TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
Kathmandu Engineering College
Department of Computer Engineering

Final Report

On

Detection of intracranial hemorrhage on CT images using deep learning

[Code No: CT 755]

By:

Safal Shrestha - KAT074BCT054

Sandeep Narshing Pradhan - KAT074BCT061

Suraj Chand - KAT074BCT083

Tirtha Shrestha - KAT074BCT092

Kathmandu, Nepal
Baisakh, 2079

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering

Detection of intracranial hemorrhage on CT images using deep learning

[Code No: CT 755]

PROJECT REPORT SUBMITTED TO THE DEPARTMENT

OF COMPUTER ENGINEERING IN PARTIAL

FULFILLMENT OF THE REQUIREMENT FOR THE

BACHELOR OF ENGINEERING

By

Safal Shrestha - KAT074BCT054

Sandeep Narshing Pradhan - KAT074BCT061

Suraj Chand - KAT074BCT083

Tirtha Shrestha - KAT074BCT092

Kathmandu, Nepal

Baisakh, 2079

TRIBHUWAN UNIVERSITY

Kathmandu Engineering College

Department of Computer Engineering

ACKNOWLEDGEMENT

Before we get into the thick of things, we would like to add a few words of appreciation for everyone who has been a part of this project, right from its inception. This project has been one of the significant academic challenges we have faced and without the support, patience, and guidance of the people involved, this task would not have been completed. We owe our deepest gratitude to them.

Our special thanks go to the entire faculty of the Computer Engineering Department, who supported us through all the thick and thin during our time at the college. We acknowledge the support and help provided by our Year Coordinator and Project Coordinator, Er. Sharad Chandra Joshi. We would also like to thank Er. Mahesh Singh Khadayat for the tremendous help that he has provided to us. Without their support, guidance and encouragement, this project couldn't have been put together. We would like to thank everyone involved in making this possible, Er. Sudeep Shakya, Head of Department of Computer Engineering and Er. Kunjan Amatya, Deputy Head of Department of Computer Engineering.

And lastly to all the respondents for all their full cooperation that made them a big part of this project.

ABSTRACT

Intracranial hemorrhage (ICH) is bleeding between the brain tissue and skull or within the brain tissue itself, which is mainly liable for stroke. It is a life- threatening emergency. X-rays and computed tomography (CT) scans are widely applied for locating the hemorrhage position and size. Since manual segmentation of the CT scans by planimetry by the use of radiologists is a time-consuming process, deep learning (DL) is used to attain effective ICH diagnosis performance. This project presents an automated intracranial hemorrhage diagnosis using Convolutional Neural Network (CNN) with global average pooling. The obtained DICOM images are preprocessed and trained using CNN with 3 layers having 16, 64 and 128 layers each. Output layer of 5 nodes is extracted for 5 classes of Intracranial Hemorrhage with Sigmoid activation function.   This helps to detect the healthy brain or five different types of intracranial hemorrhage. The obtained accuracy is 63.37% which can be further improved with future enhancement. Thus, for now this system is inadequate to use in the real system.

Keywords: *Intracranial Hemorrhage, CT, CNN.*

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATION

AI:	Artificial Intelligence
ANN:	Artificial Neural Network
CNN:	Convolutional Neural Network
CPU:	Central Processing Unit
CT:	Computer Tomography
EIT:	Electrical Impedance Tomography
GPU:	Graphics Processing Unit
ICH:	Intracranial Hemorrhage
MRI:	Magnetic Resonance Imaging
MSE:	Medical Support Equipment
SDLC:	Software Development Life Cycle
SVM:	Support Vector Machine
VGG-NET:	Visual Geometry Group Network

CHAPTER 1: INTRODUCTION

1.1 Background

Hemorrhage in the head (intracranial hemorrhage) is a relatively common condition that has many causes ranging from trauma, stroke, aneurysm, vascular malformations, high blood pressure, illicit drugs and blood clotting disorders. The neurologic consequences also vary extensively depending upon the size, type of hemorrhage and location ranging from headache to death. The role of the Radiologist is to detect the hemorrhage, characterize the hemorrhage subtypes, its size and to determine if the hemorrhage might be jeopardizing critical areas of the brain that might require immediate surgery.

Causes of Intracranial Hemorrhage

- Hypertension: elevated blood pressure may cause tiny arteries to burst inside the brain. Normal pressure is 120/80 mm Hg.
- Blood thinners: drugs such as coumadin, heparin, and warfarin used to prevent clots in heart and stroke conditions may cause ICH.
- AVM: a tangle of abnormal arteries and veins with no capillaries in between.
- Aneurysm a bulge or weakening of an artery wall.
- Head trauma: fractures to the skull and penetrating wounds (gunshot) can damage an artery and cause bleeding.
- Bleeding disorders: hemophilia, sickle cell anemia, DIC, thrombocytopenia.
- Tumors: highly vascular tumors such as angiomas and metastatic tumors can bleed into the brain tissue.
- Amyloid angiopathy: a buildup of protein within the walls of arteries.
- Drug usage: alcohol, cocaine and other illicit drugs can cause ICH.
- Spontaneous: ICH by unknown causes.

Symptoms of Intracranial Hemorrhage

- The symptoms usually come on suddenly and can vary depending on the location of the bleed. Common symptoms include:
- Headache, nausea, and vomiting

- Lethargy or confusion
- Sudden weakness or numbness of the face, arm or leg, usually on one side
- Loss of consciousness
- Temporary loss of vision
- Seizures

Types of Hemorrhage

To know about the types of hemorrhage, we first need to know about the different layers of the human cranium.

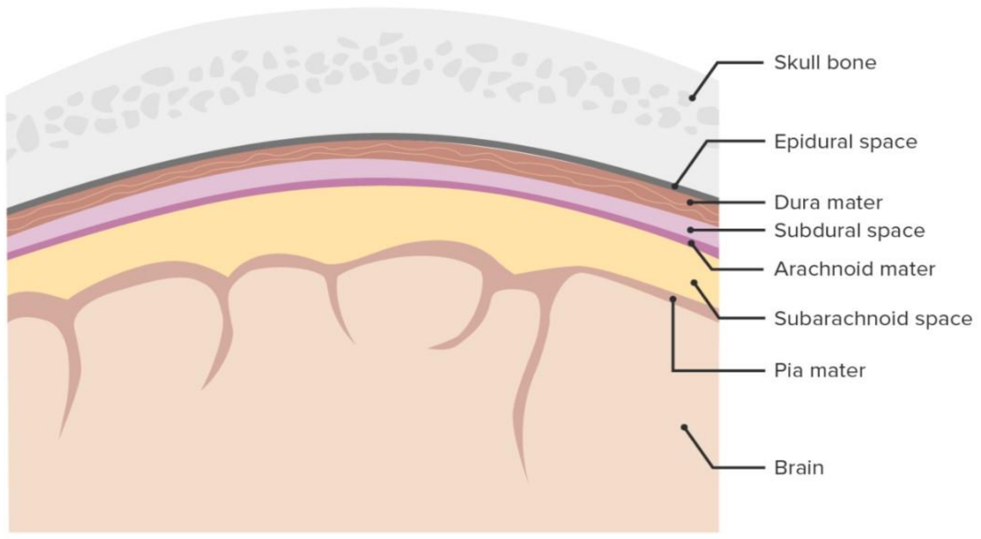


Fig 1.1.1 Layers of the cranium

The image above shows the different layers that protects the human brain. On the basis of location of discharge of blood inside the human cranium. The different hemorrhages are classified as:

1. Intraparenchymal Hemorrhage
2. Intraventricular Hemorrhage
3. Subarachnoid Hemorrhage
4. Subdural Hemorrhage
5. Epidural Hemorrhage

1. Intraparenchymal Hemorrhage

Intraparenchymal hemorrhage is bleeding into the brain parenchyma proper. There is a wide variety of reasons due to which hemorrhage can occur including, but not limited to, hypertension, arteriovenous malformation, amyloid angiopathy, aneurysm rupture, tumor, coagulopathy, infection, vasculitis, and trauma. Intraparenchymal hemorrhage accounts for 10% to 20% of all strokes. Intraparenchymal hemorrhage incidence increases for those aged 55 and older with an increasing incidence as age increases. There is some controversy regarding gender differences, but there may be a slight male predominance.

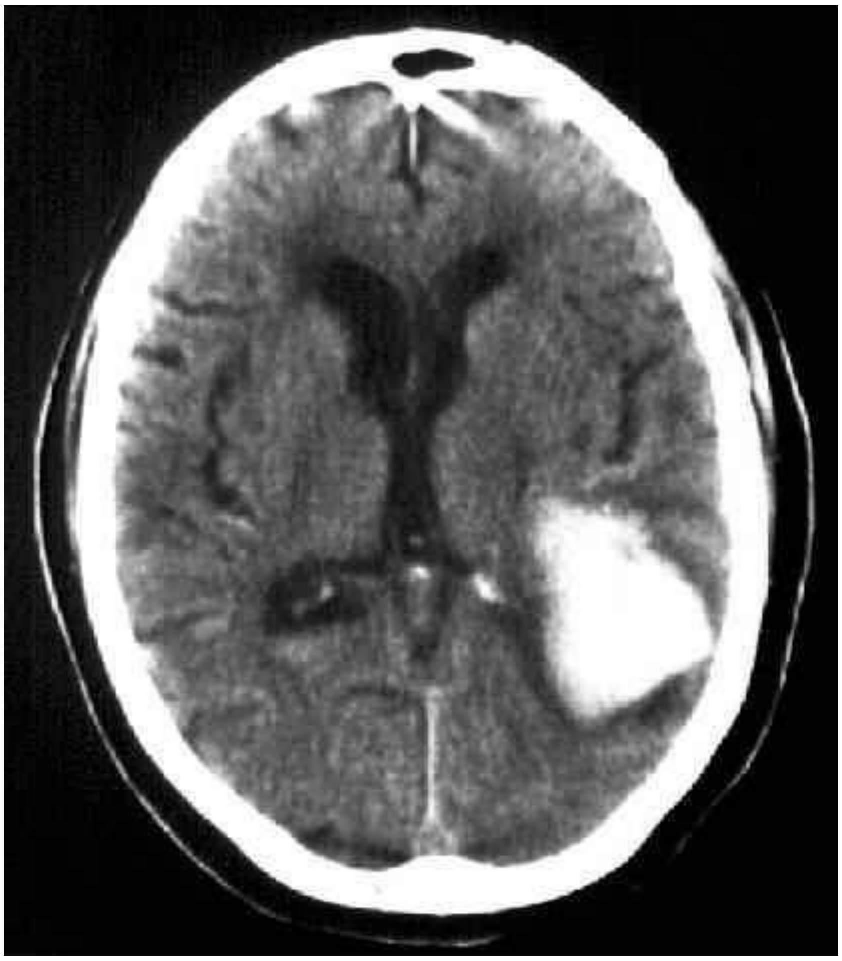


Fig 1.1.2 Intraparenchymal Hemorrhage

2. Intraventricular Hemorrhage

Intraventricular hemorrhage (IVH) is defined as the bleeding into the fluid-filled areas, or ventricles, surrounded by the brain. The condition is most often seen in premature babies, and the smaller and more premature the infant, the higher the risk for IVH. This is because blood vessels in the brain of premature infants are not yet fully developed and are extremely fragile. IVH is rarely present at birth, and if it occurs, it will usually be in the first several days of life.

The condition is quite rare after one month of age, no matter how early the baby was born. But IVH is more common in premature babies who have had physical stress, such as respiratory distress syndrome, pneumothorax or high blood pressure. The condition may also occur in healthy premature babies who were born without injury. IVH may develop in full-term babies, though it is very uncommon.

