

# **INTRACRANIAL HEMORRHAGE DETECTION**

## **A PROJECT REPORT**

*Submitted by*

**SHASHANK K. SINGH [Reg No: RA1611003030240]**

*Under the guidance of*

**Dr. MANIKANDAN R.**

(Assistant Professor, Department of Computer Science & Engineering)

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE ENGINEERING**

of

**FACULTY OF ENGINEERING AND TECHNOLOGY**



**SRM**

**INSTITUTE OF SCIENCE & TECHNOLOGY**  
(Deemed to be University u/s 3 of UGC Act, 1956)

Delhi NCR Campus, Modinagar, Ghaziabad (U.P)

**JUNE 2020**

# SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

## BONAFIDE CERTIFICATE

Certified that this project report titled “**INTRACRANIAL HEM-ORRHAGE DETECTION**” is the bonafide work of “**SHASHANK K. SINGH [Reg No: RA1611003030240]**”, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

Dr. MANIKANDAN R.  
**GUIDE**  
Assistant Professor  
Dept. of Computer Science & Engineering

Signature of the Internal Examiner

**SIGNATURE**

Dr. R. P. MAHAPATRA  
**HEAD OF THE DEPARTMENT**  
Dept. of Computer Science Engineering

Signature of the External Examiner

## **ABSTRACT**

Intracranial Hemorrhage (ICH), bleeding that occurs inside the cranium, is a serious health problem requiring rapid and often intensive medical treatment. Identifying the location and type of any hemorrhage present is a critical step in treating the patient.

Diagnosis requires an urgent procedure. When a patient shows acute neurological symptoms such as severe headache or loss of consciousness, highly trained specialists review medical images of the patient's cranium to look for the presence, location and type of hemorrhage. The process is complicated and often time consuming.

In this paper, it is treated as a multilabel classification problem. Five different types of hemorrhages are detected namely Epidural (EPD), Intraparenchymal (ITP), Intraventricular (ITV), Subarachnoid (SBC) and Subdural (SBD). It's possible for a patient to suffer from multiple hemorrhages at a time. Given the CT scan of a patient's cranium, the task is to detect and classify any hemorrhage(s) that might be present. First, detect whether or not there is any hemorrhage present at all, then try to identify the type of hemorrhage(s) present.

Two different Deep Learning approaches are utilized as possible solutions. One uses Systematic Windowing with an RNN and other uses a CNN without any windowing. This paper explains the implementation of both techniques and at the end presents the CNN based model's performance.

## **ACKNOWLEDGEMENTS**

I would like to express my deepest gratitude to my guide, Dr. Manikandan R. for his valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this research work.

**Shashank Kumar Singh**

# TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>iv</b>
<b>LIST OF TABLES</b>	<b>vii</b>
<b>LIST OF FIGURES</b>	<b>viii</b>
<b>ABBREVIATIONS</b>	<b>ix</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Research motivation and previous work . . . . .	1
1.2 Types of hemorrhages . . . . .	1
1.3 Model Input/Output . . . . .	2
<b>2 LITERATURE SURVEY</b>	<b>4</b>
2.1 See like a radiologist with Systematic windowing! . . . . .	4
2.2 DON'T see like a radiologist! (fastai) . . . . .	6
<b>3 SYSTEM ANALYSIS</b>	<b>7</b>
3.1 RSNA Intracranial Hemorrhage Detection . . . . .	7
3.2 Linear Transformation, Windowing and Min-Max scaling . . . . .	7
3.3 Weighted samples . . . . .	8
3.4 Supervised learning problem . . . . .	10
3.5 Selection of appropriate loss function . . . . .	10
3.6 Model specific details . . . . .	11
<b>4 SYSTEM DESIGN</b>	<b>12</b>
4.1 Deep RNN Model . . . . .	12
4.2 CNN Model . . . . .	12

<b>5</b>	<b>EXPERIMENT RESULTS</b>	<b>18</b>
5.1	Coding . . . . .	18
5.2	Testing . . . . .	21
5.2.1	Validation Results . . . . .	21
5.2.2	Test Results . . . . .	22
<b>6</b>	<b>CONCLUSION</b>	<b>23</b>
<b>7</b>	<b>FUTURE ENHANCEMENTS</b>	<b>24</b>

## LIST OF TABLES

2.1	Different Window settings. . . . .	4
3.1	Types of images with their respective sample count and the hemorrhages present in them . . . . .	9

## LIST OF FIGURES

1.1	<b>Types of hemorrhages and their respective location inside the cranium. . . . .</b>	2
1.2	<b>Illustration showing example of Input / Output for the Neural Networks. . . . .</b>	3
2.1	<b>DICOM Image viewed in 10 different window settings. . . . .</b>	5
2.2	<b>DICOM file under different window settings creating a temporal sequence. . . . .</b>	5
4.1	<b>RNN Model for ICH Detection. . . . .</b>	12
4.2	<b>Unit 1 of CNN Model. . . . .</b>	13
4.3	<b>Unit 2 of CNN Model. . . . .</b>	14
4.4	<b>Unit 3 of CNN Model. . . . .</b>	15
4.5	<b>Unit 4 of CNN Model. . . . .</b>	15
4.6	<b>Unit 5 of CNN Model. . . . .</b>	16
4.7	<b>Unit 6 of CNN Model. . . . .</b>	16
4.8	<b>Unit 7 of CNN Model. . . . .</b>	17
5.1	<b>Sensitivity and fall-out of ICH on validation set. . . . .</b>	21
5.2	<b>Specificity and miss-rate of ICH on validation set. . . . .</b>	21
5.3	<b>Sensitivity and fall-out of ICH on test set. . . . .</b>	22
5.4	<b>Specificity and miss-rate of ICH on test set. . . . .</b>	22



## ABBREVIATIONS

**DICOM** Digital Imaging and Communications in Medicine

**ICH** Intracranial Hemorrhage

**RNN** Recurrent Neural Network

**CNN** Convolutional Neural Network

**EPD** Epidural

**ITP** Intraparenchymal

**ITV** Intraventricular

**SBC** Subarachnoid

**SBD** Subdural

# CHAPTER 1


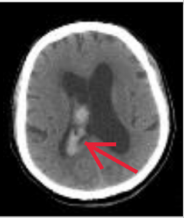
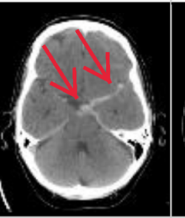
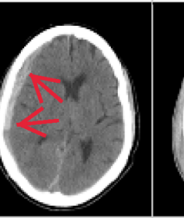

## INTRODUCTION

### 1.1 Research motivation and previous work

Intracranial hemorrhage detection is a very tedious process that is carried out by highly trained professionals. This paper aims at assisting the radiologists in this complex task by creating neural networks that are capable of detecting hemorrhages given the CT scan of a patient's cranium. A lot of great work has been done on ICH detection before (Hssayeni, Croock, Al-Ani, Al-khafaji, Yahya, and Ghoraani (Hssayeni et al.)). Though, the datasets used in similar research work were very small (Yuh et al. (2008)). The techniques used were also highly specialized and required a lot of hand engineering (Chang et al. (2018)). This paper differs from the previous work by using a much larger and more diverse dataset to train and test the neural networks. It's more of an end-to-end deep learning approach. Another major difference between this and the previous research work is the use of a single slice from the 3D CT Scan. Recurrent Neural Network (RNN) based model discussed in section 4 of this paper tries to mimic the process used by radiologists where the scan is systematically viewed under different window settings before an assessment is made. Convolutional Neural Network (CNN) based model discussed in section 5 uses convolution to extract the spatial features from the scan and make a prediction. Both the techniques are explained in great detail and results from the second technique are mentioned in the Results section.

