

# **Configuration Manual**

Intracranial Haemorrhage Detection using Machine Learning Models MSc. Research Project

> Bellana Tirupati Patro Student ID: 18196551

School of Computing National College of Ireland

Supervisor: Dr. Rashmi Gupta

### **National College of Ireland**

### **MSc Project Submission Sheet**



#### **School of Computing**

Student **BELLANA TIRUPATI PATRO** Name:

**Student ID:** X18196551

**Programme:** MSc in Data Analytics

Year: 2019-2020

Module: MSc Research Project

Lecturer: Dr Rashmi Gupta Submission Due Date: 17/08/2020

**Project Title:** Intracranial Haemorrhage Detection using Deep Learning Models

Word Count: 1037 Page Count:

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

#### Signature: .....

Date:

.....

### PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple	
copies)	
Attach a Moodle submission receipt of the online project	
submission, to each project (including multiple copies).	
You must ensure that you retain a HARD COPY of the project, both	
for your own reference and in case a project is lost or mislaid. It is not	
sufficient to keep a copy on computer.	

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

# **Configuration Manual**

Bellana Tirupati Patro Student ID: 18196551

# **1** Hardware/Software Requirements

### **1.1 Hardware Requirements**

Laptop	Lenovo Ideapad 330
RAM	8GB
Graphics Processing Unit(GPU)	Nvidia GTX 960 -2GB

### **1.2 Software Requirements**

Operating System	Windows 10
Programming Tools	Jupyter Notebook, Python Version 3, Anaconda
Graphics Processing Unit(GPU)	Nvidia GTX 960 -2GB

# 2 Data Gathering

Data Gathering is done for this research project using Kaggle API.

Step 1: Install the Kaggle in Jupyter Notebook.

💭 jupyter	Untitled30 Last Checkpoint: 4 minutes ago (unsaved changes)	P Log
File Edit	View Insert Cell Kernel Widgets Help	Trusted Python
B + × 4	] 🖪 🛧 🗸 HRun 🔳 C 🕨 Code 🗸 🖾	
In [2]:	pip install kaggle	
	Providence all and a stiffied with 1 10 is a large large large day like a share (from here large	n kaggie) (1.24.2)
	Requirement already satisfied: six>=1.10 in c:\users\tirup\anaconda3\lib\site-packages (from kaggle) (1. Requirement already satisfied: certifi in c:\users\tirup\anaconda3\lib\site-packages (from kaggle) (2015 Requirement already satisfied: python-dateutil in c:\users\tirup\anaconda3\lib\site-packages (from kaggle) (2015 Requirement already satisfied: requests in c:\users\tirup\anaconda3\lib\site-packages (from kaggle) (2.2 Requirement already satisfied: requests in c:\users\tirup\anaconda3\lib\site-packages (from kaggle) (2.2 Requirement already satisfied: requests in c:\users\tirup\anaconda3\lib\site-packages (from kaggle) (2.2 Requirement already satisfied: top (from kaggle) Downloading https://files.pythonhosted.org/packages/9f/42/e336f96a8b6007428df772d0d159b8eee9b2f181592 on-slugify-4.0.1.tar.gz Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\tirup\anaconda3\lib\site-packages (from request (3.0.4) Requirement already satisfied: idna<2.9,>=2.5 in c:\users\tirup\anaconda3\lib\site-packages (from request Collecting text-unidecode>=1.3 (from python-slugify-kaggle) Davaloading https://files.puthonhosted.org/packages/firy.kaggle)	<pre>m kaggle ( 1.24.2) 12.0) .6.16) le) (2.8.0) 22.0)</pre>
	Requirement already satisfied: six>=1.10 in c:\users\tirup\anaconda3\lib\site-packages (from kaggle) (1. Requirement already satisfied: certifi in c:\users\tirup\anaconda3\lib\site-packages (from kaggle) (2015 Requirement already satisfied: python-dateutil in c:\users\tirup\anaconda3\lib\site-packages (from kaggle) (2015 Requirement already satisfied: requests in c:\users\tirup\anaconda3\lib\site-packages (from kaggle) (2.2 Requirement already satisfied: chardetd.org/packages/9f/42/e336f96a8b6007428df772d0df59b8eee9b2f181f59 on-slugify-4.0.1:tar.gz Requirement already satisfied: chardetd3.1.0,>=3.0.2 in c:\users\tirup\anaconda3\lib\site-packages (from request Collecting text-unidecode>1.3 (from python-slugify->kaggle) Downloading https://files.pythonhosted.org/packages/a6/a5/c0b6468d3824fe3fde30dbb5e1f687b291608f947362 _unidecode=1.3-py2.py3-none-any.wh1 (78kB) _unidecode=1.3-py2.py3-none-any.wh1 (78kB)	<pre>m kaggle) (1.24.2) 12.0) 9.6.16) le) (2.8.0) 22.0) ) 3a4931150660402c0/pyth m requests-&gt;kaggle) sts-&gt;kaggle) (2.8) 31bbf7dabbf5a87d7/text</pre>