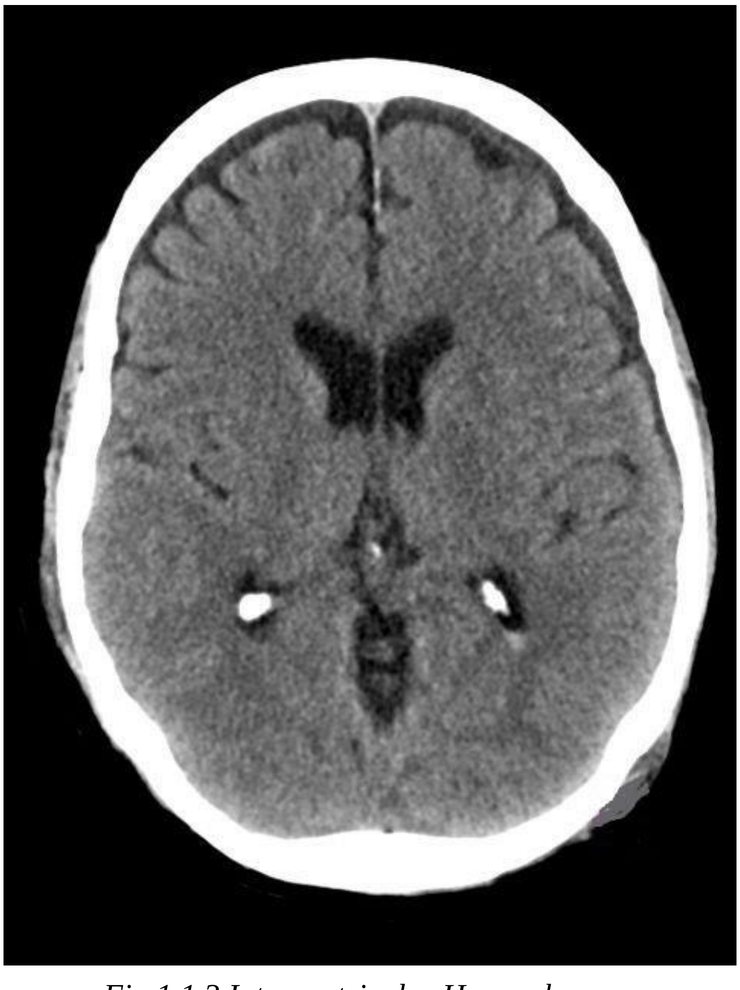


Fig 1.1.3 Intraventricular Hemorrhage

3. Subarachnoid Hemorrhage

A subarachnoid hemorrhage is bleeding into the subarachnoid. Subarachnoid hemorrhage is divided into traumatic versus non-traumatic subarachnoid hemorrhage. A second categorization scheme divides subarachnoid hemorrhage into an aneurysmal and nonaneurysmal subarachnoid hemorrhage. Aneurysmal subarachnoid hemorrhage occurs after the rupture of a cerebral aneurysm allowing for bleeding into the subarachnoid space. Nonaneurysmal subarachnoid hemorrhage is bleeding into the subarachnoid space without identifiable aneurysms. Non-aneurysmal subarachnoid hemorrhage most commonly occurs after trauma with a blunt head injury with or without penetrating trauma or sudden acceleration changes to the head.

Subarachnoid hemorrhage accounts for approximately 5% of all strokes and has an incidence of approximately two to 25 per 100,000 person-years for those over the age of 35. The incidence trends up slowly as patients age and may be very slightly more frequent in females than males (1.15:1 for the female to male ratio).

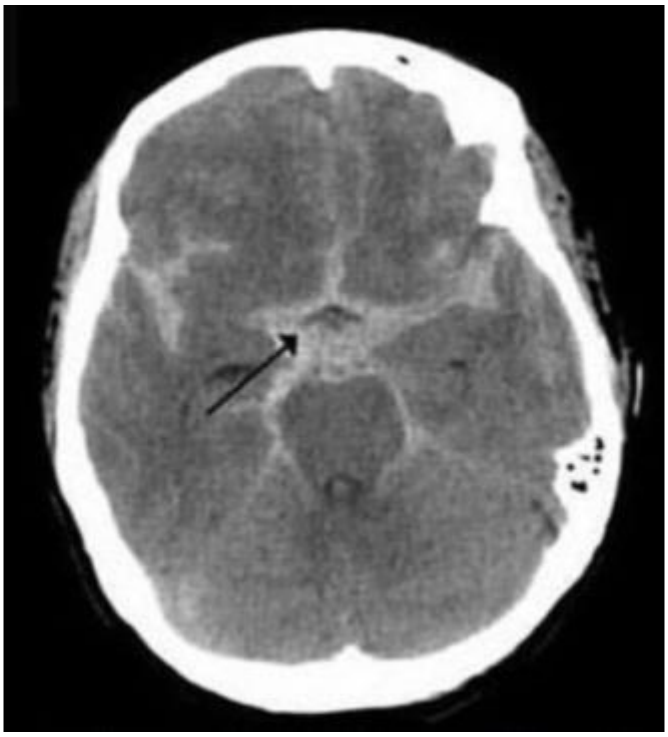


Fig 1.1.4 Subarachnoid Hemorrhage

4. Subdural Hemorrhage

Subdural hemorrhage occurs when blood enters the subdural space which is anatomically the arachnoid space. Commonly subdural hemorrhage occurs after a vessel traversing between the brain and skull is stretched, broken, or torn and begins to bleed into the subdural space. These most commonly occur after a blunt head injury but may also occur after penetrating head injuries or spontaneously. The incidence of subdural hematoma is estimated to be between 5% to 25% of patients with a significant head injury. There is an annual incidence of one to five cases per 100,000 population per year with a male to female ratio of 2:1. The incidence of subdural hematomas increases throughout life.

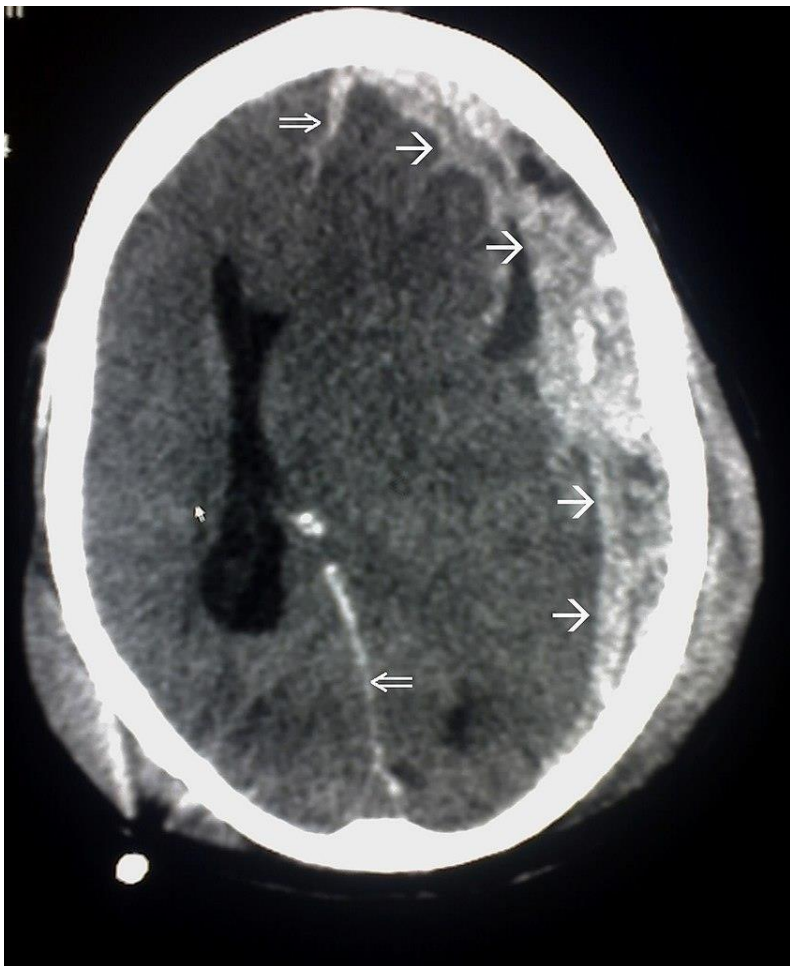


Fig 1.1.5 Subdural Hemorrhage

5. Epidural Hemorrhage

An epidural hemorrhage can either be arterial or venous in origin. The classical arterial epidural hemorrhage occurs after blunt trauma to the head, typically the temporal region. They may also occur after a penetrating head injury. There is typically a skull fracture with damage to the middle meningeal artery causing arterial bleeding into the potential epidural space. Although the middle meningeal artery is the classically described artery, any meningeal artery can lead to arterial epidural hemorrhage.

A venous epidural hemorrhage occurs when there is a skull fracture, and the venous bleeding from the skull fracture fills the epidural space. Venous epidural hemorrhages are common in pediatric patients. Epidural hematomas are present in approximately 2% of head injury patients and account for 5% to 15% of fatal head injuries. Approximately 85% to 95% of epidural hematomas have an overlying skull fracture.

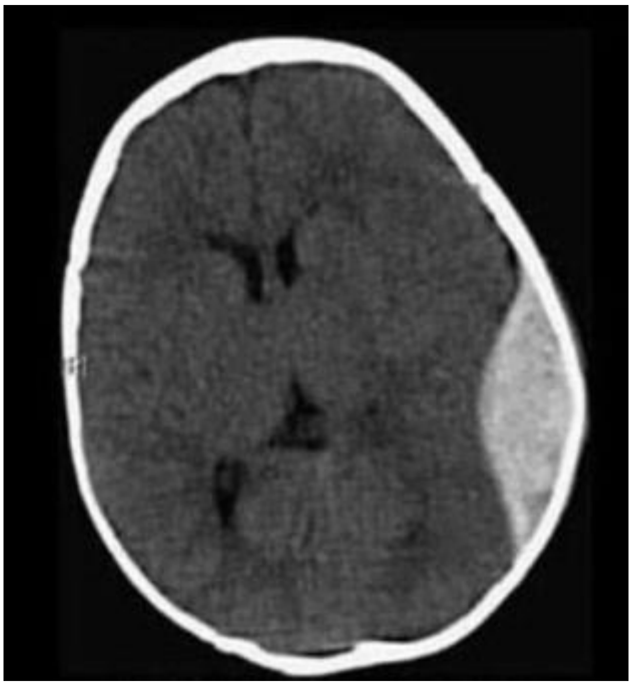


Fig 1.1.6 Epidural Hemorrhage

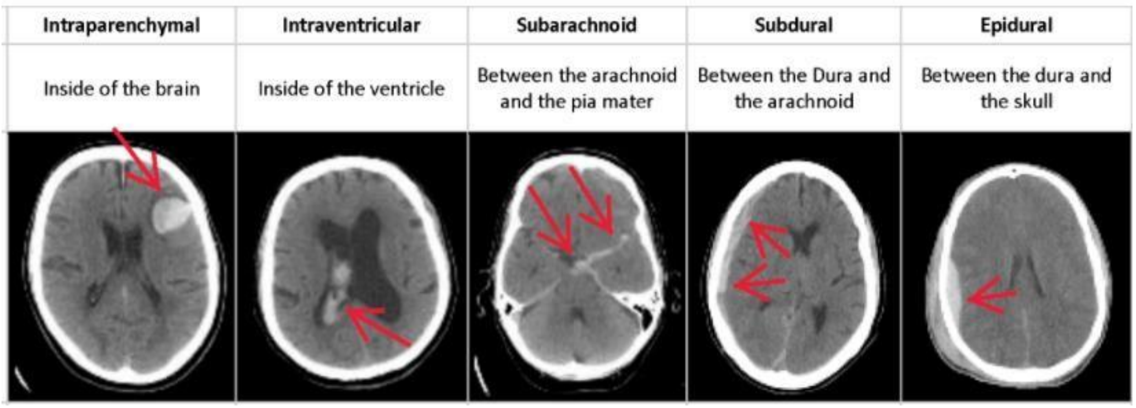

Intraparenchymal
Inside of the brain
Intraventricular
Inside of the ventricle
Subarachnoid
Between the arachnoid and the pia mater
Subdural
Between the Dura and the arachnoid
Epidural
Between the dura and the skull

Fig 1.1.7 Different types of Intracranial Hemorrhage

1.2 Problem Statement

Intracranial Hemorrhage is a critical situation when the bleeding occurs inside the cranium. It is a very serious health problems and it can be deadly. So, the earlier the ICH is detected, the better it will be. Computer Tomography is a standard way of identifying the ICH. To identify hemorrhage from CT images, a high level of expertise is needed. Even if the individual is expert in identifying hemorrhage from CT, there is a certain possibility that there can be any error, and this error can be very fatal. This is done to get accurate detection of hemorrhage. So, this program is responsible for detecting hemorrhage using convolutional neural network.

1.3 Objective

The main objective is to detect healthy brain form hemorrhage one with its five subtypes (cerebral parenchymal, intraventricular, subdural, epidural, and subarachnoid) from CT images using CNN.

1.4 Organization of the Report

The material presented at the report is organized into five chapters. After this introductory chapter, chapter 2 provides review about the projects related to the subject along with the methods and algorithms used with obtained accuracy.

Chapter 3 describes about the methodology of the project. This includes description about the used process model, algorithm, tools used, verification and validation.

Chapter 4 discusses about the o result obtained from the project.

Chapter 5 discusses the viability of the project in the real-world application.

CHAPTER 2: LITERATURE REVIEW

There are numerous studies in the literature that employ quite different methods, in order to detect intracranial hemorrhage from CT scans. A variety of methods have been developed to detect intracranial hemorrhages or to measure hemorrhage volume using standard image processing approaches. These approaches typically take a sequential approach of detecting the head within the image, aligning the head, removing the skull, compensating for CT cupping artifacts, extracting handcrafted features from the imaged brain tissue, and classifying intracranial hemorrhage voxels based on the features. Medical diagnosis has been quite effective with the enormous development of different models of neural network [1].

Deep learning techniques were employed to classify brain computer tomography (CT) images into hemorrhage or healthy. The authors used autoencoders and deep convolutional neural networks to perform this task. As authors claimed, the employed models performed differently when trained and tested on 2527 images. It was found that the stacked autoencoder used in their paper consists of three hidden layers and outperformed other employed networks, where it achieved the highest classification rate and the lowest MSE. The authors concluded that the possible reason of this outperformance on the stacked autoencoder over convolutional neural network is due to the small number of data used for training, as a CNN needs large amount of training examples in order to converge [2].