### 1.2 Types of hemorrhages

There are mainly five type of hemorrhages that we're trying to detect. Each one of them occurs at a different location inside the cranium.

	Intraparenchymal	Intraventricular	Subarachnoid	Subdural	Epidural
Location	Inside of the brain	Inside of the ventricle	Between the arachnoid and the pia mater	Between the Dura and the arachnoid	Between the dura and the skull
Imaging					
Mechanism	High blood pressure, trauma, arteriovenous malformation, tumor, etc	Can be associated with both intraparenchymal and subarachnoid hemorrhages	Rupture of aneurysms or arteriovenous malformations or trauma	Trauma	Trauma or after surgery
Source	Arterial or venous	Arterial or venous	Predominantly arterial	Venous (bridging veins)	Arterial
Shape	Typically rounded	Conforms to ventricular shape	Tracks along the sulci and fissures	Crescent	Lentiform
Presentation	Acute (sudden onset of headache, nausea, vomiting)	Acute (sudden onset of headache, nausea, vomiting)	Acute (worst headache of life)	May be insidious (worsening headache)	Acute (skull fracture and altered mental status)

**Figure 1.1: Types of hemorrhages and their respective location inside the cranium.**

From 1.1, it is clear that knowing the type of hemorrhage gives us a very good idea about its location as well. This will be very helpful in locating the hemorrhage by just looking at the output from the neural network even though neural network output itself doesn't give any location information.

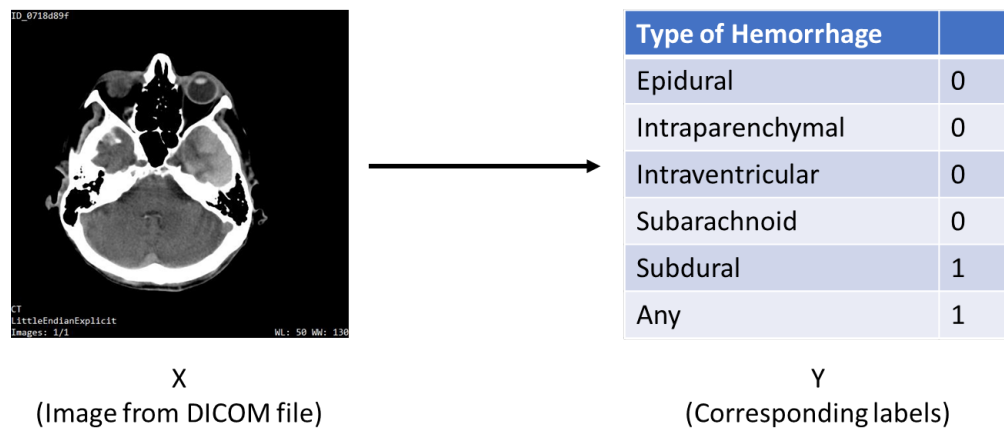
Possible output labels are Epidural (EPD), Intraparenchymal (ITP), Intraventricular (ITV), Subarachnoid (SBC) and Subdural (SBD) and any (ANY). The last label will be true if there is any hemorrhage present at all. There can be multiple hemorrhages present at the same time.

### 1.3 Model Input/Output

From the previous subsection we have pretty good idea of what will be the desired input and output for this kind of neural network. The input will be a single slice from the CT scan of a patient's cranium with dimension 512 x 512 in DICOM format. More about the input data is discussed in Dataset section.

The output will be probability (between 0 and 1) for each label. As already discussed in abstract and in the previous subsection, there can be multiple hemorrhages

at the same time. So, the output labels are NOT mutually exclusive. All the labels are treated independently.



**Figure 1.2: Illustration showing example of Input / Output for the Neural Networks.**

1.2 shows CT scan slice of a patient's cranium with subdural hemorrhage viewed in default window. It is mapped to its output where Subdural and Any label are 1 and remaining are 0. This shows that there is some hemorrhage present and chances of it being subdural are highly likely. It also shows that there are no other hemorrhages.

## CHAPTER 2

### LITERATURE SURVEY

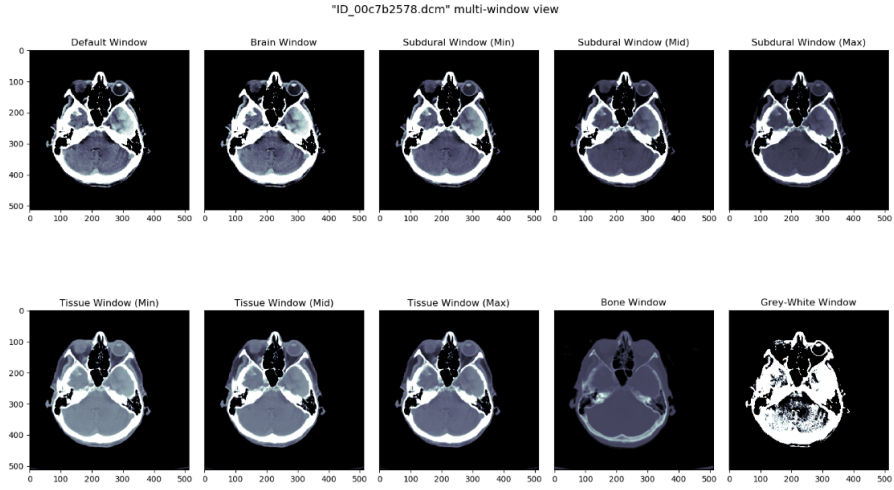
#### 2.1 See like a radiologist with Systematic windowing!

David Tang’s kernel “See like a Radiologist with Systematic Windowing” talks about viewing DICOM files under different window settings before making a prediction. It is the same methodology that is used by radiologist when they review CT scans in real life scenario.