Step 2: Login to Kaggle.com using your credentials.

Step 3: Goto My Account page.

Step 4: Click on "Create New API Token"

$\leftrightarrow$ $\rightarrow$ C $$ kaggle.com/tpath	ro/account
≡ kaggle	Q Search
Ø Home	Home Competitions Datasets Notebooks Discussion •••
<b>Ψ</b> Compete	Email Address tirupati.patro071291@gmail.com
🔟 Data	
<> Notebooks	Phone Verification
🔲 Discuss	Verified
🔂 Courses	Email Preferences
🔁 Jobs	Your email preferences can now be controlled on the Notification settings page.
✓ More	
Recently Viewed	API
RSNA Intracranial Hem	command line. Read the docs
SVM for Multiclass Cla	Create New API Token Expire API Token
🗃 🛛 keras Efficientnetb4 fo	
AutoEncoder with SV	Quota Private Datasets 0 B / 100 GB

Step 5: The kaggle.json file is downloaded in Downloads folder of local machine.

Step 6: Copy this kaggle.json file from Dowloads folder to .kaggle folder present in Users folder inside C: drive of your local machine.

🛃 📜 🗧 🛛 .kaggle			
Home Share View			
Quick       Copy       Paste       ↓ Cut         Copy       Paste       ↓ Copy path         Paste       Paste shortcut	Move Copy to * Copy	New item ▼ ☐ Easy access ▼ New folder	Properties
Clipboard	Organize	New	Open
→ 👻 ↑ 📜 > This PC > Windo	ws (C:) > Users > tirup > .kago	gle	
Name	^	Date modified	Туре
Quick access     Desktop	ison	15/08/2020 23:46	JSON File
Downloads X			
E Pictures			
DICOM			
📜 png			
📕 Report			
E rsna-intracranial-he			
OneDrive			

Step 7: Goto dataset page by clicking on this link <u>https://www.kaggle.com/c/rsna-intracranial-hemorrhage-detection/data</u>

Step 8: Scroll down a bit and click on the link as per screenshot below.

÷	→ C 🌘 kaggle.com/c/rsna-int	rracranial-hemorrhage-detection/data					
=	kaggle	Q Search					
Ø	Home	Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions Late Submission					
ዋ	Compete						
	Data	What files do I need?					
$\langle \rangle$	Notebooks	This is a two-stage challenge. You will need the images for the current stage - provided as stage_2_test.zip. You will also need the					
	Discuss	as well as a sample of what your submission should look like.					
ଡ	Courses	Note: The timeline page outlines the two-stage format and deadlines. Stage 2 data is now available in accordance with this timeline. Also review two-stage FAQs for more details.					
₿	Jobs						
$\checkmark$	More	What should I expect the data format to be?					
Rece	Recently Viewed						
2555	RSNA Intracranial Hem						
d.	SVM for Multiclass Cla	kaggle competitions download -c rsna-intracranial-hemorrhage-detection					
- 1	koras Efficientneth/l fo						

Step 9: Goto Anaconda Prompt and paste the command: **kaggle competitions download -c rsna-intracranial-hemorrhage-detection and click enter .** (refer the screenshot below).