In another study, brain hemorrhage was examined in more refined manner by feeding using the watershed algorithm along with artificial neural network (ANN) for CT identification of brain hemorrhage type. The authors of this work used features extraction before feeding images to the neural classifier, in which different features were extracted using grey-level co-occurrence matrix (GLCM). Features were then classified by a conventional backpropagation neural network used to identify the type of hemorrhage. They found that adequate image processing techniques such as noise removal and high segmentation methods are required for accurate identification of hemorrhage [3]

A report on CT brain images of head trauma focused on dividing brain CT images into regions, where each region could either be normal or hemorrhage. For images containing hemorrhage, the regions which did not include hemorrhage were treated as normal regions resulting in a highly imbalanced dataset. The researcher had utilized an image segmentation scheme that used ellipse fitting, background removal, and wavelet decomposition technique. The weighted precision and recall value for this approach were approximately 83.6% and 88.5%, respectively [4].

Recent study [5], has identified the brain hemorrhage using CNN along with the types. They have made comparison between convolutional neural network created from scratch, the original AlexNet and modified pretrained AlexNet from which they have concluded that modified AlexNet has outperform other two.

A demonstration on deep learning algorithm was made for detection and localization of acute intracranial hemorrhage on head CT, based on a strong supervision approach and a relatively small training dataset. They have used PatchFCN model and have made comparison with Mask R-CNN. They have shown that FCN with pixel-level supervision was well-suited for their application then Mask R-CNN [6].

Brain hemorrhage detection using a SVM classifier with electrical impedance tomography measurement frames [7] was done. A 2-layer model of the head, along with a series of hemorrhages, was designed as both numerical models and physical phantoms. The phantom models provided with maximal sensitivity and specificity of 75% when used with the linear SVM.

A smart IoT-based application [8], was presented using machine learning algorithms for the human brain hemorrhage diagnosis. Based on the computerized tomography scan images for intracranial dataset, the support vector machine and feedforward neural network have been applied for the classification purposes. Overall, classification results of 80.67% and 86.7% were calculated for the support vector machine and feedforward neural network, respectively.

In another study [9], brain hemorrhage was detected using CT scans of the brain. Concepts of data augmentation and windowing were used by them. The dataset was quite small therefore, they used data augmentation to reduce overfitting and to help the model generalize better. Keras Densenet121 encoder with a Unet decoder - Adam optimizer and dice loss were used as model. Instead of using DenseNet pre-processing, they simply normalized the images by dividing by 255 [9].

CHAPTER 3: METHODOLOGY

3.1 Process Model

Our project has been developed using Agile software model. It is a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or do not directly involve long term planning. Different steps involved in this software development models are explained as:

1. Concept and Inception

This is the first step in agile process model, also known as envision phase. The main purpose of this stage is to discuss about the project vision and the return of investment justification. This is a high-level feasibility discussion and does not delve into specific details. During this step, we identified about the team members, resources and the total time required to complete the project.

2. Construction and Iteration

After discussing the feasibility of the project, the actual system was designed and developed. In this step, we prepared a prototype of what the end result of the project would be like. Designing was done using different UML diagrams like DFD, Use case diagrams, etc. After designing, prototype of the system was made and iterated for different amount of data. While developing different prototypes, modifications were made on the versions according to the newer requirements/

3. Release

After reviewing through the different versions, the final product of the system was implemented and used for predictions. In this step, the final software is released and made into production.

4. Production

This is the final step in agile software model. In this step, all the maintenance and implementation of the software is done. Feedbacks are also received during this step and new features are implemented during this step.

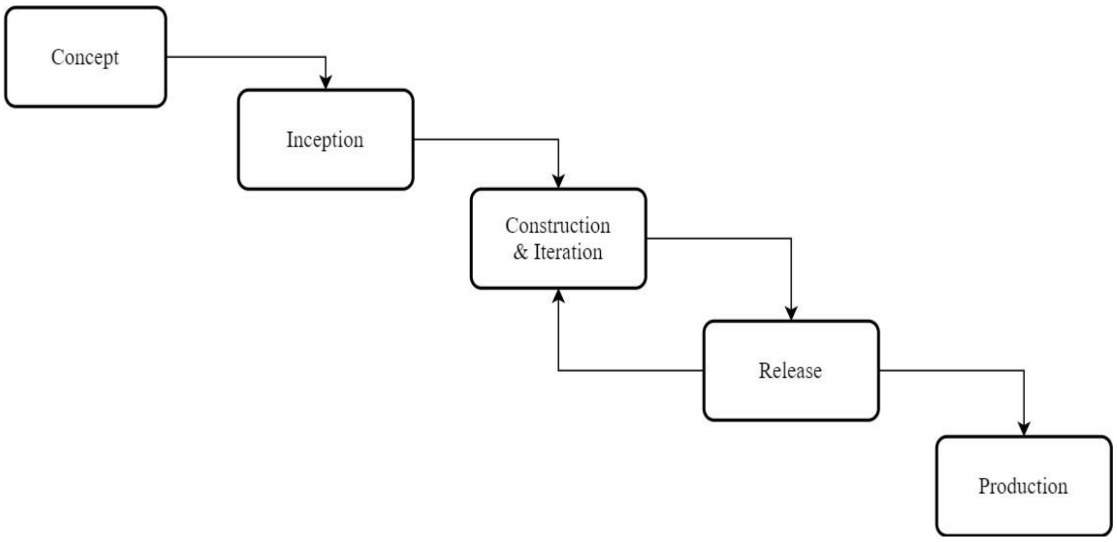


Fig 3.1 Agile Process Model

3.2 System Block Diagram

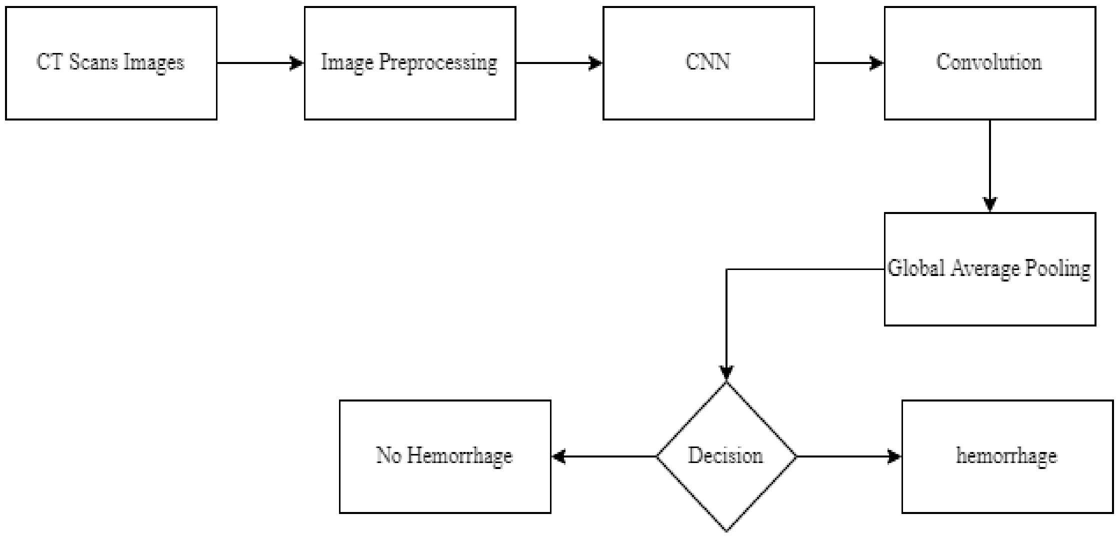


Fig 3.2 System Block Diagram

This system takes a CT image as an input. The task of the program is to convert a CT image into a machine-readable format, which can then be used to diagnose the types of intracranial hemorrhage. After preprocessing we use a CNN to classify each type.

The steps of the process are described below.

3.2.1 Preprocessing

Computer Tomography is a scanning that takes images of X-rays which are sent to the body from different angles and combined using a computer processor to access cross-sectional images (slices) of bones, blood vessels, and soft tissues in various parts of the body. These images provide more detailed information than regular x-ray images. In this way, anomalies in the bones, veins or tissues of the patient are detected. That's why, a more precise diagnosis can be essential for patient and the treatment would continue accordingly.

DICOM is an acronym for Digital Imaging and Communication in Medicine. Files in this format are most likely saved with a 'dcm' file extension. DICOM is both a communication protocol and a file format. This means that a patient can store medical information such as ultrasound, CT and MRI images along with their information in a single file. While png or jpg files contain only the name, date, and number of pixels of the picture, dicom format includes the patient's information, windowing intervals of the picture, which we call meta data. Briefly it includes more detailed information of patients. So, preprocessing these CT images is vital and the steps involved in this project are explained below.

1. Image read

In this step, images are extracted from .dcm files that were obtained from the source data. DICOM library for python was used for this.

2. Resampling

Resampling is the mathematical technique to create a new version of the image with a different width and height in pixels. Down sampling was done during the preprocessing step to reduce the pixels of the image.

3. Conversion to Hounsfield unit

The Hounsfield unit (HU) scale is a linear transformation of the original linear attenuation coefficient measurement in one in which the radiodensity of distilled water at standard pressure and temperature (STP) is defined as zero Hounsfield units (HU), while the radiodensity of air at STP is defined as -1000 HU. Mathematically, it can be represented as follows:

HU = Pixel value * Slope + Intercept

Where, HU = Hounsfield unit

Pixel value = Intensity of image of specific grid.

Slope = Rescale Slope stored in the DICOM file.

Intercept = Rescale Intercept stored in the DICOM file.

4. Windowing

Windowing, also known as grey-level mapping, contrast stretching, histogram modification or contrast enhancement is the process in which the CT image greyscale component of an image is manipulated via the CT numbers; doing this will change the appearance of the picture to highlight particular structures. The brightness of the image is adjusted via the window level. The contrast is adjusted via the window width.

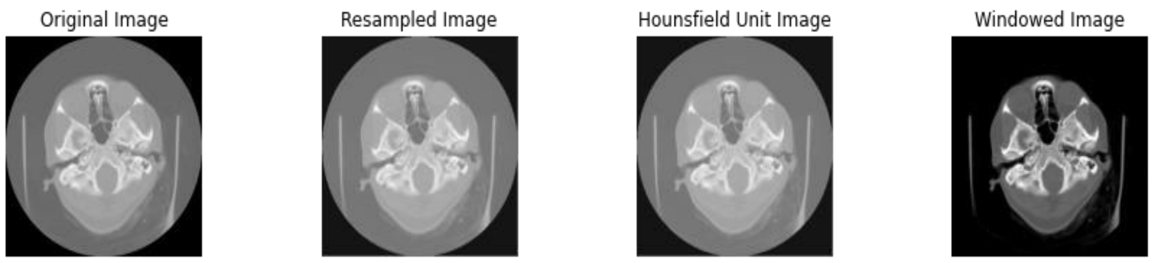


Fig 3.2.1.1 Image Processing

5. Noise Removal via masking

In this step, noise removal of the CT images was done using masking. The edges where the hemorrhage was seen was made prominent and differentiable from the rest of the portion of the image.

6. Cropping

In this step, cropping of the CT images was done. For this, a mask with background pixels was calculated, the brain area was selected, and the background was removed.

7. Padding

In this step, extra pixels were added to the CT images. This was done to avoid the data loss of the edges of the image.

8. Conversion to png

In this step, the .dcm images were converted to .png extension. For this, images were saved in .png extension with gray map coloring.

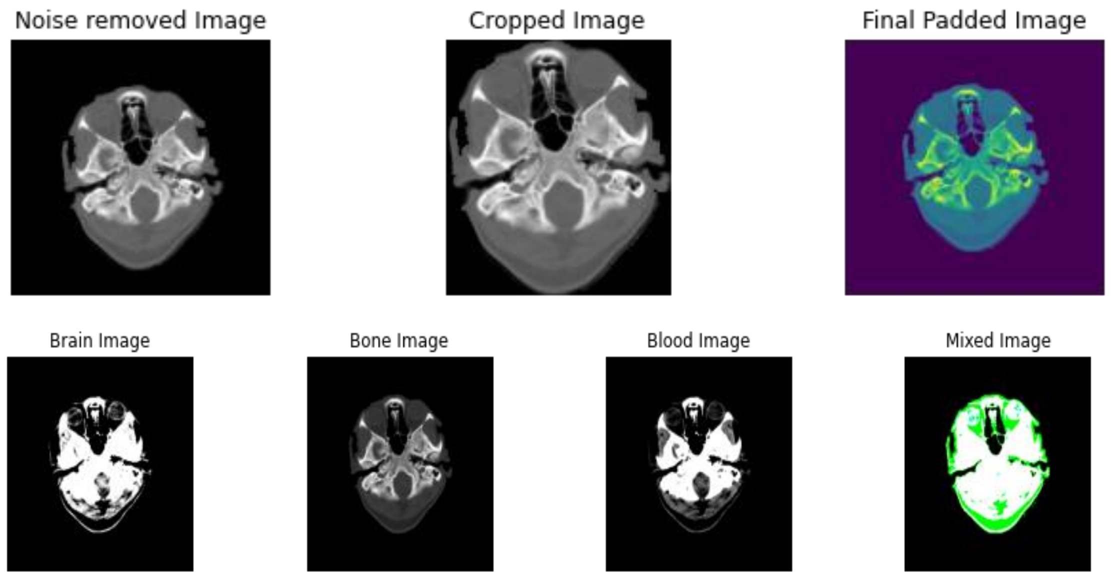


Fig 3.2.1.2 Mapping of Image Using Different Filters

3.2.2 Neural network

Convolution neural network is a popular deep learning algorithm which is mostly used for analyzing visual imagery. This CNN takes in preprocessed CT images with average pooling and a filter is assigned. This filter in the convolutional layer allows to detect abstracts like boundary and edges. By stacking the layers of convolutions on top of each other, more abstract and in-depth information is obtained. In this way the neural network was trained, and decision is done on whether Hemorrhage is present or not in the CT images [11].