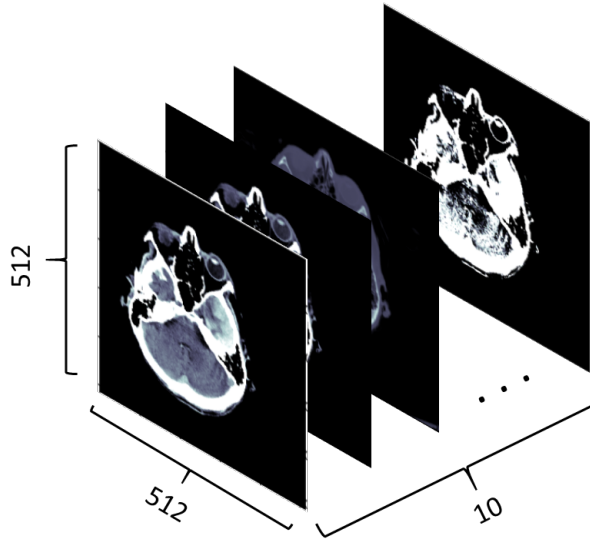
DICOM files appear different under different window settings, some hemorrhages are much easier to detect under specific window settings. For this particular reason windowing is used. Based on this kernel and our own understanding of the process, 10 window settings were selected.

Window Name	Window Center (WL/WC)	Window Width (WW)
Default	from DICOM tags	from DICOM tags
Brain	40	80
Subdural (min)	50	130
Subdural (mid)	75	215
Subdural (max)	100	300
Tissue (min)	20	350
Tissue (mid)	40	375
Tissue (max)	60	400
Bone	600	2800
Grey-white differentiation	32	8

Table 2.1: Different Window settings.



**Figure 2.1: DICOM Image viewed in 10 different window settings.**



**Figure 2.2: DICOM file under different window settings creating a temporal sequence.**

Using this methodology, a temporal sequence is created. All window settings mentioned in Table 2.1 are applied and the result is stacked. Figure 2.2 shows the creation of a temporal sequence that is taken as input by an RNN. RNN extracts the spatiotemporal data makes a prediction.

## **2.2 DON'T see like a radiologist! (fastai)**

Jeremy Howard's kernel "DON'T see like a radiologist! (fastai)" talks about NOT viewing DICOM files under different window settings before making a prediction. It is opposite of idea discussed in the previous section. The argument is that computers can process a much greater range and don't necessarily need windowing like radiologists do. Hence, instead of performing windowing (this is why windowing was an optional step in preparing the data for input), full range of the DICOM image is used after normalization. This is the chief idea for the CNN based model discussed in the next section.

## CHAPTER 3

### SYSTEM ANALYSIS

#### 3.1 RSNA Intracranial Hemorrhage Detection

RSNA (Radiological Society of North America) provided this dataset under a Kaggle competition “RSNA Intracranial Hemorrhage Detection”. The dataset is freely available for non-commercial and academic research purposes (point 7(A) under Competition Rules).

The original compressed data is 180 GB and in uncompressed format it is 448 GB. It consists of two splits, a labelled training set and an unlabeled test set. Labelled training data has 752,803 DICOM files and their corresponding output labels inside a CSV file. This is the part of dataset that was used in this research for training as well as validation and testing purposes.

Exactly 269 files out of 752,803 files had irregular dimensions. They were filtered to get 752,534 DICOM files. All of them have a consistent dimension of 512 x 512 pixels. Very carefully images were chosen from this new set of data for both the neural networks that we have discussed in later sections.

#### 3.2 Linear Transformation, Windowing and Min-Max scaling

Digital Imaging and Communications in Medicine (DICOM) files are quite different from typical image files like JPEG / PNG. DICOM files have a lot of metadata associated with them and the range of values for each pixel (voxel) is quite large compared to typical image files.

To parse the DICOM files, Pydicom library was used. A simple linear transformation is applied on the raw pixel array of the DICOM file to get the original Hounsfield



values.

$$HounsfieldValue = RawPixelValue * RescaleSlope + RescaleIntercept \quad (3.1)$$

NOTE: Rescale Slope and Rescale Intercept are present in DICOM file's metadata.

After this linear transformation, an optional windowing might be performed. To apply a window, the window width and window center values must be known.

$$Min = WindowCenter - \frac{WindowCenter}{2} \quad (3.2)$$

$$Max = WindowCenter + \frac{WindowCenter}{2} \quad (3.3)$$

$$value = \begin{cases} Min, & \text{if } value < Min \\ Max, & \text{if } value > Max \end{cases} \quad (3.4)$$

After the optional windowing, Min-Max rescaling is performed to get all the values between 0 and 1. The rescaling formula is given below.

$$RescaledValue = \frac{(Original - Minimum)}{(Maximum - Minimum)} \quad (3.5)$$

These steps are applied in training, validation and testing. Even when making predictions with the model this linear transformation and Min-Max rescaling should be done.

### 3.3 Weighted samples

Based on the types of hemorrhages present, there are total 32 types of images in this dataset. All of them are present in varying ratios, which creates a problem of severe imbalance as some types are under represented and some are over represented. Type 1 images are severely under represented whereas Type 32 images are present in the largest numbers. This creates severe class imbalance and makes it difficult to train the neural network.

Type	Samples	EPD	ITP	ITV	SBC	SBD	ANY
1	12	1	0	1	1	0	1
2	17	1	0	1	0	1	1
3	19	1	1	1	0	1	1
4	23	1	1	1	1	1	1
5	30	1	0	1	1	1	1
6	36	1	1	1	1	0	1
7	40	1	0	1	0	0	1
8	50	1	1	1	0	0	1
9	52	1	1	0	1	1	1
10	88	1	1	0	1	0	1
11	93	1	1	0	0	1	1
12	108	1	0	0	1	1	1
13	220	1	0	0	1	0	1
14	281	1	1	0	0	0	1
15	382	1	0	0	0	1	1
16	728	0	1	1	0	1	1
17	871	0	1	1	1	1	1
18	968	0	0	1	1	1	1
19	1084	0	0	1	0	1	1
20	1694	1	0	0	0	0	1
21	1955	0	1	1	1	0	1
22	2198	0	1	0	1	1	1
23	3309	0	1	0	0	1	1
24	3677	0	0	1	1	0	1
25	3932	0	1	0	1	0	1
26	5082	0	0	0	1	1	1
27	6816	0	1	1	0	0	1
28	9878	0	0	1	0	0	1
29	15664	0	1	0	0	0	1
30	16423	0	0	0	1	0	1
31	32096	0	0	0	0	1	1
32	644708	0	0	0	0	0	0

Table 3.1: Types of images with their respective sample count and the hemorrhages present in them

### 3.4 Supervised learning problem

Ground truth values are known, making this a supervised learning problem. This is a multi-label classification problem as opposed to a multi-class classification problem because the output classes are NOT mutually exclusive here. Each output label is independent and acts as an independent binary classification problem.