Step 10: The dataset starts downloading in Zip folder inside Windows folder in C drive. (refer the screenshot below).

L   ☑ L =   File Home Share	View Compre	Extract ssed Folder Tools	tirup		
Pin to Quick Copy Paste	Cut Copy path Paste shortcut	e Copy to	Rename	New item • The basy access • New folder	Properties
Clipboard Organize New Open					
Clipboard		Organize		New	Open
Clipboard $\leftarrow \rightarrow \checkmark \uparrow \boxed{1} \rightarrow \text{This}$	s PC > Windows (C:	Organize ) > Users > tiru	p	New	Open
Clipboard $\leftarrow \rightarrow \checkmark \uparrow  \rightarrow This$	s PC > Windows (C:	Organize ) > Users > tiru	p	New Date modified	Ореп
Clipboard ← → ∽ ↑ I→ This A Quick access	s PC > Windows (C: Name rsna-intracran	Organize ) > Users > tirup ^ ial-hemorrhage-d	etection	New Date modified 16/08/2020 00:01	Open Type WinRAR ZIP archive

## **3** Code Structure

	>	This PC	>	Downloads	>	Thesis	Code
	~	THISFC	*	Downloaus	~	1110313	Coue

Name	Date modified	Туре	Size
Random Forest Classifier.ipynb	16/08/2020 18:01	IPYNB File	139 KB
Sparse Autoencoder SVM.ipynb	16/08/2020 17:01	IPYNB File	35 KB
PCA and SVM.ipynb	16/08/2020 15:48	IPYNB File	22 KB
CNN.ipynb	16/08/2020 07:19	IPYNB File	33 KB
ImageConversiontoPNG.ipynb	16/08/2020 04:50	IPYNB File	4 KB

# 4 Data Conversion

Step 1: Unzip the data set downloaded to any specific folder you need.

Step 2: Upload the thesis code folder to jupyter notebook and open ImageConversiontoPNG.ipynb to jupyter notebook.

Step 3: Change the path for reading the DICOM file and change the path for storing the PNG file in ImageConversiontoPNG.ipynb.( Refer the screenshot below)



Step 4: After replacing the directory paths, import the necessary libraries.

```
In [8]: #Libraries
import os
import pydicom
import cv2
import glob
import multiprocessing as mp
import numpy as np
```

Step 5: Run the remaining part of the code.

```
In [ ]: # Functions
            def window_image(img, window_center, window_width, intercept, slope):
    img = (img * slope + intercept)
    img_min = window_center - window_width // 2
    img_max = window_center + window_width // 2
    img[img < img_min] = img_min
    img[img > img_max] = img_max
                  return img
            def get_first_of_dicom_field_as_int(x):
                 if type(x) == pydicom.multival.MultiValue:
    return int(x[0])
                  else:
                        return int(x)
            def get_windowing(data):
                 def convert_to_png(dcm_in):
                 dicomimage = pydicom.dcmread(dcm_in)
                 window_center, window_width, intercept, slope = get_windowing(dicomimage)
img = pydicom.read_file(dcm_in).pixel_array
img = window_image(img, window_center, window_width, intercept, slope)
                 cv2.imwrite(os.path.join(directory_png, os.path.basename(dcm_in)[:-3] + 'png'), img)
             # Extract images in parallel
            dicom = glob.glob(os.path.join(directory_dicom, '*.dcm'))
type(dicom)
            print(dicom)
            len(dicom)
            import dask as dd
all_images = [dd.delayed(convert_to_png)(dicom[x]) for x in range(len(dicom))]
dd.compute(all_images)
```

