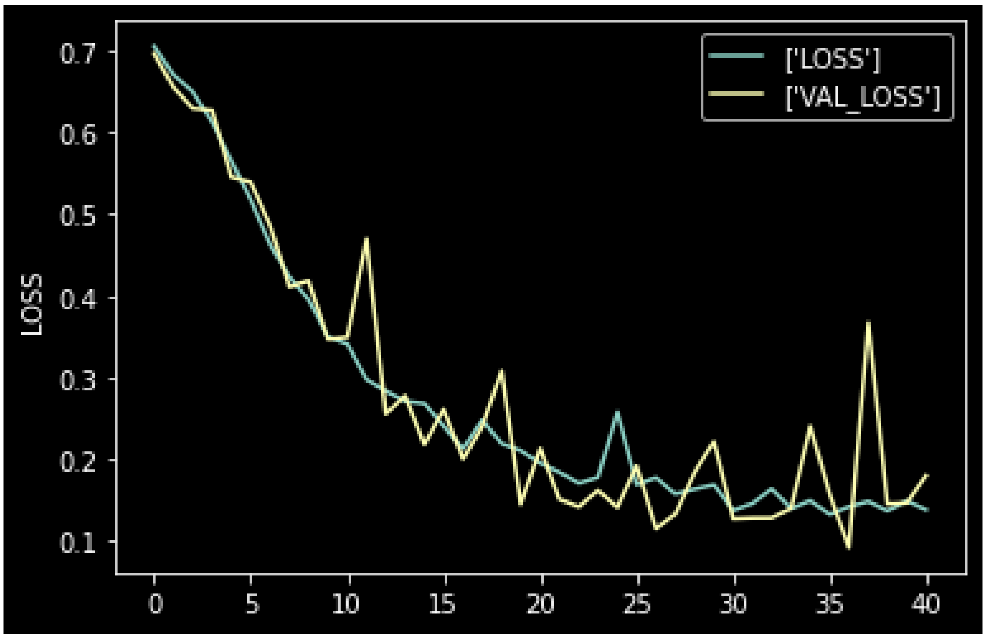


Figure 5.2: Loss vs Validation Loss plot

In figure 5.3, we can see the actual loss vs accuracy function graph. To determine a good model, we can see how lower the loss function is. The loss function mainly calculates training and validation datasets by how well it interprets the model. Forming the summation of the errors, our training set loss curve is in a decreasing slope, which means our model minimizes the error in every training step.

On the other hand, accuracy determines how the model parameters' learnings and fixings occur. In our experiment, after 36 epochs, accuracy is around 95.94%, which means for our 679 test samples, 652 were classified correctly, and in the graph, the accuracy function is constantly increasing.