3.2.3 CNN Model explanation

CNN is a type of neural network model which allows us to extract higher representations for the image content. CNN takes the image's raw pixel data, trains the model, then extracts the features automatically for better classification.

The CNN for this project has the following features:

1. Image size of 275 * 275 pixels was used as input. A multilayered input was used for this.
2. Three Convolutional layers were used of 3 * 3 kernel size.
    a. Convolution Layer 1 has 16 filters.
    b. Convolution Layer 2 has 64 filters.
    c. Convolution Layer 3 has 128 filters.
3. Convolutional Layers has Batch Normalization, relu as activation function and Average pooling.
4. At first, three hidden layers of 128 nodes and 16 nodes were used with relu activation function.
5. A hidden layer of 5 nodes was added with sigmoid activation function.
6. All the three Input layers are now concatenated with a hidden layer of 15 nodes with relu activation function.
7. Finally, an output layer of 5 nodes was added for 5 classes of Intracranial Hemorrhage with Sigmoid activation function.

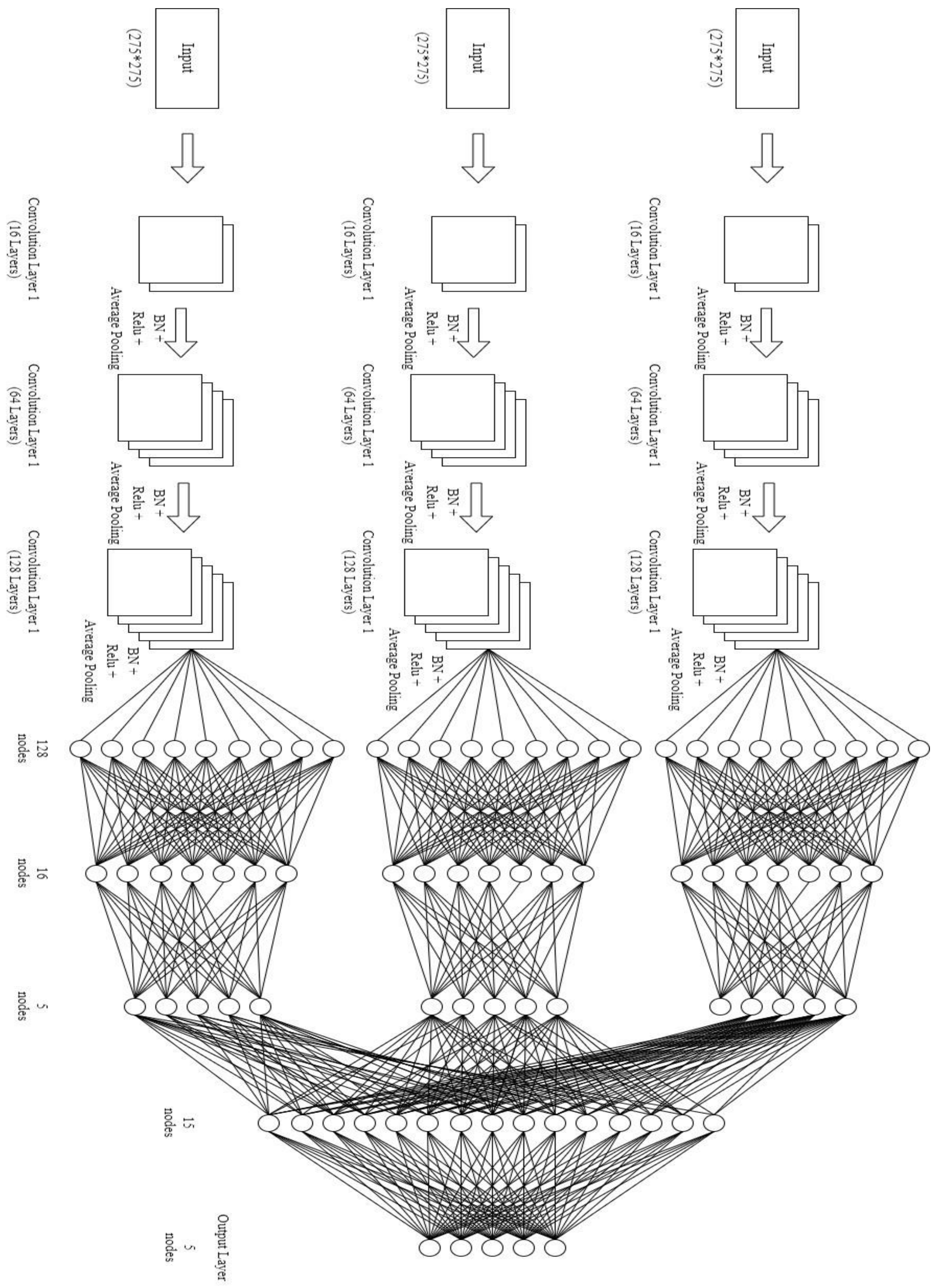


Fig 3.2.3 CNN Model

3.2.4 Convolution

A convolutional layer is the main part of an CNN or a multilayer perceptron. The convolution in the CNN is done by the convolutional layer. There are mainly three layers in a CNN called the input layer, output layer and the convolutional layer. This process starts with a filter or a kernel that slides over the input data. This performs multiplication of the slide part and the kernel and summing up the result into a single output pixel. The kernel repeats this process for every location of the input data. This layer allows us to do transformation with each output feature, instead of looking at every input feature [12].

3.2.5 Global average pool

Pooling is mainly responsible for reducing the spatial size of the convolved feature. This is done to decrease the computational power required to process the data through reduction in the dimensions. It is also useful for extracting dominant feature which are rotational and positional invariant. There are two types of pooling methods called max pooling and average pooling. In this, average pooling is done. In this average pooling, it returns the average of all the values from the portion of the image covered by the kernel. Thus, this maintains the process of effective training of the model [12].

3.3 Algorithm

Convolutional Neural Network

In this project, we will be using the CNN (Convolutional Neural Network) for the image and video recognition/analysis. In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing. A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a sliding dot product or cross-correlation. This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point [11].

Global Average Pooling

It is a pooling operation designed to replace fully connected layers in classical CNNs. The idea is to generate one feature map for each corresponding category of the classification task in the last mlpconv layer. Instead of adding fully connected layers on top of the feature maps, we take the average of each feature map, and the resulting vector is fed directly into the softmax layer.

One advantage of global average pooling over the fully connected layers is that it is more native to the convolution structure by enforcing correspondences between feature maps and categories. Thus, the feature maps can be easily interpreted as categories confidence maps. Another advantage is that there is no parameter to optimize in the global average pooling thus over fitting is avoided at this layer. Furthermore, global average pooling sums out the spatial information, thus it is more robust to spatial translations of the input [12].

3.4 Flowchart

This flowchart shows the steps involved in this project. They are explained as below:

1. At first, the CT images are collected and input to the system. The images are in the DICOM format initially.

2. After receiving the images, preprocessing is done on the images. The different steps applied during the preprocessing are masking, padding, noise reduction, etc.

3. From the preprocessed data, a sample library is created.

4. Training and test dataset is created from the library of the images. The ratio of training dataset to test dataset was 8:2.

5. A CNN model was then built, The CNN was multilayered with blood, bone, and brain layer. The multilayered CNN were then concatenated through another CNN finally.

6. The training dataset was fed to the model and a trained model are prepared.

7. The test data was then used for the validation of the model. Validation accuracy was obtained to determine the accuracy of the model.

8. After preparing the model, the user input the CT image to the system for prediction.

9. The trained model then finally gets the CT image, preprocesses it, and predicts the type of hemorrhage present in the CT image.

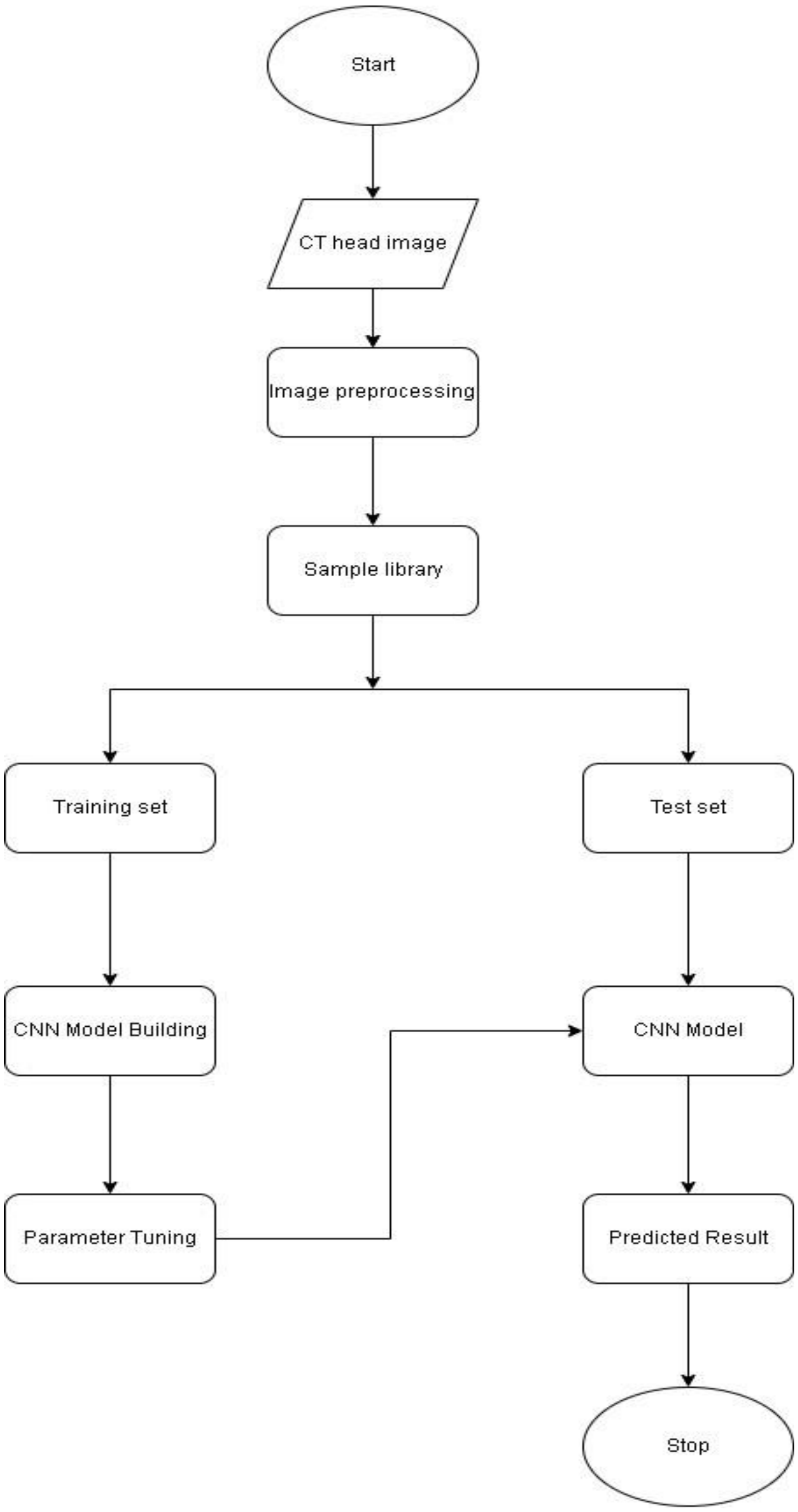


Fig 3.4 Flowchart

3.5 UML Diagrams

3.5.1 Use Case Diagram

This is a UML diagram that shows the relation between actors, real world entities and the whole system. The purpose of use case diagram is to extract the dynamic purpose of the system. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified. Different steps involved in this use case diagram are explained below:

1. User can upload CT images to the system. The CT images are in dcm format.

2. The image uploaded by the user are in dcm format and is the preprocessed, fed to the model.

3. The CNN model is then used to predict what kind of intracranial hemorrhage is present in the CT image.

4. Finally, the result of the prediction is shown to the user through user interface.

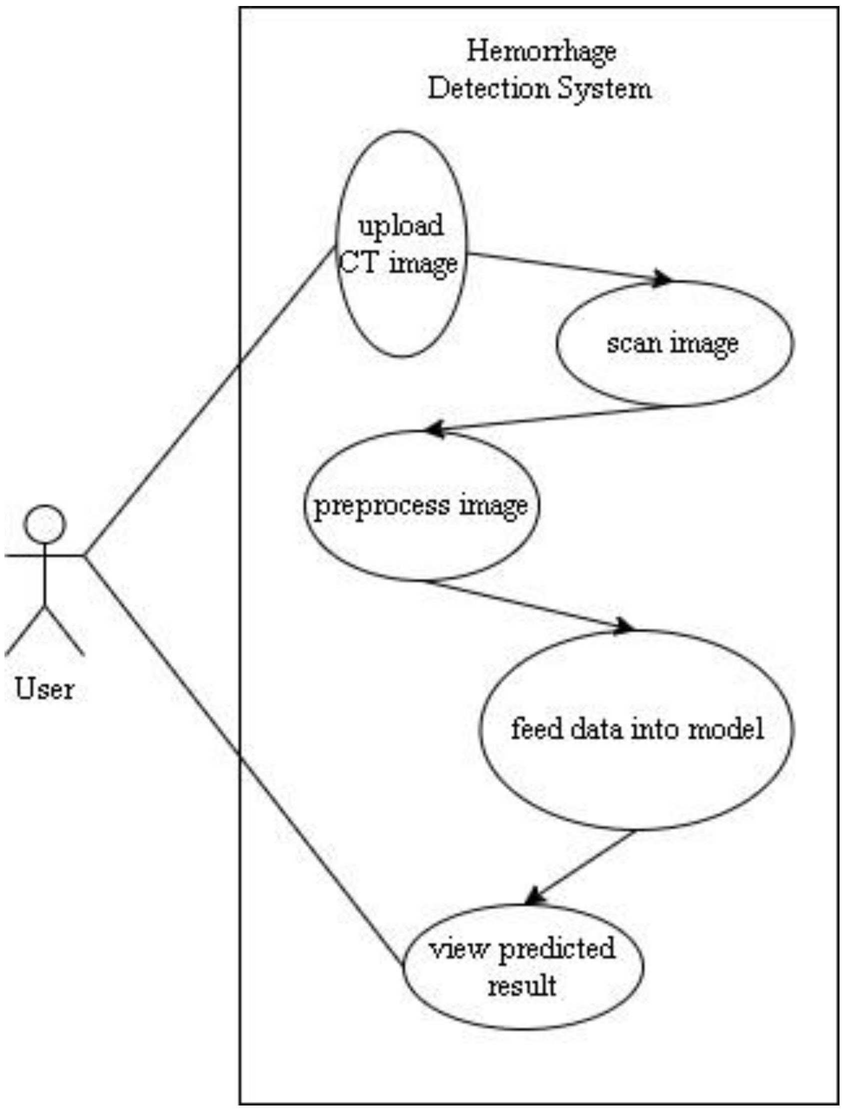


Fig 3.5.1 Use Case Diagram

3.5.2 DFD Level 0

This is the Data flow diagram level 0 of this system. This shows the flow of data between all the entities at the lowest level of implementation. These are the two main entities involved in the system. Here, the use is responsible to input data to the system as DICOM images of the CT images. And the System is responsible to preprocess the image, and finally predict the type of hemorrhage using the CNN model.

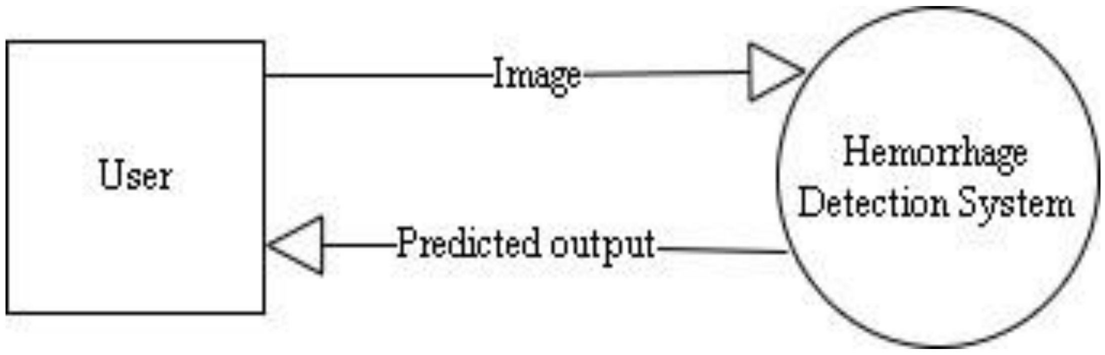


Fig 3.5.2 DFD Level 0

3.5.3 DFD Level 1

This Data flow diagram shows the flow of data between entities at a level higher than that of DFD level 0. Here, the involved entities are User and the whole system. CT images are an important element of the system. User are responsible to provide CT images as an input to the system. Image preprocessing is another important element of this whole system. After the image is uploaded by the user, it is preprocessed by the system and fed into the model. The CNN model is responsible to take the data and predict the type of hemorrhage in the CT image.

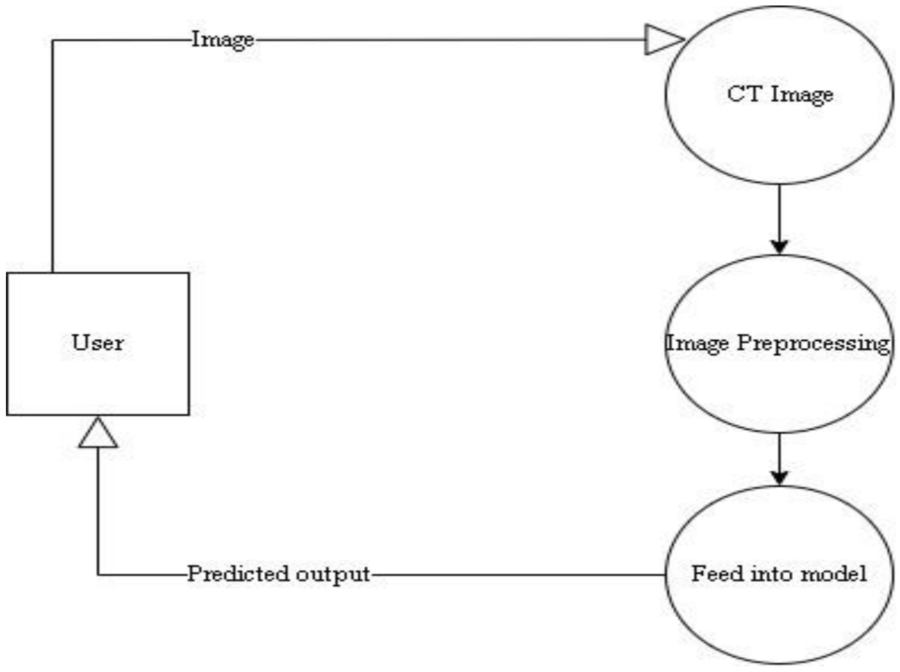


Fig 3.5.3 DFD Level 1

3.6 Tools Used

Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. It supports multiple programming paradigms, including structured (particularly, procedural), object oriented and functional programming. Python interpreters are supported for mainstream operating systems and available for a few more (and in the past supported many more). A global community of programmers develops and maintains CPython, a free and open-source reference implementation [13].

Pydicom

Pydicom is the python library to manipulate dicom images. Dicom is the international standard for medical images and related information. It defines the formats for medical images that can be exchanged with the data and quality necessary for clinical use. It is implemented in almost every radiology, cardiology imaging, and radiotherapy device (X-ray, CT, MRI, ultrasound, etc.), and increasingly in devices in other medical domains such as ophthalmology and dentistry [14].

NumPy

NumPy is a Python library that provides a simple yet powerful data structure: the n-dimensional array. This is the foundation on which almost all the power of Python's data science toolkit is built, and learning NumPy is the first step on any Python data scientist's journey. NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multidimensional container of generic data. Arbitrary datatypes can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases [15].

Vscode

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. It comes with builtin support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity) [16].

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery [17].

Tensorflow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Tensorflow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google. TensorFlow offers multiple levels of abstraction so one can choose the right one needs [18].

Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL database tables or queries, and Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features [19].

Keras

Keras is a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research. It allows for easy and fast prototyping (through total modularity, minimalism, and extensibility). It supports both convolutional networks and recurrent networks, as well as combinations of the two. It supports arbitrary connectivity schemes (including multi-input and multi-output training). It runs seamlessly on CPU and GPU [20].

SciPy

SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering. The basic data structure used by SciPy is a multidimensional array provided by the NumPy module [21].

3.7 Verification and Validation

3.7.1 Model Performance for Testing Data

Total datasets used = 15000

From 15000 dataset, 12000(80%) data were used for training purpose and 3000(20%) were used for testing purpose.

Training Accuracy = 63.37

Validation Accuracy = 50.34

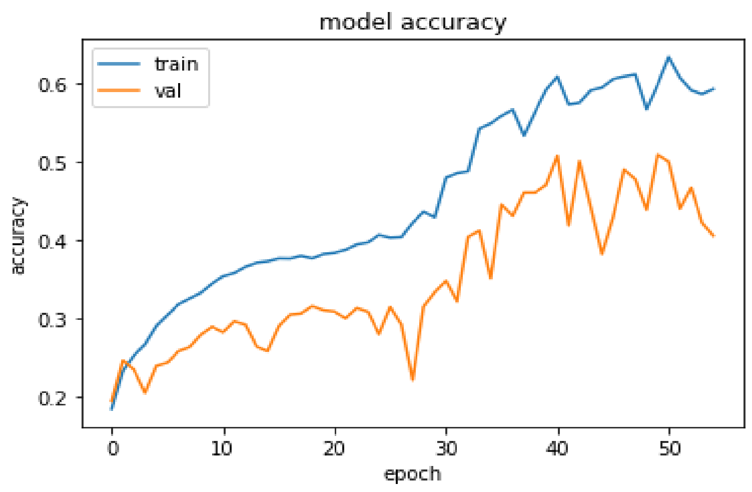


Fig 3.7.1 Training and validation accuracy

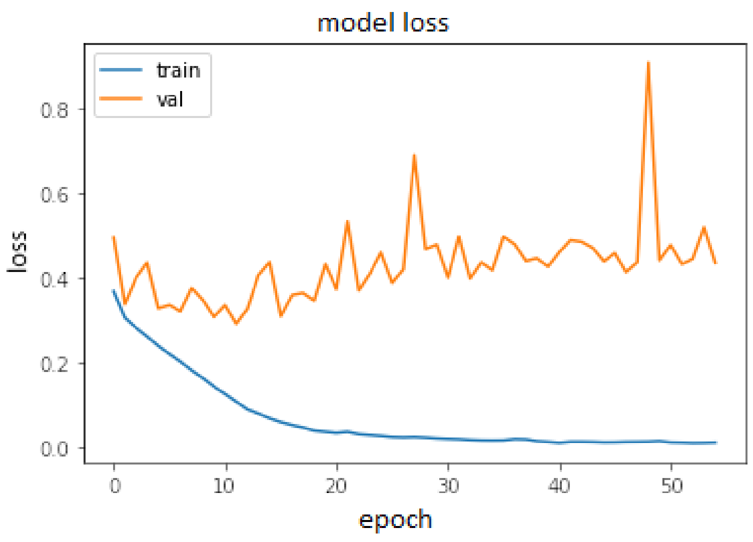


Fig 3.7.2 Training and validation loss

CHAPTER 4: RESULT AND DISCUSSION

Therefore, we were able to successfully create a program that carries out the following tasks:

1. Retrieves DICOM CT-Scan Images from the user.
2. Process and Convert the image to required format before feeding into the model.
3. Predict whether any of the 5 sub types of ICH is detected or not.

Upon final discussion, our program was built using CNN which was trained with 15,000 training datasets. We had a complex task of pre-processing our dataset which also involved applying different windows to our dataset before training phase. Due to resource limitation, we chose an optimum of 3 windows. By training our model for 11 epochs, it yielded us an accuracy of 24.25%. Upon further training the model with a higher specification GPU for 55 epochs, it yielded us a final model accuracy of 63.37%.

Future Enhancement

We aim to implement the following enhancements in future:

- Create a lightweight and portable web app which can be hosted on Cloud
- Applying newer algorithms for increased accuracy
- Increment of dataset

CHAPTER 5: CONCLUSION

Hence in conclusion, this system is inadequate to use in real-world application, because of its low accuracy. Due to the critical nature of the condition, we must be able to achieve higher accuracy and detect the correct type of hemorrhage on top of it for the system to be adequately accepted.