### 3.5 Selection of appropriate loss function

Binary cross-entropy is the chosen loss function taken element wise over the entire output. Loss formula is given below.

$$loss = -(y \log(p) + (1 - y) \log(1 - p)) \quad (3.6)$$

\* where y is the ground truth value, and p is the predicted value

This is taken element wise over all the output labels and then averaged.

$$loss_{EPD} = -(y_{EPD} \log(p_{EPD}) + (1 - y_{EPD}) \log(1 - p_{EPD})) \quad (3.7)$$

$$loss_{ITP} = -(y_{ITP} \log(p_{ITP}) + (1 - y_{ITP}) \log(1 - p_{ITP})) \quad (3.8)$$

$$loss_{ITV} = -(y_{ITV} \log(p_{ITV}) + (1 - y_{ITV}) \log(1 - p_{ITV})) \quad (3.9)$$

$$loss_{SBC} = -(y_{SBC} \log(p_{SBC}) + (1 - y_{SBC}) \log(1 - p_{SBC})) \quad (3.10)$$

$$loss_{SBD} = -(y_{SBD} \log(p_{SBD}) + (1 - y_{SBD}) \log(1 - p_{SBD})) \quad (3.11)$$

$$loss_{ANY} = -(y_{ANY} \log(p_{ANY}) + (1 - y_{ANY}) \log(1 - p_{ANY})) \quad (3.12)$$

$$loss = \frac{loss_{EPD} + loss_{ITP} + loss_{ITV} + loss_{SBC} + loss_{SBD} + loss_{ANY}}{6} \quad (3.13)$$

### 3.6 Model specific details

Both models have a 6-unit dense layer with sigmoid activation to make sure the output value for each label stays between 0 and 1.

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \quad (3.14)$$

Internal layers use ReLU as the activation function to make the training process faster.

$$relu(x) = \begin{cases} 0, & \text{if } value \leq 0 \\ x, & \text{if } value > 0 \end{cases} \quad (3.15)$$

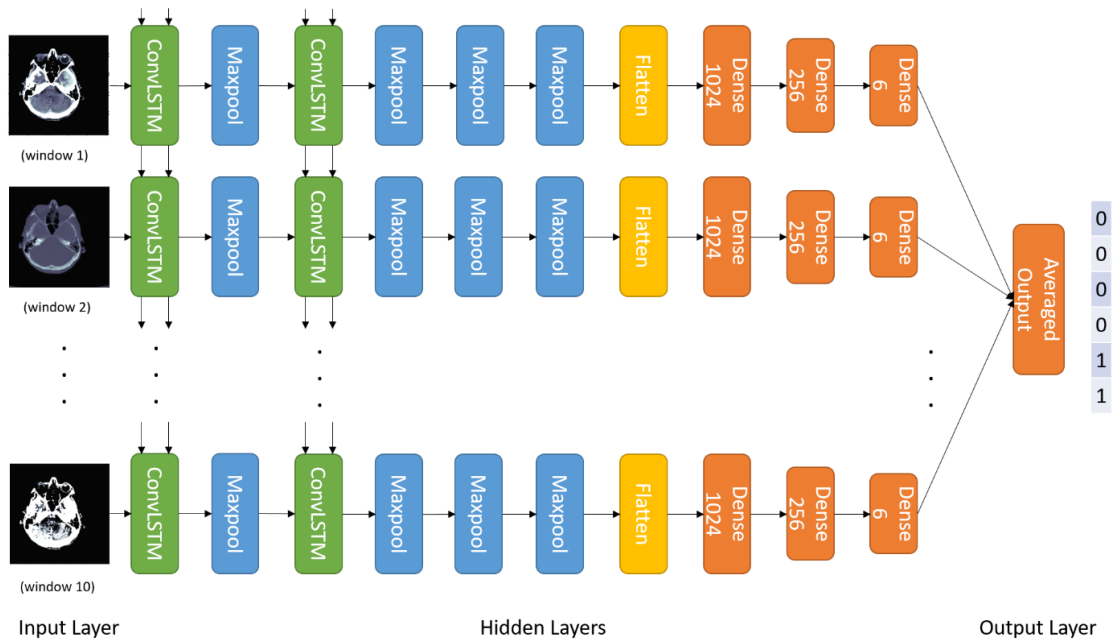
Selected optimizer is Adam. To avoid the problem of exploding and vanishing gradients from the network being too deep, GlorotUniform is the kernel and bias initializer. L2 regularization is used to prevent overfitting. Keras is the chosen framework with Tensorflow 2.0 as backend.

# CHAPTER 4

## SYSTEM DESIGN

### 4.1 Deep RNN Model

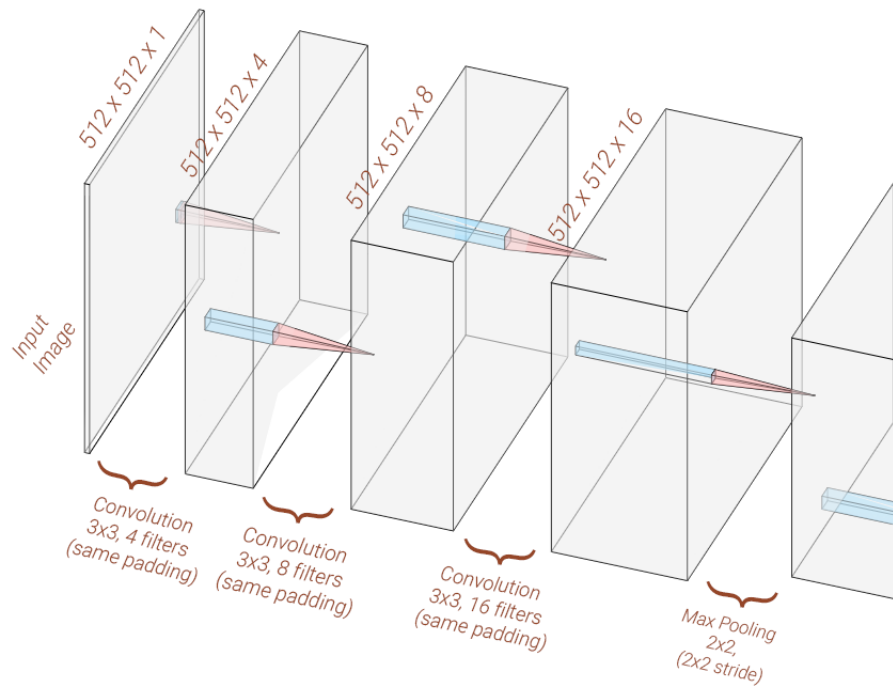
A Recurrent Neural Network takes the above shown temporal sequence as input and makes a prediction. RNNs are capable of mapping long-term spatiotemporal relations, hence it mimics a radiologist going through multiple windows before an assessment is made.