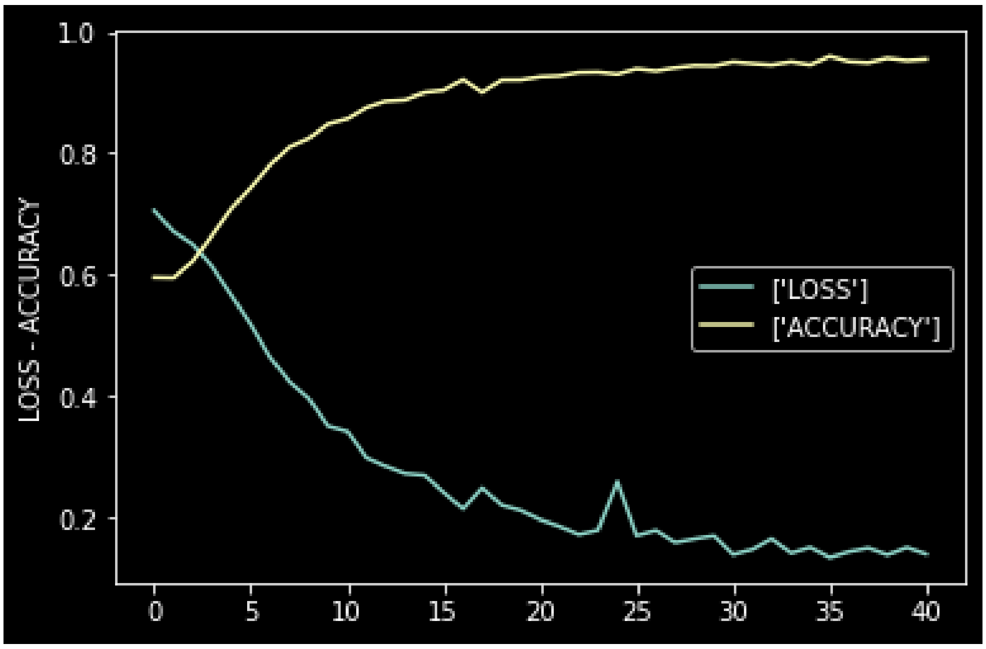


Figure 5.3: Loss vs Accuracy plot

Similar to figure 5.3 in the figure 5.4 graph, we observe how our model will perform for new future datasets. From the start of our chart, the validation loss decreases, whereas the validation accuracy increases behavior. Validation loss and accuracy are a future determination of the model; val_loss of 0.1572 and val_accuracy of 0.9594 are effective for any new dataset.

Validation loss is usually used for training sets by finding the best parameters that will fit the model in future actions. Validation accuracy evaluates how the accuracy will change in future new datasets. Val_accurasy is slightly higher than our training accuracy in our model, which is quite acceptable as it makes the model lesser overfitted.

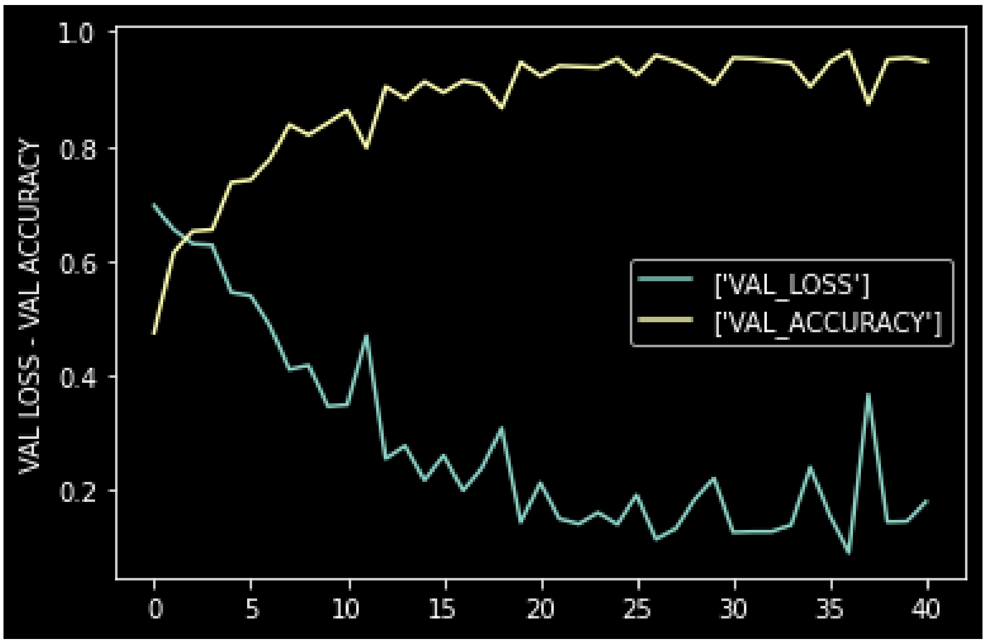


Figure 5.4: Validation Loss vs Validation Accuracy plot

Figure 5.5 shows a summary of the previous graphs with the labels. As we can see, the validation accuracy, training accuracy, and validation loss maintained quite a similar outline in our model, which is a good sign for new datasets.