Step 5: The image starts getting converted to PNG in the corresponding folder. (Refer Screenshot below)

Hie Home Share	view Picture 1001	5					
🖈 📄 🔺 🖌	ut 📃	🗎 🗙 🖃	New i	tem 👻 🛃 🛃	🔮 Open 👻 🔠 Select a	ш	
in to Quick Copy Paste	opy path Move	Copy Delete Renam	e New Easy a	Properties	Edit Select r	ione	
access M P	aste shortcut to •	to •	folder	· · ·	History 📑 Invert s	election	
Clipboard		Organize	New	Ope	n Seleo	t	
$\leftarrow \rightarrow \checkmark \uparrow$ > This	PC > New Volume (E	:) > rsna-intracranial-	hemorrhage-detect	ion > stage_2_train_	png		
🖈 Quick access			A LA			19 G	
🔜 Desktop 🛛 🖈		26	$\langle - \rangle$	(-2.)	(A)		A HOLE
🖶 Downloads 🛛 🖈					Canal P	0	
🖹 Documents 🛛 🖈							
📰 Pictures 🛛 🖈	ID_0000aee4b	ID_0000f1657	ID_000a050f3	ID_000a631d3	ID_000a90167	ID_000a18499	ID_000a33979
DICOM	<000	_	87 D	-91 (D)		<u>~~~</u>	At S
🧼 New Volume (E:)				3			()
📕 png							
📕 rsna-intracranial-he	ID 000b76199	ID 000bd8380	ID 000c1f118	ID 000c7ccf6	ID 000c8cc79	ID 000cb0b16	ID 000ce08f0
<ul> <li>OneDrive</li> </ul>					_	_	
S This PC	2123			$\bigcirc$			
3D Objects			2 - D			1 1 A	
Desktop							
Documents	ID_000d44b3c	ID_000d457ad	ID_000d502a4	ID_000d69988	ID_000da5e3a	ID_000da30e0	ID_000dc257f
Downloads			$\frown$	$\frown$	$\frown$		
Music	( )		$r \rightarrow$		(		
E Pictures							
Videos							
💺 Windows (C:)	ID_000e60ed1	ID_000e918cb	ID_000ec5e66	ID_000ed207a	ID_000edbf38	ID_000ef9a73	ID_000ere40a
🧼 New Volume (E:)	515		CTO:		$\bigcirc$		
🥩 Network		$\bigcirc$				Ú,	
	ID_000f83a10	ID_000fbb8e0	ID_000fd6d15	ID_000ff2f62	ID_00a00c3f9	ID_00a0c71e9	ID_00a0d23b6
	10 R ()	A 300		-	200	othe	
587,354 items 1 item selec	ted 4.47 KB						

# 5 Executing Sparse autoencoder and SVM Model

Step 1: Open the sparseautoencoder and SVM.ipynb from Thesis code folder.

Step 2: Set the path in the code from where data labels and PNG images are read.



Step 3: Reading the train label file and creating a dataframe.

```
n [4]: import pandas as pd
import numpy as np
train_df = pd.read_csv(csv_path)
train_df['filename'] = train_df['ID'].apply(lambda st: "ID_" + st.split('_')[1] + ".png")
train_df['type'] = train_df['ID'].apply(lambda st: st.split('_')[2])
train_df.head()
```

. .