Furthermore, the model can be further improved by using more efficient algorithms, optimizing the network, and adding more hidden layers, feeding more dataset etc. to increase its accuracy.

REFERENCES

[1] Arjun Majumdar, Laura Brattain, Brian Telfer, Chad Farris, Jonathan Scalera, "Detecting Intracranial Hemorrhage with Deep Learning," July 2018. [Online]. Available:
https://pubmed.ncbi.nlm.nih.gov/30440464/.

[2] Helwan, Abdulkader, Georges El Fakhri, Hadi Sasani and Dilber Uzun Ozsahin, "Deep networks in identifying CT brain hemorrhage," *Journal of Intelligent & Fuzzy Systems,* vol. 35, no. 2, pp. 2215-2228, 2018.

[3] R. M. a. P. M. Mahajan, "Survey on diagnosis of brain haemorrhage by using artificial neural network," *International Journal of Scientific Research Engineering & Technology,* vol. 5, no. 6, pp. 378-381, 2016.

[4] T. Gong, Ruizhe Liu,, "Classification of CT Brain Images of Head Trauma," in *PRIB,* Melbourne, 2007.

[5] Awwal Muhammad Dawud, Kamil Yurtkan, Huseyin Oztoprak, "Application of Deep
Learning in Neuroradiology: Brain Haemorrhage Classification Using Transfer Learning," *Computational Intelligence and Neuroscience,* vol. 2019, p. 12, 2019.

[6] Weicheng Kuo, Christian Häne, Pratik Mukherjee, Jitendra Malik, and Esther L. Yuh, "Expert-level detection of acute intracranial hemorrhage on head computed tomography using deep learning," *National Academy of Sciences,* vol. 116, no. 45, pp. 22737-22745, 2019.

[7] Barry McDermott, Martin O'Halloran, Emily Porter and Adam Santorelli, "Brain haemorrhage detection using a SVM classifier with electrical impedance tomography measurement frames," *PLOS ONE,* vol. 7, p. 13, 2018.

[8] Hang Chen, Sulaiman Khan , Bo Kou, Shah Nazir , Wei Liu , and Anwar Hussain , "A Smart Machine Learning Model for the Detection of Brain Hemorrhage Diagnosis Based Internet of Things in Smart Cities," *Complexity,* p. 10, 2020.

[9] D. BHATTACHARJYA, "Brain Hemorrhage Detection Using Machine Learning," West Bengal, 2020.

[10] Nivedhitha P, Dr Sankar S, "HEMORRHAGE DETECTION SYSTEM USING WATERSHED SEGMENTATION," *International Journal of Applied Engineering Research,* vol. 14, p. 5, 2019.

[11] Prabhu, "Understanding of Convolutional Neural Network (CNN) — Deep Learning," medium, 4 march 2018. [Online]. Available:

https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-networkcnn-deep-learning-99760835f148.

[12] "convolutions-for-deep-learning," towardsdatascience, [Online]. Available:

https://towardsdatascience.com/intuitively-understanding- convolutions-for-deeplearning-1f6f42faee1..

[13] "Python," [Online]. Available: https://www.python.org/doc/.

[14] "About DICOM: Overview," [Online]. Available: https://www.dicomstandard.org/about.

[15] "What is Numpy in Python | Python Numpy Tutorial," Great Learning Team, 11 january 2022. [Online]. Available: https://www.mygreatlearning.com/blog/python-numpytutorial/#:~:text=NumPy%2C%20which%20stands%20for%20Numerical,stands%20for %20'Numerical%20Python'..

[16] "Visual Studio Code," [Online]. Available: https://code.visualstudio.com/docs.

[17] "matplotlib," [Online]. Available: https://matplotlib.org/.

[18] "TensorFlow," [Online]. Available: https://www.tensorflow.org/resources/learn-ml?gclid=CjwKCAiAvaGRBhBlEiwAiY-yMFTMlC-XzXHIbIGpbJTxUBf68DOsK6KrzC71iI1Qp3RtXRgaiSnA_xoCGlEQAvD_BwE.

[19] pandas," [Online]. Available: https://pandas.pydata.org/.

[20] EmmanuelleRieuf, "Keras: Deep Learning library for Theano and TensorFlow," 2017.

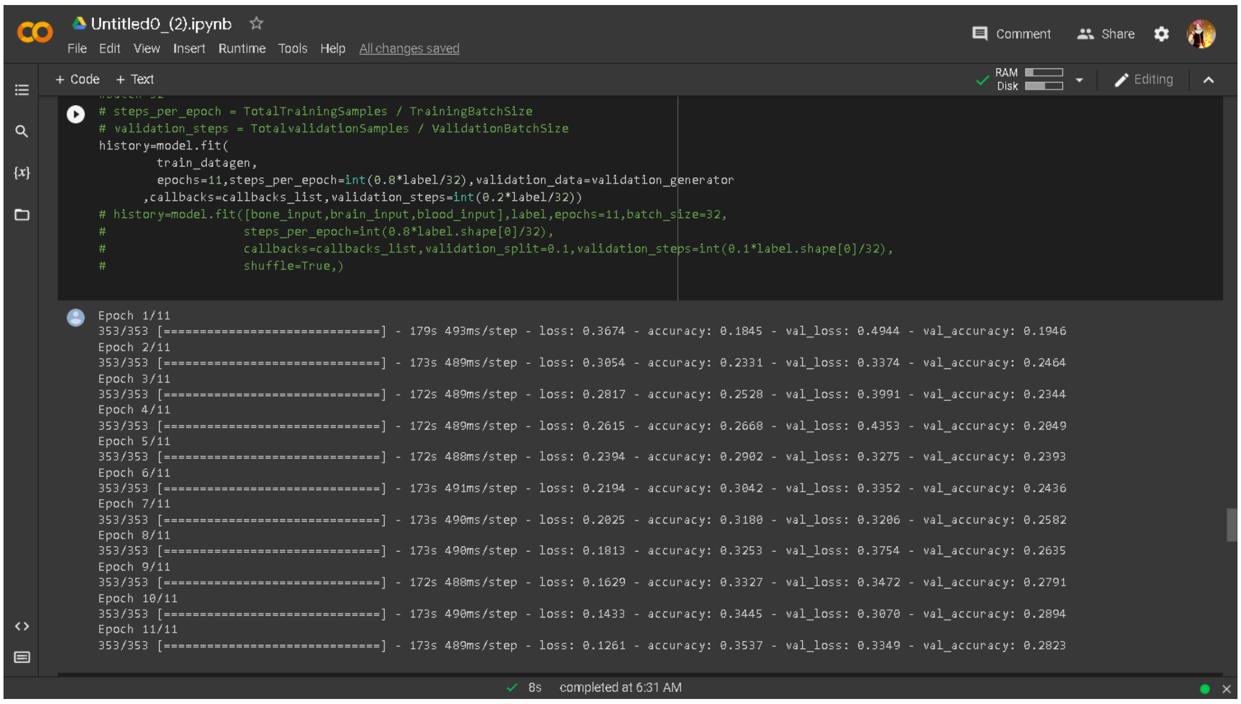
[21] "SciPy," [Online]. Available:

https://scipy.org/.

[22] "seaborn," [Online]. Available: https://seaborn.pydata.org/. [Accessed December 2021].

ANNEX

Screenshots

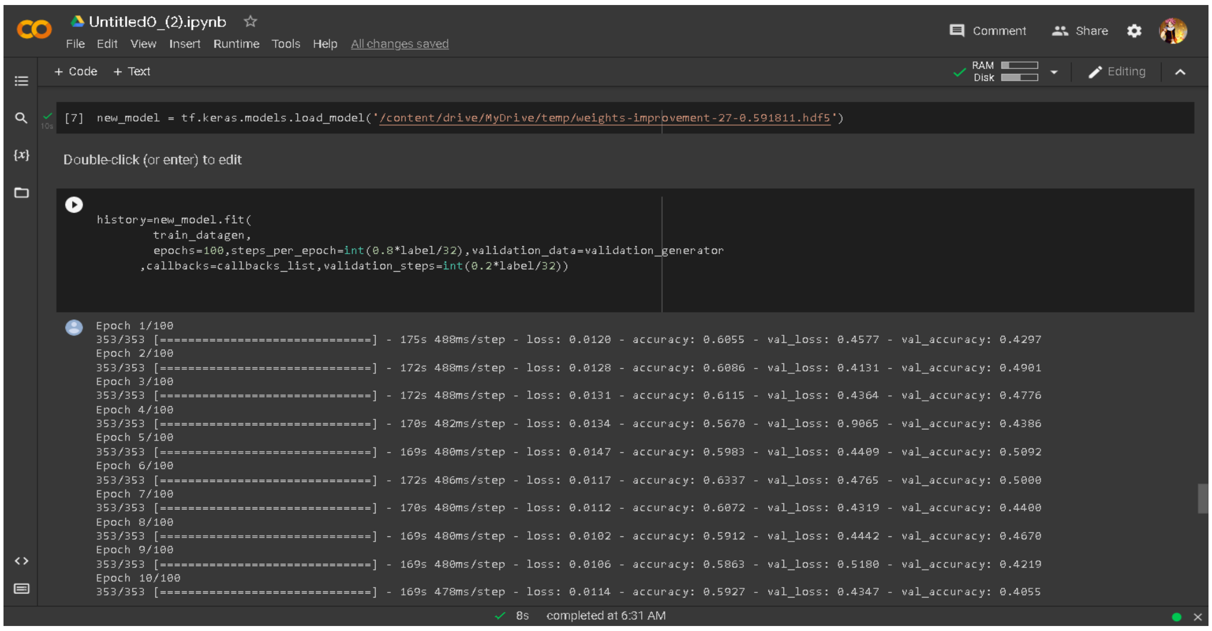


```
# steps_per_epoch = TotalTrainingSamples / TrainingBatchSize
# validation_steps = TotalvalidationSamples / ValidationBatchSize
history=model.fit(
        train_datagen,
        epochs=11,steps_per_epoch=int(0.8*label/32),validation_data=validation_generator
      ,callbacks=callbacks_list,validation_steps=int(0.2*label/32))
# history=model.fit([bone_input,brain_input,blood_input],label,epochs=11,batch_size=32,
#                   steps_per_epoch=int(0.8*label.shape[0]/32),
#                   callbacks=callbacks_list,validation_split=0.1,validation_steps=int(0.1*label.shape[0]/32),
#                   shuffle=True,)
```

```
Epoch 1/11
353/353 [==============================] - 179s 493ms/step - loss: 0.3674 - accuracy: 0.1845 - val_loss: 0.4944 - val_accuracy: 0.1946
Epoch 2/11
353/353 [==============================] - 173s 489ms/step - loss: 0.3054 - accuracy: 0.2331 - val_loss: 0.3374 - val_accuracy: 0.2464
Epoch 3/11
353/353 [==============================] - 172s 489ms/step - loss: 0.2817 - accuracy: 0.2528 - val_loss: 0.3991 - val_accuracy: 0.2344
Epoch 4/11
353/353 [==============================] - 172s 489ms/step - loss: 0.2615 - accuracy: 0.2668 - val_loss: 0.4353 - val_accuracy: 0.2049
Epoch 5/11
353/353 [==============================] - 172s 488ms/step - loss: 0.2394 - accuracy: 0.2902 - val_loss: 0.3275 - val_accuracy: 0.2393
Epoch 6/11
353/353 [==============================] - 173s 491ms/step - loss: 0.2194 - accuracy: 0.3042 - val_loss: 0.3352 - val_accuracy: 0.2436
Epoch 7/11
353/353 [==============================] - 173s 490ms/step - loss: 0.2025 - accuracy: 0.3180 - val_loss: 0.3206 - val_accuracy: 0.2582
Epoch 8/11
353/353 [==============================] - 173s 490ms/step - loss: 0.1813 - accuracy: 0.3253 - val_loss: 0.3754 - val_accuracy: 0.2635
Epoch 9/11
353/353 [==============================] - 172s 488ms/step - loss: 0.1629 - accuracy: 0.3327 - val_loss: 0.3472 - val_accuracy: 0.2791
Epoch 10/11
353/353 [==============================] - 173s 490ms/step - loss: 0.1433 - accuracy: 0.3445 - val_loss: 0.3070 - val_accuracy: 0.2894
Epoch 11/11
353/353 [==============================] - 173s 489ms/step - loss: 0.1261 - accuracy: 0.3537 - val_loss: 0.3349 - val_accuracy: 0.2823
```

✓ 8s completed at 6:31 AM

Training Neural Network



```
[7]  new_model = tf.keras.models.load_model('/content/drive/MyDrive/temp/weights-improvement-27-0.591811.hdf5')
```