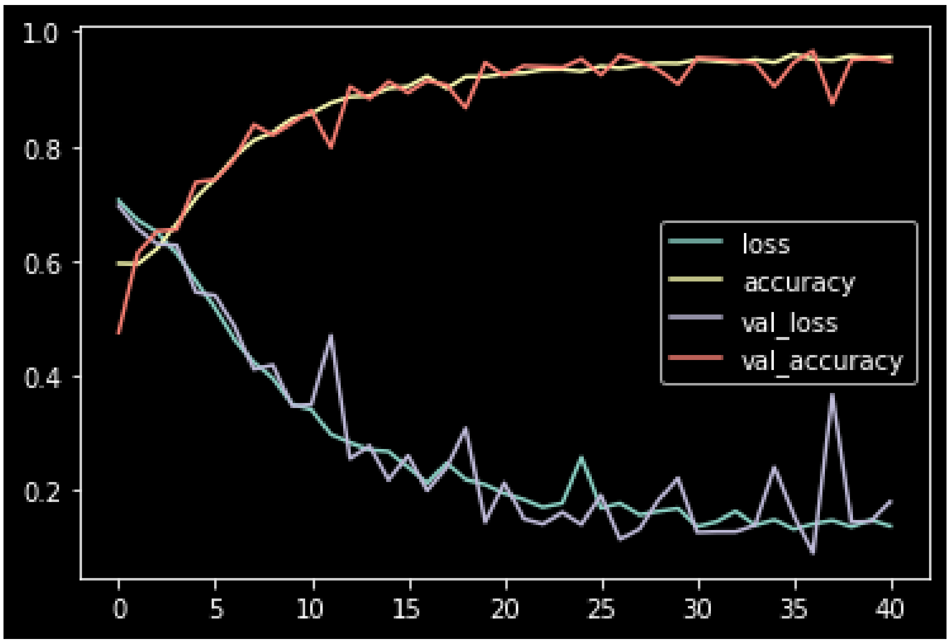


Figure 5.5: Summary plot

5.2.1 Confusion Matrix

Confusion Matrix consists of four parameters. These are True positive, False positive, False-negative, and True negative. True positive (TP) refers to when a model takes a correct sample and classifies it as correct, which means the system is right. False-positive (FP) occurs when the model takes a wrong sample and classifies it as a correct sample, which means the system is wrong. False-negative (FN) occurs when the model takes a correct sample and classifies it as a wrong sample, which means the system is incorrect again. True negative (TN) arises when the model takes a false sample and classifies it as a wrong sample, which means the system is correct in this case.

In our model, we have used the test data to create the following confusion matrix in figure 5.6.

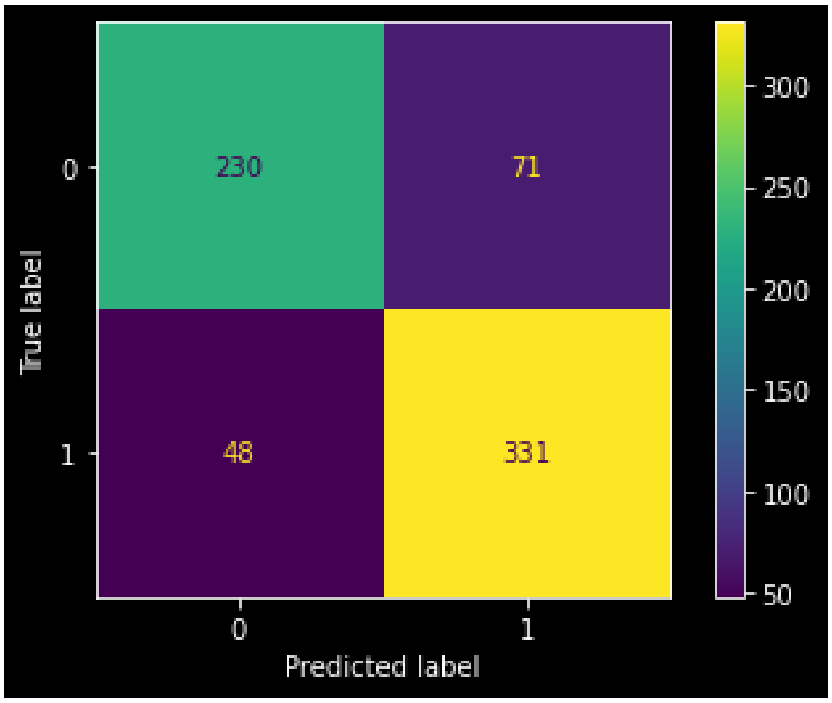


Figure 5.6: Confusion Matrix

The confusion matrix gave us a result of TP = 230, FP = 71, FN = 48, TN = 331. It whos that our model is performing significantly well.