**Figure 4.1: RNN Model for ICH Detection.**

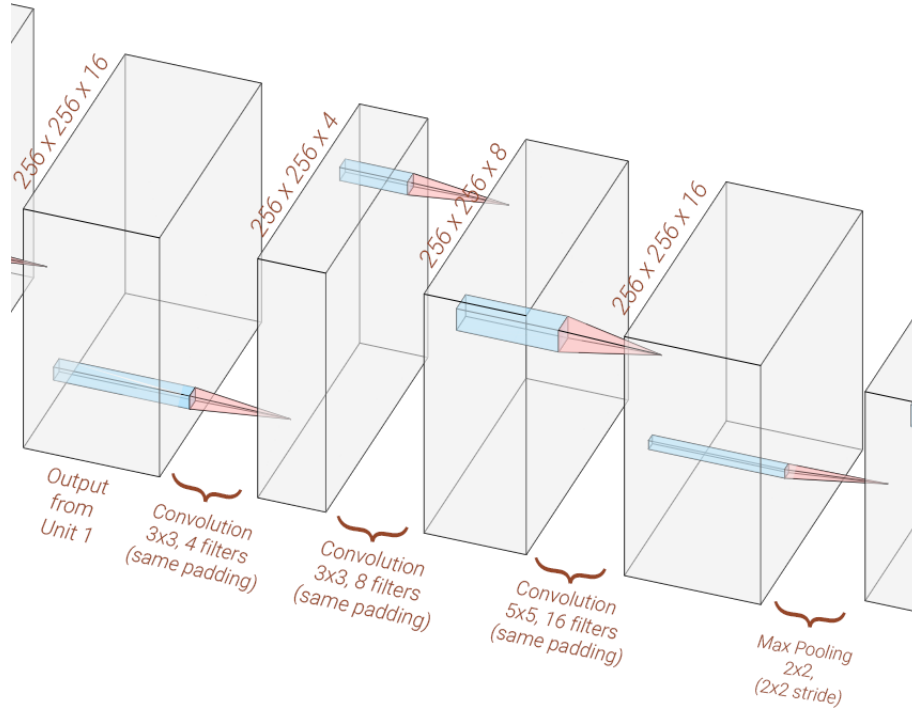
### 4.2 CNN Model

This neural network model is divided into 7 major units. First six units have a Batch Normalization layer at the end (not present in images). This gives independence to each unit. The first five units consist of convolution and max-pooling layers whereas the last unit is formed of fully-connected or dense layers.



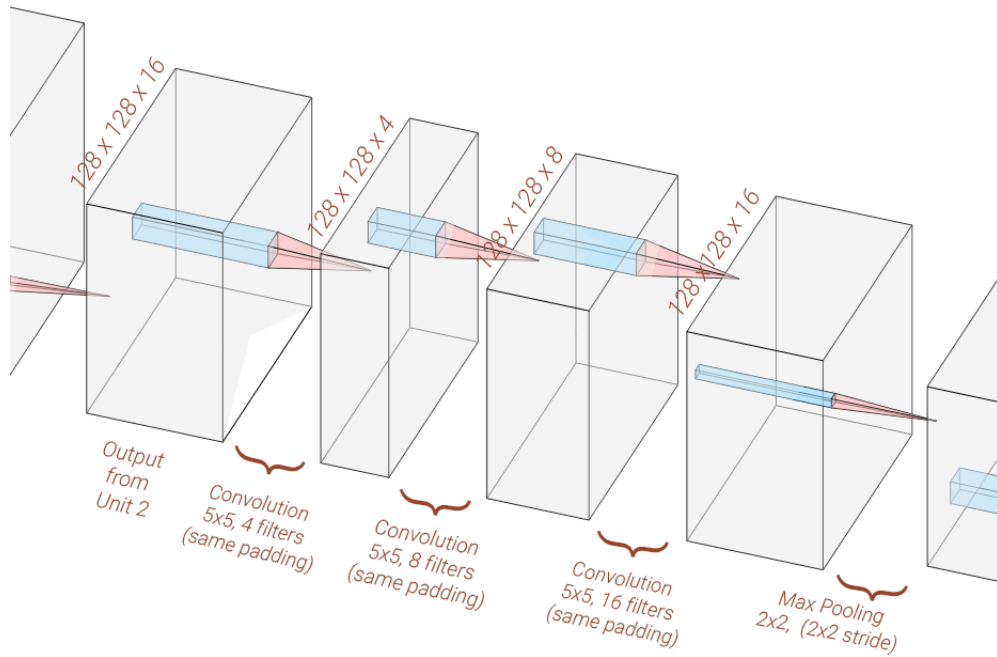
**Figure 4.2: Unit 1 of CNN Model.**

Unit 1 takes a NumPy matrix of  $512 \times 512 \times 1$  as input, it has three different convolutional layers with 4, 8 and 16 filters of dimensions  $3 \times 3$ ,  $3 \times 3$  and  $3 \times 3$  respectively. All these convolution operations preserve the original dimension by the use of “same” padding. A max pooling layer then reduces the image size in half by using a  $2 \times 2$  pool size with a stride of  $2 \times 2$ .



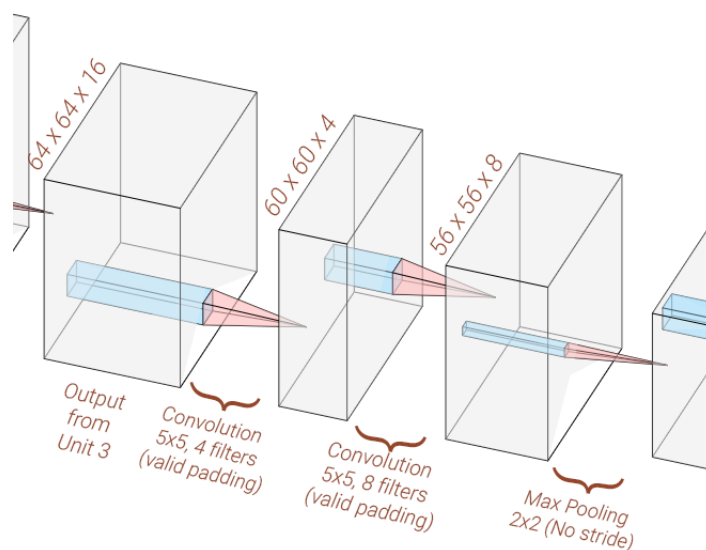
**Figure 4.3: Unit 2 of CNN Model.**

Unit 2 takes the output from the previous unit of dimension  $256 \times 256 \times 16$  as input, it has three different convolutional layers with 4, 8 and 16 filters of dimensions  $3 \times 3$ ,  $3 \times 3$  and  $5 \times 5$  respectively. All these convolution operations preserve the original dimension by the use of “same” padding. Again, a max pooling layer then reduces the image size in half by using a  $2 \times 2$  pool size with a stride of  $2 \times 2$ .