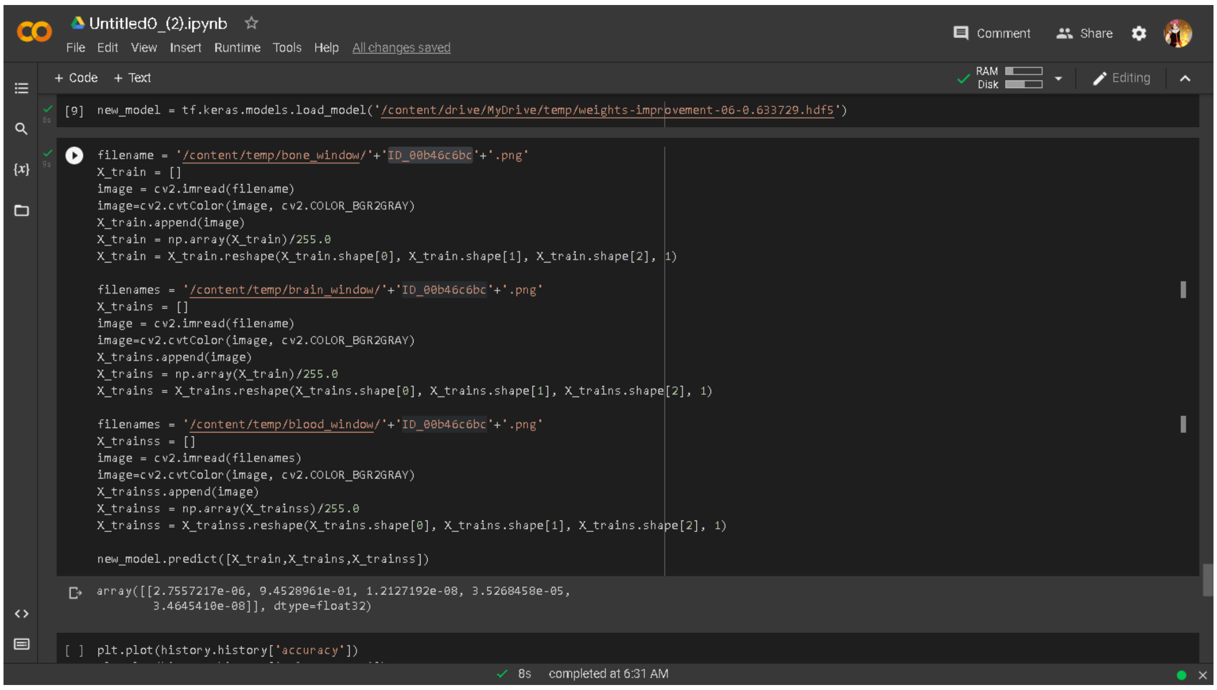
Double-click (or enter) to edit

```
history=new_model.fit(
        train_datagen,
        epochs=100,steps_per_epoch=int(0.8*label/32),validation_data=validation_generator
      ,callbacks=callbacks_list,validation_steps=int(0.2*label/32))
```

```
Epoch 1/100
353/353 [==============================] - 175s 488ms/step - loss: 0.0120 - accuracy: 0.6055 - val_loss: 0.4577 - val_accuracy: 0.4297
Epoch 2/100
353/353 [==============================] - 172s 488ms/step - loss: 0.0128 - accuracy: 0.6086 - val_loss: 0.4131 - val_accuracy: 0.4901
Epoch 3/100
353/353 [==============================] - 172s 488ms/step - loss: 0.0131 - accuracy: 0.6115 - val_loss: 0.4364 - val_accuracy: 0.4776
Epoch 4/100
353/353 [==============================] - 170s 482ms/step - loss: 0.0134 - accuracy: 0.5670 - val_loss: 0.9065 - val_accuracy: 0.4386
Epoch 5/100
353/353 [==============================] - 169s 480ms/step - loss: 0.0147 - accuracy: 0.5983 - val_loss: 0.4409 - val_accuracy: 0.5092
Epoch 6/100
353/353 [==============================] - 172s 486ms/step - loss: 0.0117 - accuracy: 0.6337 - val_loss: 0.4765 - val_accuracy: 0.5000
Epoch 7/100
353/353 [==============================] - 170s 480ms/step - loss: 0.0112 - accuracy: 0.6072 - val_loss: 0.4319 - val_accuracy: 0.4400
Epoch 8/100
353/353 [==============================] - 169s 480ms/step - loss: 0.0102 - accuracy: 0.5912 - val_loss: 0.4442 - val_accuracy: 0.4670
Epoch 9/100
353/353 [==============================] - 169s 480ms/step - loss: 0.0106 - accuracy: 0.5863 - val_loss: 0.5180 - val_accuracy: 0.4219
Epoch 10/100
353/353 [==============================] - 169s 478ms/step - loss: 0.0114 - accuracy: 0.5927 - val_loss: 0.4347 - val_accuracy: 0.4055
```

✓ 8s completed at 6:31 AM

Loading checkpoint and training



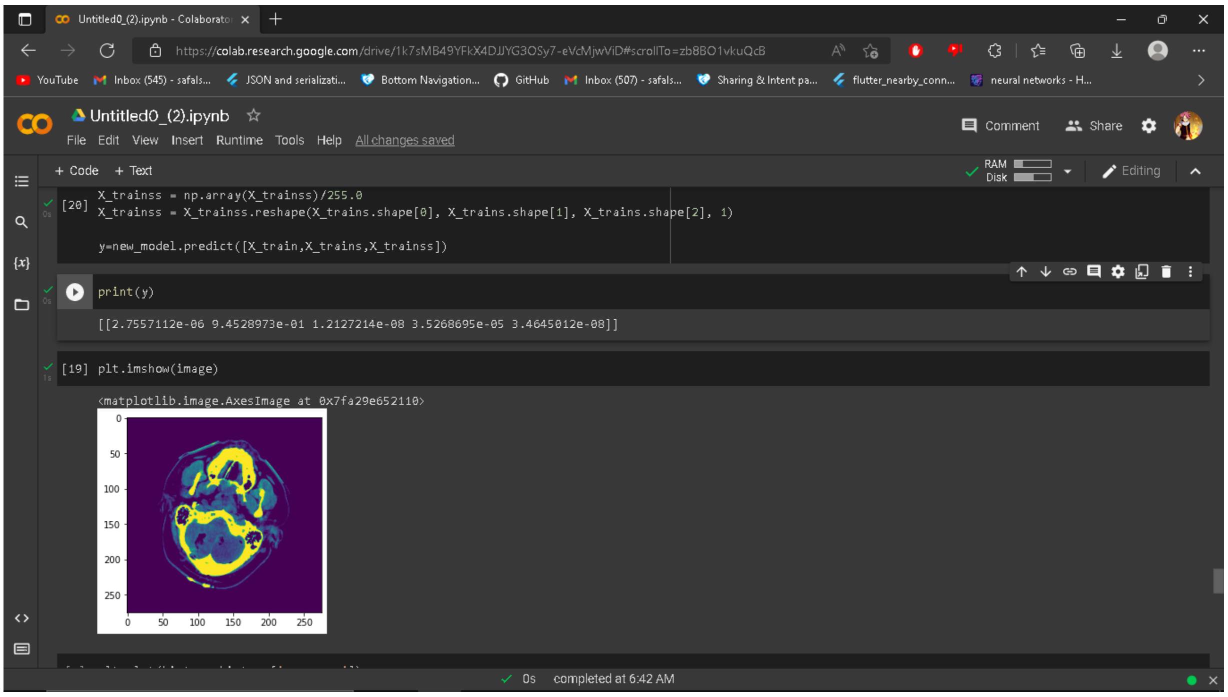
```
[9] new_model = tf.keras.models.load_model('/content/drive/MyDrive/temp/weights-improvement-06-0.633729.hdf5')
```

```
filename = '/content/temp/bone_window/'+'ID_00b46c6bc'+'.png'
X_train = []
image = cv2.imread(filename)
image=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
X_train.append(image)
X_train = np.array(X_train)/255.0
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2], 1)

filenames = '/content/temp/brain_window/'+'ID_00b46c6bc'+'.png'
X_trains = []
image = cv2.imread(filename)
image=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
X_trains.append(image)
X_trains = np.array(X_train)/255.0
X_trains = X_trains.reshape(X_trains.shape[0], X_trains.shape[1], X_trains.shape[2], 1)

filenames = '/content/temp/blood_window/'+'ID_00b46c6bc'+'.png'
X_trainss = []
image = cv2.imread(filenames)
image=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
X_trainss.append(image)
X_trainss = np.array(X_trainss)/255.0
X_trainss = X_trainss.reshape(X_trains.shape[0], X_trains.shape[1], X_trains.shape[2], 1)

new_model.predict([X_train,X_trains,X_trainss])
```

```
array([[2.7557217e-06, 9.4528961e-01, 1.2127192e-08, 3.5268458e-05,
        3.4645410e-08]], dtype=float32)
```

```
[ ] plt.plot(history.history['accuracy'])
```

Prediction



```
[20] X_trainss = np.array(X_trainss)/255.0
     X_trainss = X_trainss.reshape(X_trains.shape[0], X_trains.shape[1], X_trains.shape[2], 1)

     y=new_model.predict([X_train,X_trains,X_trainss])
```

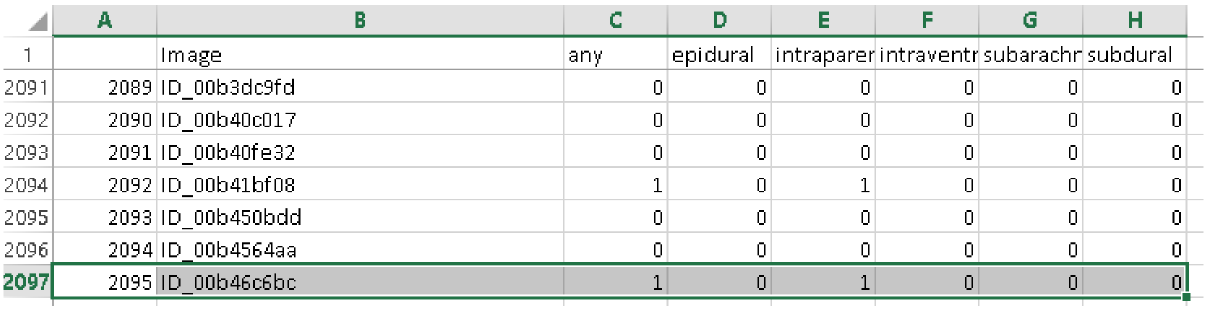
```
print(y)
```

```
[[2.7557112e-06 9.4528973e-01 1.2127214e-08 3.5268695e-05 3.4645012e-08]]
```

```
[19] plt.imshow(image)
```

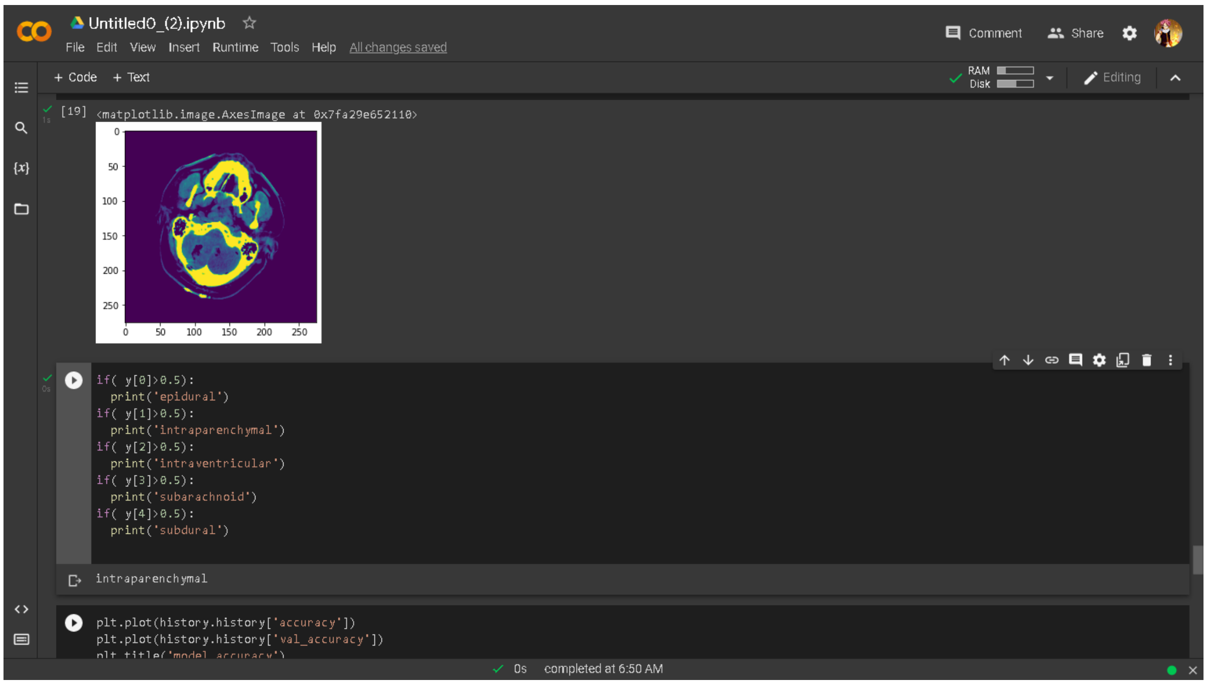
```
<matplotlib.image.AxesImage at 0x7fa29e652110>
```