5.2.2 Performance Measure Functions

In any kind of neural network to evaluate how the model teaches itself, we use some performance measure functions. Some popular performance measure functions are precision, recall or sensitivity, F1 score, specificity, MCC, etc. Precision is several samples classified as correct or false positive with how many of them are correct or a number of a true positive. On the other hand, Recall or sensitivity is what the model remembers from reality. Similar to precision, it deals with some samples classified as correct or true positive, but in addition to the number of samples classified as correct when they are incorrect or true negative. Specificity is the combination of samples classified as correct or false positive and samples are classified as correct when they are incorrect or true negative.

In our experiment, we rallied up a result consisting precision of 0.8233, sensitivity of 0.7641, and specificity of 0.8733. Figure 5.7 shows the formula for Precision, Sensitivity, and Specificity for calculation. Figure 5.7: Formula for Precision, Sensitivity, and Specificity is bellow:

$$Precision = \frac{TP}{TP + FP}\ ; \tag{5.1}$$

$$Recall\ or\ Sensitivity = \frac{TN}{TN + TP}\ ; \tag{5.2}$$

$$Specificity = \frac{TN}{TN + FP} \tag{5.3}$$

The F1 score is a special type of harmonic mean of precision and recall. For our experiment, we have found an F1 score of 0.8476. Figure 5.8 shows the formula used for obtaining the F1 score for calculation.Figure 5.8: Formula for F1 score is bellow:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.4}$$

In some situations, the F1 score becomes not valid. When we have highly imbalanced datasets, the classes are (positive and negative) inverted or switched, and the F1 score cares less about true negative values. Matthew's correlation coefficient or MCC comes in handy, which is symmetric and cares about true negative and other three values. While experimenting we got the value of 0.6440 for our MCC. Figure 5.9 shows the formula used for obtaining the MCC.Figure 5.9: Formula for MCC is bellow:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{5.5}$$

To summarize the achieved validation accuracy and loss can be considered valid test results. Table:1 shows the comparison between our train and test result.

The following Table:1 shows the generated result obtained by our dataset.