**Figure 4.4: Unit 3 of CNN Model.**

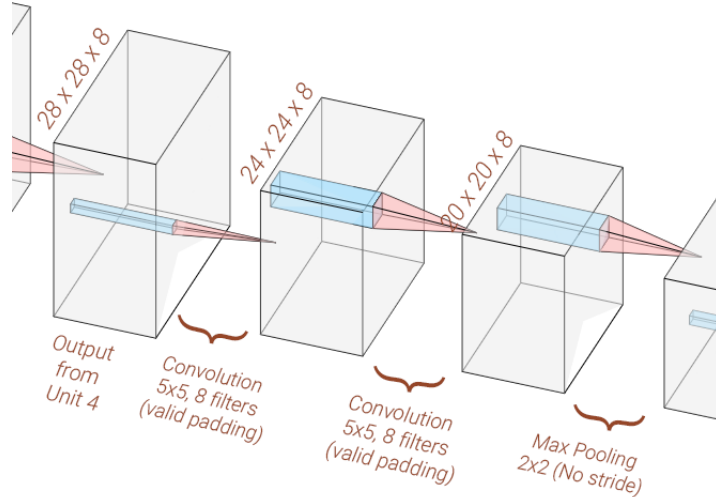
Unit 3 takes the output from the previous unit of dimension  $128 \times 128 \times 16$  as input, it has three different convolutional layers with 4, 8 and 16 filters of dimensions  $5 \times 5$ ,  $5 \times 5$  and  $5 \times 5$  respectively. All these convolution operations preserve the original dimension by the use of “same” padding. Again, a max pooling layer then reduces the image size in half by using a  $2 \times 2$  pool size with a stride of  $2 \times 2$ .



**Figure 4.5: Unit 4 of CNN Model.**

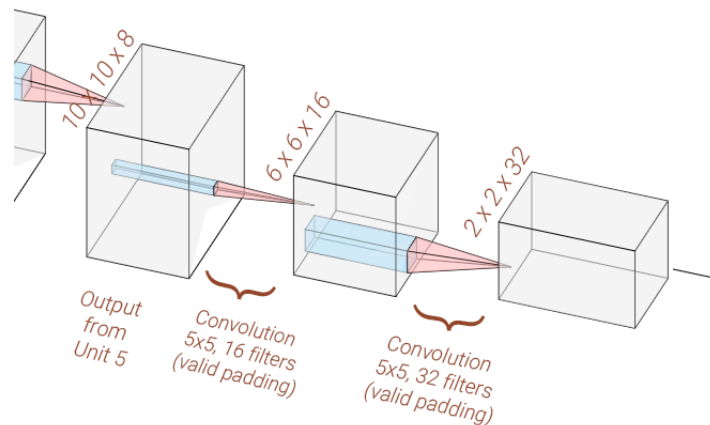


Unit 4 takes the output from the previous unit of dimension  $64 \times 64 \times 16$  as input, it has two convolutional layers with 4 and 8 filters of dimensions  $5 \times 5$  and  $5 \times 5$  respectively. A max pooling layer then reduces the image size by using a  $2 \times 2$  filter.



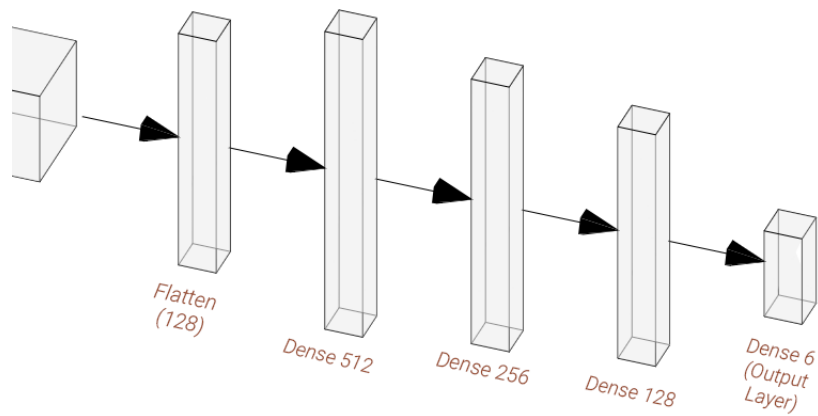
**Figure 4.6: Unit 5 of CNN Model.**

Unit 5 takes the output from the previous unit of dimension  $28 \times 28 \times 8$  as input, it has two convolutional layers with 8 and 8 filters of dimensions  $5 \times 5$  and  $5 \times 5$  respectively. A max pooling layer then reduces the image size by using a  $2 \times 2$  filter.



**Figure 4.7: Unit 6 of CNN Model.**

Unit 6 is different from all the previous layers as it has no max-pooling layer. It takes the output from the previous unit of dimension  $10 \times 10 \times 8$  as input, it has two convolutional layers with 16 and 32 filters of dimensions  $5 \times 5$  and  $5 \times 5$  respectively.



**Figure 4.8: Unit 7 of CNN Model.**

Unit 7 flattens the output from the previous unit to get an input of 128. There are three dense layers after that having 512, 256 and 128 units respectively. The final / output layer is a 6-unit dense layer. The activation function in the final layer is sigmoid. The output shape is (6,).

# CHAPTER 5

## EXPERIMENT RESULTS

### 5.1 Coding

Model compilation code:

[1] This is the code for CNN Model discussed in System Design's CNN Model section.

[2] It uses the tensorflow library to define and compile the model.

[3] Following which the model is saved into a H5 file named "testing.h5".

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense, BatchNormalization
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import binary_crossentropy
```

```
model = Sequential(name='ConvNet_Updated')
```

```
model.add(Conv2D(input_shape=(512, 512, 1), filters=4, kernel_size=(3, 3),
padding='same', activation='relu', kernel_initializer=glorot_uniform(),
bias_initializer=glorot_uniform(), kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(Conv2D(filters=8, kernel_size=(3, 3), padding='same', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(Conv2D(filters=16, kernel_size=(3, 3), padding='same', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(MaxPool2D(pool_size=(2, 2), strides=(2,2)))
model.add(BatchNormalization())
```

```
model.add(Conv2D(filters=4, kernel_size=(3, 3), padding='same', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(Conv2D(filters=8, kernel_size=(3, 3), padding='same', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(Conv2D(filters=16, kernel_size=(5, 5), padding='same', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(MaxPool2D(pool_size=(2, 2), strides=(2,2)))
model.add(BatchNormalization())
```

```
model.add(Conv2D(filters=4, kernel_size=(5, 5), padding='same', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(Conv2D(filters=8, kernel_size=(5, 5), padding='same', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(Conv2D(filters=16, kernel_size=(5, 5), padding='same', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(MaxPool2D(pool_size=(2, 2), strides=(2,2)))
model.add(BatchNormalization())
```