Image being predicted



	A	B	C	D	E	F	G	H
1		Image	any	epidural	intraparer	intraventr	subarachr	subdural
2091	2089	ID_00b3dc9fd	0	0	0	0	0	0
2092	2090	ID_00b40c017	0	0	0	0	0	0
2093	2091	ID_00b40fe32	0	0	0	0	0	0
2094	2092	ID_00b41bf08	1	0	1	0	0	0
2095	2093	ID_00b450bdd	0	0	0	0	0	0
2096	2094	ID_00b4564aa	0	0	0	0	0	0
2097	2095	ID_00b46c6bc	1	0	1	0	0	0

Snapshot of CSV file



```
[19] <matplotlib.image.AxesImage at 0x7fa29e652110>
```

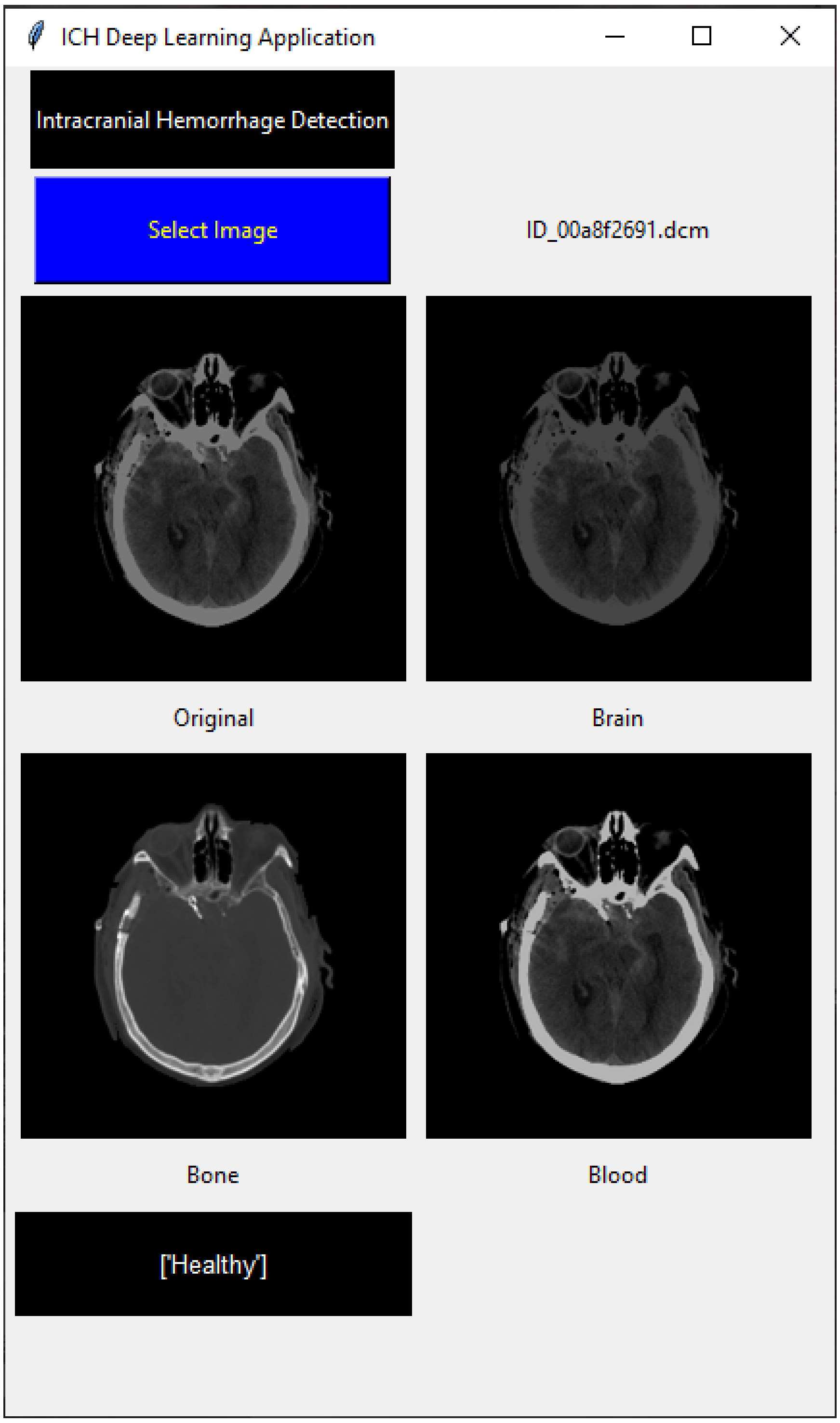
```
if( y[0]>0.5):
  print('epidural')
if( y[1]>0.5):
  print('intraparenchymal')
if( y[2]>0.5):
  print('intraventricular')
if( y[3]>0.5):
  print('subarachnoid')
if( y[4]>0.5):
  print('subdural')
```

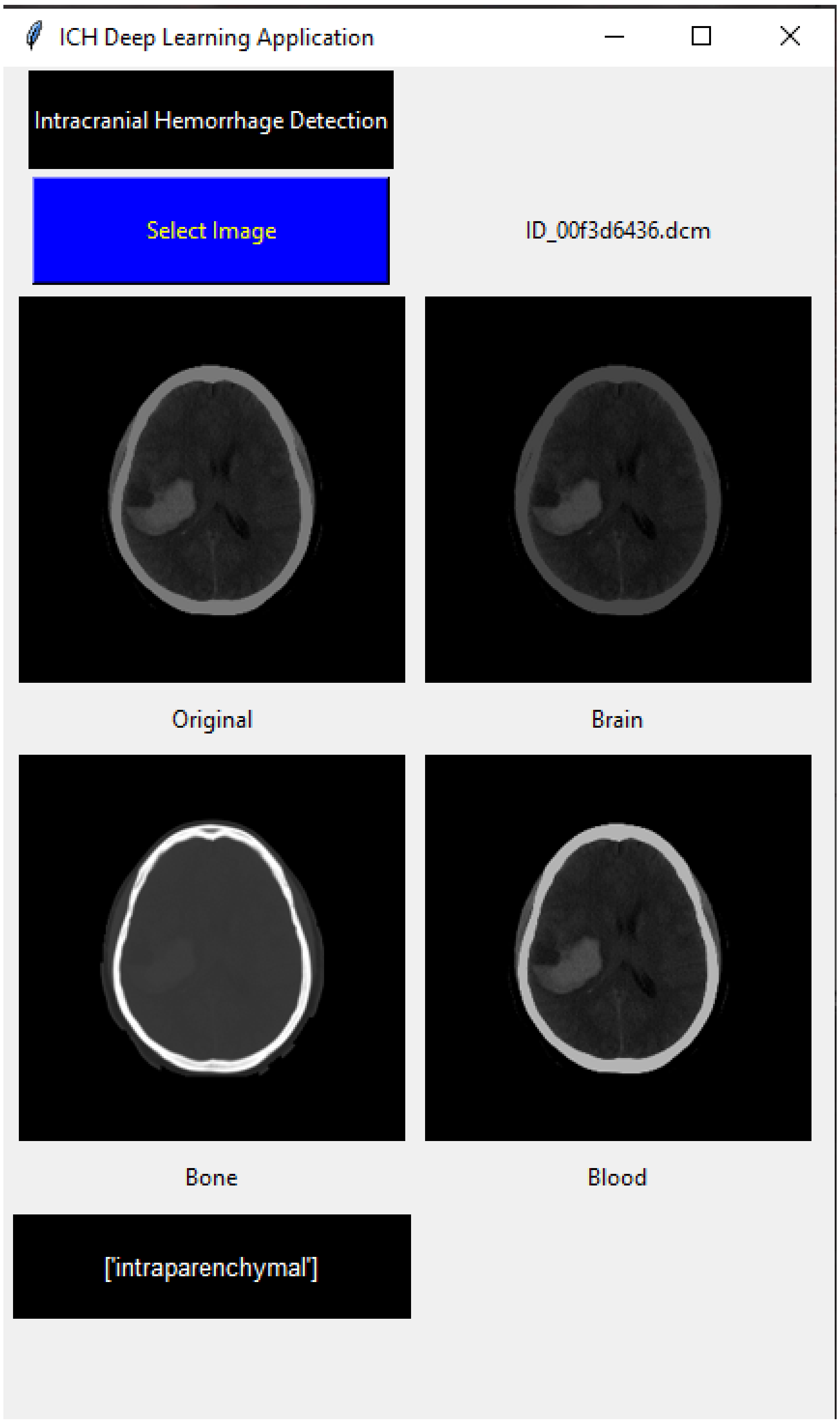
```
intraparenchymal
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
```

```
0s  completed at 6:50 AM
```

Predicted output

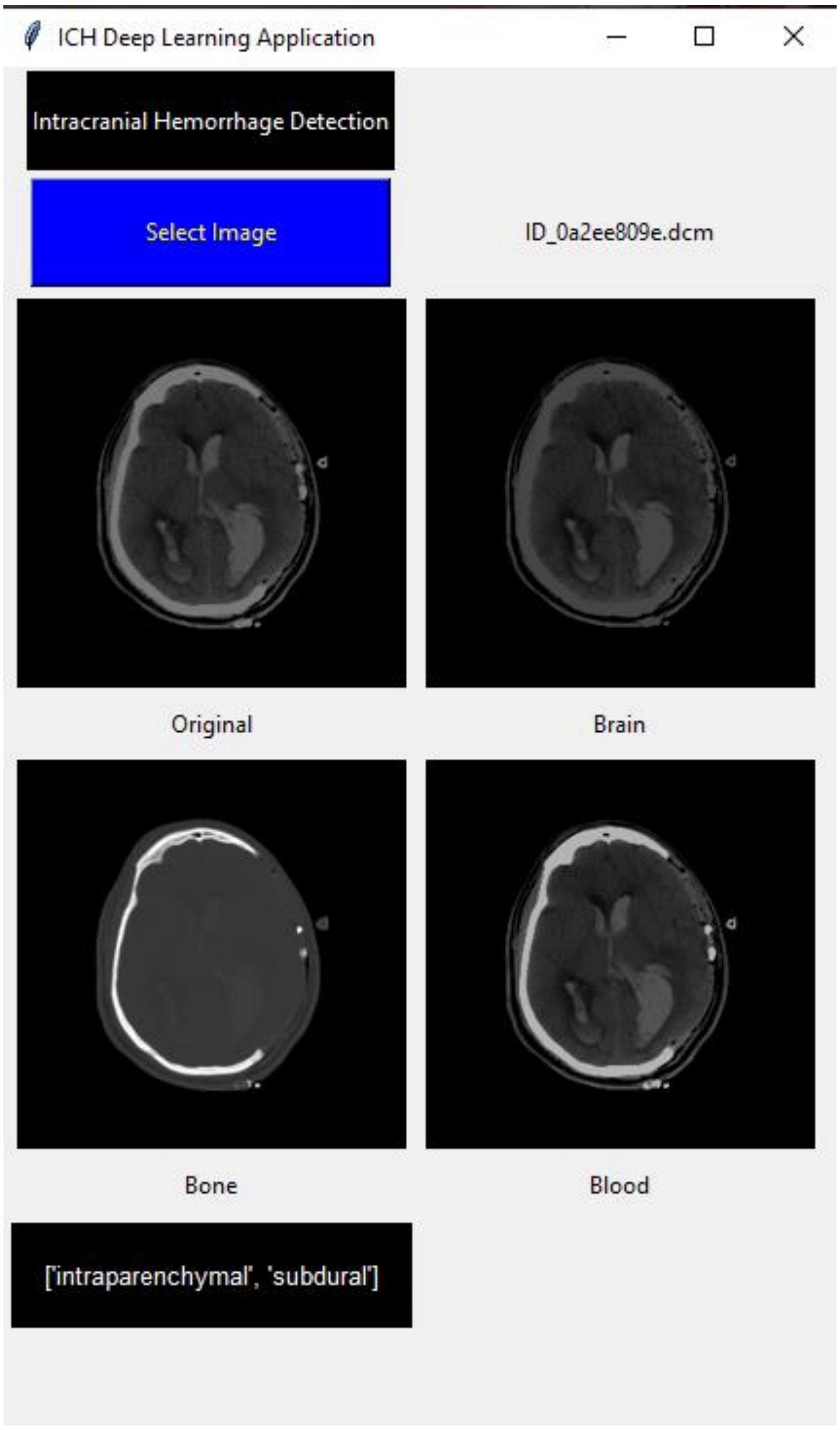

ICH Deep Learning Application
Intracranial Hemorrhage Detection
Select Image
ID_00a8f2691.dcm
Original
Brain
Bone
Blood
['Healthy']


ICH Deep Learning Application

Intracranial Hemorrhage Detection

ID_00f3d6436.dcm

['intraparenchymal']


ICH Deep Learning Application
Intracranial Hemorrhage Detection
Select Image
Original
Brain
Bone
Blood

ID_0a2ee809e.dcm

['intraparenchymal', 'subdural']