Function	Train	Test
Accuracy (%)	95.94	94.56

Table 5.1: Comparison between Train-Test Accuracy

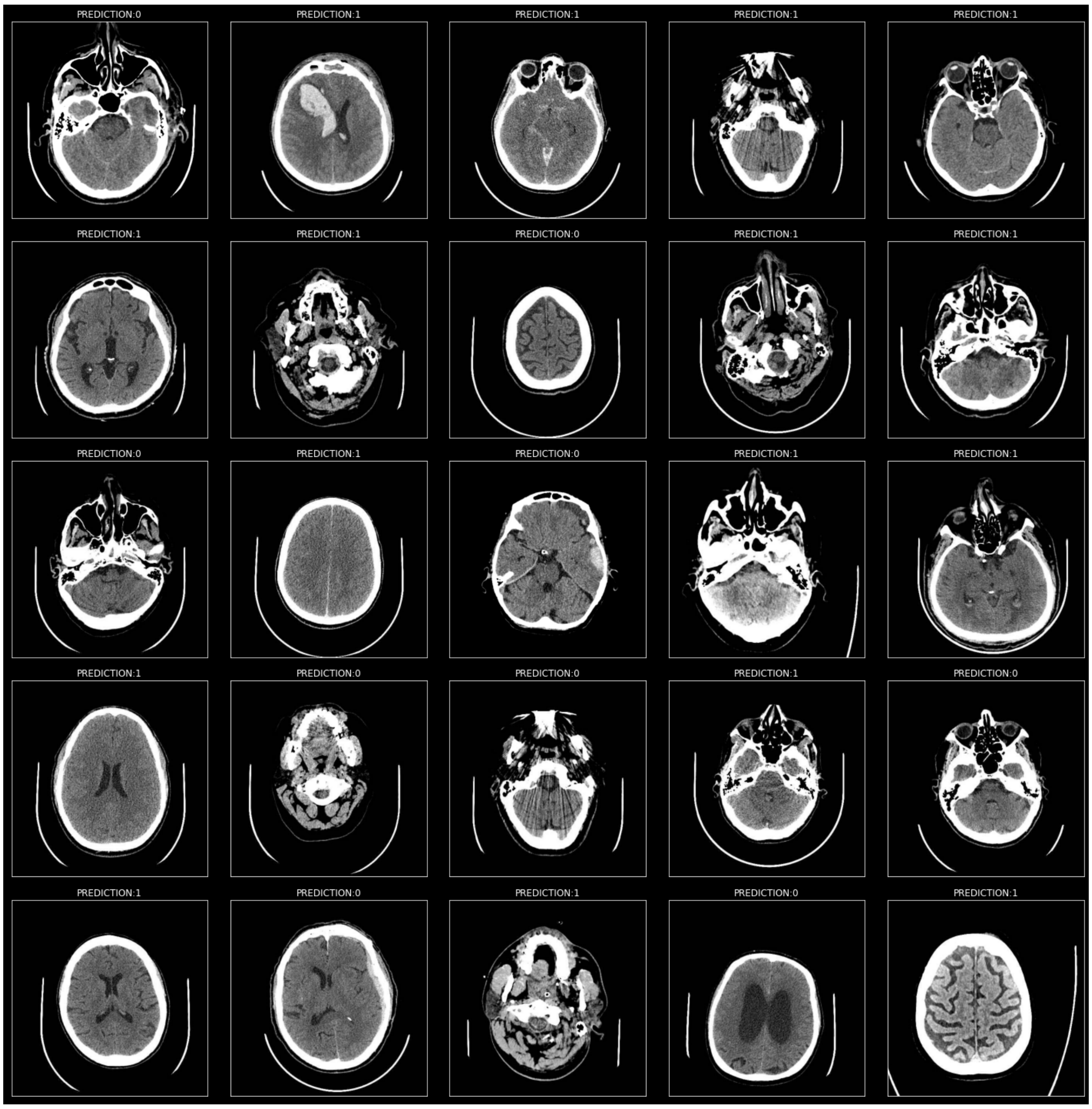


Figure 5.7: Image-set after prediction

Figure 5.7 shows the predictions generated by our model. Here, 0 indicates if a brain is affected by hemorrhage, while 1 indicates that the brain is normal.

Chapter 6

Conclusion

To summarize, because of the high morbidity and death rate, detecting a brain hemorrhage is one of the most important responsibilities. Due to its weak and uneven borders, hemorrhage segmentation from brain CT images is a difficult task. The volume of a hemorrhage can be measured using brain CT scans for prognosis and therapy trials. In our thesis, we attempted to offer an accurate technique for identifying intracranial hemorrhages using CT scan images. Furthermore, our suggested approach has the potential to be utilized for further future works in other image classification fields. The approach provided in our research can also be used for diverse anatomy and disease detection. We have obtained substantial results using our suggested CNN-LSTM Fusion model. However, as we discussed a few DRL methods, instead of using LSTM we should be able to use DRL for the classification after analyzing the images with CNN. Additionally, we can also increase the batch for further improvements. With this, it can be concluded that convolutional neural networks are an effective tool for detecting cerebral hemorrhages in computed tomography images and can assist in the diagnosis of these illnesses. Furthermore, it was discovered that the strategy for selecting the train and test sets has an impact on the performance of deep learning algorithms. As a result, more research into the data independence in the usage of computed tomography images is needed for classification using convolutional neural networks.

Bibliography

[1] "Brain bleed/hemorrhage (intracranial hemorrhage): Causes, symptoms, treatment." https://my.clevelandclinic.org/health/diseases/14480-brain-bleed-hemorrhage-intracranial-hemorrhage. Accessed: 2022-5-23.

[2] M. Hussain, Q. Mohammad, M. Habib, M. Hoque, M. Badrul, and D. M. A. Yusuf, "Aetiology of spontaneous intracerebral haemorrhage in young adults admitted at a tertiary care hospital in dhaka," *American Journal of Neuroscience*, vol. 6, pp. 20–25, 10 2010.

[3] J. L. Ruíz-Sandoval, C. Cantú, and F. Barinagarrementeria, "Intracerebral hemorrhage in young people: analysis of risk factors, location, causes, and prognosis," *Stroke*, vol. 30, pp. 537–541, Mar. 1999.