```
model.add(Conv2D(filters=4, kernel_size=(5, 5), padding='valid', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(Conv2D(filters=8, kernel_size=(5, 5), padding='valid', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(BatchNormalization())
```

```
model.add(Conv2D(filters=8, kernel_size=(5, 5), padding='valid', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(Conv2D(filters=8, kernel_size=(5, 5), padding='valid', activation=
'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
kernel_regularizer=l2(), bias_regularizer=l2()))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(BatchNormalization())
```

```
# this unit has been removed
```

```
#model.add(Conv2D(filters=16, kernel_size=(5, 5), padding='valid', activation=
#relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
```

```
#kernel_regularizer=l2(), bias_regularizer=l2()))
#model.add(Conv2D(filters=32, kernel_size=(5, 5), padding='valid', activation=
#'relu', kernel_initializer=glorot_uniform(), bias_initializer=glorot_uniform(),
#kernel_regularizer=l2(), bias_regularizer=l2()))
#model.add(BatchNormalization())

model.add(Flatten())

# i added this layer
model.add(Dense(units=1024, activation='relu', bias_initializer=glorot_uniform(),
kernel_initializer=glorot_uniform(), kernel_regularizer=l2(), bias_regularizer=l2(),
name='dense_new'))

# weight matrix of this layer has changed now
model.add(Dense(units=512, activation='relu', bias_initializer=glorot_uniform(),
kernel_initializer=glorot_uniform(), kernel_regularizer=l2(), bias_regularizer=l2()))

model.add(Dense(units=256, activation='relu', bias_initializer=glorot_uniform(),
kernel_initializer=glorot_uniform(), kernel_regularizer=l2(), bias_regularizer=l2()))

model.add(Dense(units=128, activation='relu', bias_initializer=glorot_uniform(),
kernel_initializer=glorot_uniform(), kernel_regularizer=l2(), bias_regularizer=l2()))

model.add(Dense(units=6, activation='sigmoid', bias_initializer=glorot_uniform(),
kernel_initializer=glorot_uniform(), kernel_regularizer=l2(), bias_regularizer=l2()))

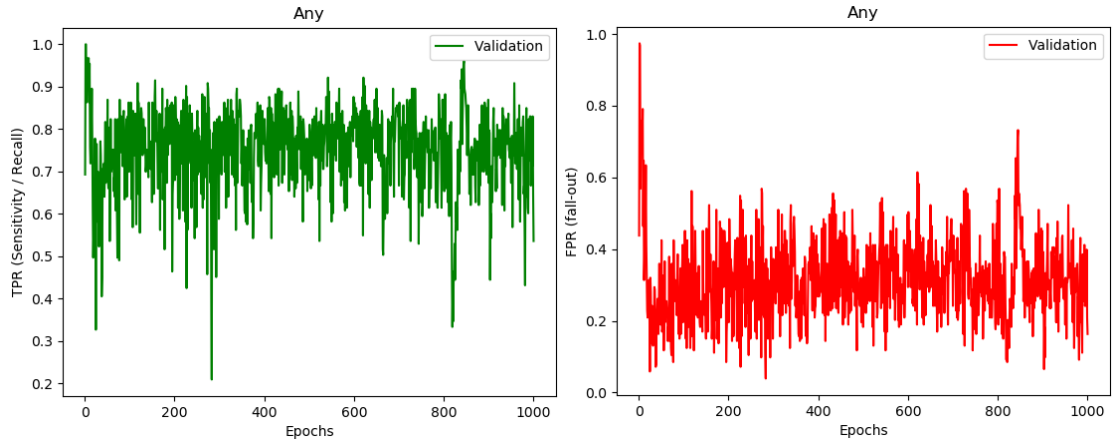
model.compile(optimizer=Adam(), loss=binary_crossentropy)

model.summary()