[4] B. Hathaway, "Brain hemorrhage surgery boosts survival, but disability risk still high," Oct 2019.

[5] J. A. Caceres and J. N. Goldstein, "Intracranial hemorrhage," *Emergency Medicine Clinics of North America*, vol. 30, no. 3, p. 771–794, 2012.

[6] W. Kuo, C. Hne, P. Mukherjee, J. Malik, and E. L. Yuh, "Expert-level detection of acute intracranial hemorrhage on head computed tomography using deep learning," *Proceedings of the National Academy of Sciences*, vol. 116, no. 45, p. 22737–22745, 2019.

[7] S. Hojjatoleslami and F. Kruggel, "Segmentation of large brain lesions," *IEEE Transactions on Medical Imaging*, vol. 20, no. 7, p. 666–669, 2001.

[8] M. Sun, R. Hu, H. Yu, B. Zhao, and H. Ren, "Intracranial hemorrhage detection by 3d voxel segmentation on brain ct images," *2015 International Conference on Wireless Communications amp; Signal Processing (WCSP)*, 2015.

[9] C. Nin, "A beginner's guide to deep reinforcement learning," 2020.

[10] Y.-n. Dong and G.-s. Liang, "Research and discussion on image recognition and classification algorithm based on deep learning," *2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, 2019.

[11] L. Li, M. Wei, B. Liu, K. Atchaneeyasakul, F. Zhou, Z. Pan, S. A. Kumar, J. Y. Zhang, Y. Pu, D. S. Liebeskind, and et al., "Deep learning for hemorrhagic lesion detection and segmentation on brain ct images," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 5, p. 1646–1659, 2021.

[12] H. Ko, H. Chung, H. Lee, and J. Lee, "Feasible study on intracranial hemorrhage detection and classification using a cnn-lstm network," *2020 42nd Annual International Conference of the IEEE Engineering in Medicine amp; Biology Society (EMBC)*, 2020.

[13] A. Patel, S. C. van de Leemput, M. Prokop, B. Van Ginneken, and R. Manniesing, "Image level training and prediction: Intracranial hemorrhage identification in 3d non-contrast ct," *IEEE Access*, vol. 7, p. 92355–92364, 2019.

[14] J. M. Jose Valanarasu, R. Yasarla, P. Wang, I. Hacihaliloglu, and V. M. Patel, "Learning to segment brain anatomy from 2d ultrasound with less data," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 6, p. 1221–1234, 2020.

[15] P. R. Tabrizi, A. Mansoor, R. Obeid, J. J. Cerrolaza, D. A. Perez, J. Zember, A. Penn, and M. G. Linguraru, "Ultrasound-based phenotyping of lateral ventricles to predict hydrocephalus outcome in premature neonates," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 11, p. 3026–3034, 2020.

[16] M. Togacar, Z. Comert, B. Ergen, and U. Budak, "Brain hemorrhage detection based on heat maps, autoencoder and cnn architecture," *2019 1st International Informatics and Software Engineering Conference (UBMYK)*, 2019.

[17] Z. Li and Y. Xia, "Deep reinforcement learning for weakly-supervised lymph node segmentation in ct images," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 3, p. 774–783, 2021.

[18] H. Ko, H. Chung, H. Lee, and J. Lee, "Feasible study on intracranial hemorrhage detection and classification using a cnn-lstm network," *2020 42nd Annual International Conference of the IEEE Engineering in Medicine amp; Biology Society (EMBC)*, 2020.

[19] A. Helwan, G. El-Fakhri, H. Sasani, and D. Uzun Ozsahin, "Deep networks in identifying ct brain hemorrhage," *Journal of Intelligent amp; Fuzzy Systems*, vol. 35, no. 2, p. 2215–2228, 2018.

[20] S. Chilamkurthy, R. Ghosh, S. Tanamala, M. Biviji, N. G. Campeau, V. K. Venugopal, V. Mahajan, P. Rao, and P. Warier, "Development and validation of deep learning algorithms for detection of critical findings in head ct scans," *ArXiv*, vol. abs/1803.05854, 2018.