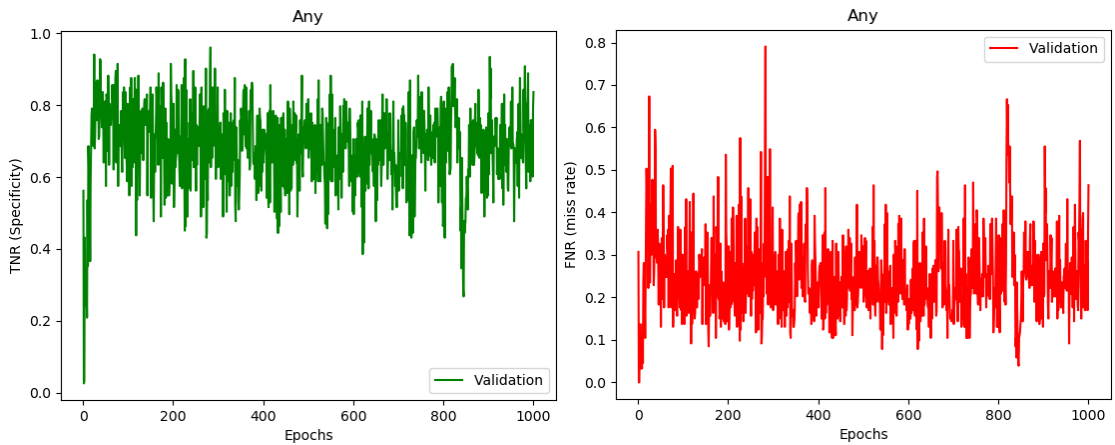
model.save('testing.h5')
```

## 5.2 Testing

### 5.2.1 Validation Results

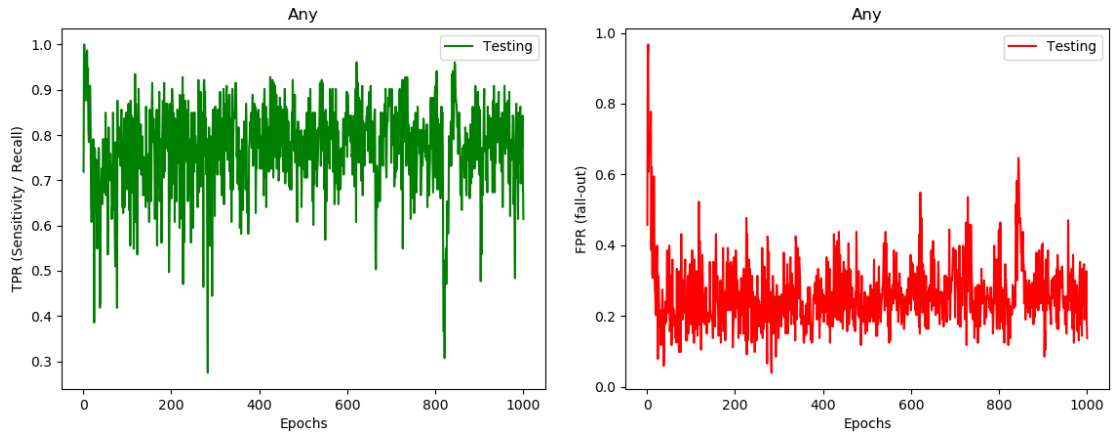


**Figure 5.1: Sensitivity and fall-out of ICH on validation set.**

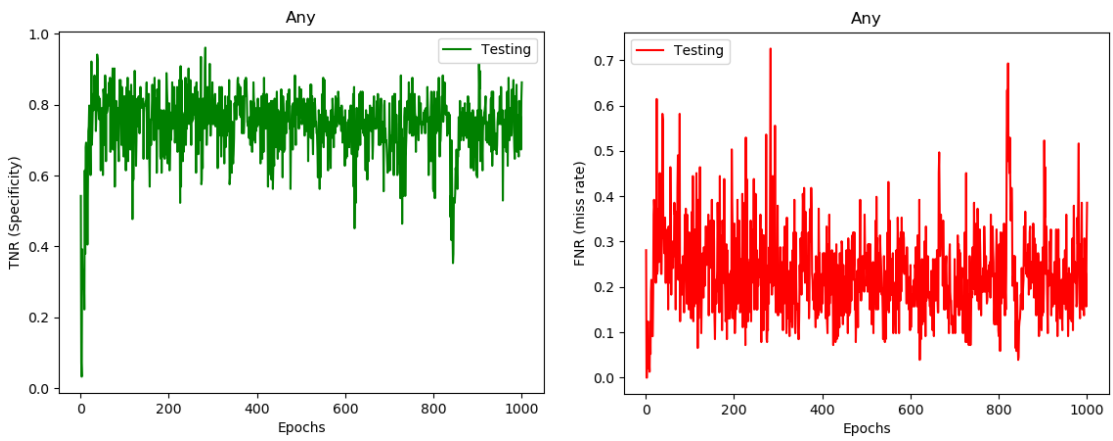


**Figure 5.2: Specificity and miss-rate of ICH on validation set.**

## 5.2.2 Test Results



**Figure 5.3: Sensitivity and fall-out of ICH on test set.**



**Figure 5.4: Specificity and miss-rate of ICH on test set.**

## **CHAPTER 6**

### **CONCLUSION**

From the results mentioned in the previous section we can conclude the following:

1. This particular neural network has a good sensitivity i.e., to correctly identify the cases where a hemorrhage is present. Also, it has a fairly good specificity i.e., to correctly identify the cases where a hemorrhage is absent.
2. From figure 5.1 and 5.3 we can conclude the sensitivity part. From figure 5.2 and 5.4 we can confirm the specificity part.
3. Naturally there are some false positives and false negatives as well. But, overall it has a good performance in observing whether a hemorrhage is present or not given a single slice from the CT scan of a patient's cranium.

This model can be incorporated into things like a web application, a mobile application or a desktop application. It aims at using Artificial Intelligence for helping radiologist in making quick and accurate decisions while making an assessment.



## **CHAPTER 7**

### **FUTURE ENHANCEMENTS**

While this neural network is very good at detecting whether a hemorrhage exists or not there are several parts where it needs further improvement.

They are mentioned below:

1. The specificity can be further improved so that True Negative Rate (TNR) goes up.
2. Individual hemorrhage type detection still needs improvement. Even though they have good specificity and sensitivity, it can be improved even further.

## REFERENCES

1. Chang, P. D., Kuoy, E., Grinband, J., Weinberg, B. D., Thompson, M., Homo, R., Chen, J., Abcede, H., Shafie, M., Sugrue, L., Filippi, C. G., Su, M.-Y., Yu, W., Hess, C., and Chow, D. (2018). “Hybrid 3d/2d convolutional neural network for hemorrhage evaluation on head ct.” *American Journal of Neuroradiology*, (1609-1616).
2. Howard, J. “Don’t see like a radiologist! (fastai). [Online]. Available from <https://www.kaggle.com/jhoward/don-t-see-like-a-radiologist-fastai>.
3. Hssayeni, M. D., Croock, M. S., Al-Ani, A., Al-khafaji, H. F., Yahya, Z. A., and Ghoraani, B. “Intracranial hemorrhage segmentation using deep convolutional model. Arxiv [Preprint]. Available from <https://arxiv.org/pdf/1910.08643.pdf>.
4. Kaggle. “Rsna intracranial hemorrhage detection | data. [Online]. Available from <https://www.kaggle.com/c/rsna-intracranial-hemorrhage-detection/data/>.
5. Kaggle. “Rsna intracranial hemorrhage detection | rules. [Online]. Available from <https://www.kaggle.com/c/rsna-intracranial-hemorrhage-detection/rules>.
6. Lenail, A. “Nn-svg. [Online]. Available from <http://alexlenail.me/NN-SVG/AlexNet.html>. [Accessed: 01 April 2020].
7. Mason, D. and contributors. “Pydicom. [Online]. Available from <https://pypi.org/project/pydicom/>.
8. Tang, D. “See like a radiologist with systematic windowing. [Online]. Available from <https://www.kaggle.com/dcstang/see-like-a-radiologist-with-systematic-windowing>.
9. Tensorflow. “Keras. [Online]. Available from <https://www.tensorflow.org/guide/keras>.
10. Tensorflow. “Tensorflow. [Online]. Available from <https://www.tensorflow.org/>.
11. Tensorflow. “tf.keras.initializers.glorotuniform. [Online]. Available from [https://www.tensorflow.org/api\\_docs/python/tf/keras/initializers/GlorotUniform](https://www.tensorflow.org/api_docs/python/tf/keras/initializers/GlorotUniform). [Accessed: 04 April 2020].
12. Tensorflow. “tf.keras.optimizers.adam. [Online]. Available from [https://www.tensorflow.org/api\\_docs/python/tf/keras/optimizers/Adam](https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam). [Accessed: 04 April 2020].
13. Tensorflow. “tf.keras.regularizers.l2. [Online]. Available from [https://www.tensorflow.org/api\\_docs/python/tf/keras/regularizers/l2](https://www.tensorflow.org/api_docs/python/tf/keras/regularizers/l2). [Accessed: 04 April 2020].
14. Wikipedia. “Cross entropy. [Online]. Available from [https://en.wikipedia.org/wiki/Cross\\_entropy](https://en.wikipedia.org/wiki/Cross_entropy).
15. Yuh, E. L., Gean, A. D., Manley, G. T., Callen, A. L., and Wintermark, M. (2008). “Computer-aided assessment of head computed tomography (ct) studies in patients with suspected traumatic brain injury.” *Journal of neurotrauma*, 1163–1172.