Step 4: Creating a dataframe by checking the filename from labels dataframe and checking if the file exists for 50,000 images in the Image folder



Step 5: Splitting the label data frame created into test and train labels data frame.

```
In [93]: from sklearn.model_selection import train_test_split
train_label, test_label = train_test_split(sample_df, test_size=0.2)
print(train_label.shape)
print(test_label.shape)
arraylist1=list(train_label.index.values)
print(len(arraylist1))
arraylist2=list(test_label.index.values)
print(len(arraylist2))
```

Step 6: Import libraries for loading training image and convert to numpy array

```
: from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img import PIL
```

Step 6: After splitting the label dataframe, read filename and from numpy array for training images.

```
In [95]: from keras.preprocessing import image
all_images=[]
for i in arraylist1:
    file = train_label.loc[i]["filename"]
    if(os.path.isfile(imgpath+file)):
        img = image.load_img(imgpath+file, target_size=(128,128))
        img = image.img_to_array(img)
        all_images.append(img)
print(len(all_images))
print(len(train_label))
```

Trainimagearray=np.array(all\_images)

Step 7: Loading image for testing data and converting to numpy array

```
5]: test_images=[]
for j in arraylist2:
    file =test_label.loc[j]["filename"]
    if(os.path.isfile(imgpath+ file)):
        img = image.load_img(imgpath+file, target_size=(128,128))
        img = image.img_to_array(img)
        test_images.append(img)
    print(len(test_images))
    print(len(test_label))
```

In [99]: Testimagearray=np.array(test\_images)

Step 8: One hot encoding for train and test labels

```
#converting encoded data to categories, one hot encoded data provided in the datas
def label_img(label_arr):
   labels=[]
   for i in range(len(label arr)):
        if (label_arr[i][0]==0):
            labels.append("Normal")
        elif (label_arr[i][1]==1):
           labels.append("Epidural")
        elif (label_arr[i][2]==1):
            labels.append("Intraparenchymal")
        elif (label_arr[i][3]==1):
            labels.append("Intraventricular")
        elif (label arr[i][4]==1):
            labels.append("Subarachnoid")
        else:
            labels.append("Subdural")
    return labels
```

```
train_label=np.array(train_label)
train_label= label_img(train_label)
```

```
test_label=np.array(test_label)
test_label= label_img(test_label)
test_label
```

Step 9: Normalize the pixel values of training and testing image numpy arrays

```
In [30]: # Normalize the pixels
Trainimagearray = Trainimagearray/255.
Testimagearray = Testimagearray/255.
```

Step 10: Building the encoder part of the auto encoder

Step 11: Building the decoder part of autoencoder

```
In [33]:
# Decoder Part of the autoencoder
decoded = Dense(500, activation='relu')(encoded)
decoded = Dense(500, activation='relu')(decoded)
decoded = Dense(2000, activation='relu')(decoded)
decoded = Dense(2352,)(decoded)
# this model maps an input to its reconstruction
autoencoder = Model(encoder_input, decoded)
```

Step 12. Fitting the autoencoder

```
|: # Fitting the model
autoencoder.compile(optimizer='adam', loss='mse')
autoencoder.fit(X_train, X_train, epochs=50, batch_size=28, validation_data=(X_test, X_test))
```

Step 13: Retrieving the encoded image from autoencoder.

```
39]: # retrieving the encoded training image from auto encoder
Encodeimagetrain=encoder.predict(X_train)
2]: # retrieving encoded image for testing images
Encodeimagetest=encoder.predict(X_test)
```

Step 14: Training the SVM classifier and predicting for training set of images.

```
4]: # Training the SVM classifier with encoder training images
from sklearn.svm import SVC
svm = SVC(kernel='rbf', probability=True, random_state=0, gamma=.01, C=1)
# fit model
SVMmodel=svm.fit(Encodeimagetrain, train_label)
```

5]: # Testing the SVM classifier with encoded image of test image array SVMPredict=SVMmodel.predict(Encodeimagetest)

Step 15. Accuracy of the Model

```
]: # Accuracy of the model
from sklearn.metrics import accuracy_score
# calculate accuracy
accuracy = accuracy_score(test_label, SVMPredict)
print('Model accuracy is: ', accuracy)
```

Step 16: Generating the confusion matrix.

```
: from sklearn.metrics import confusion_matrix,classification_report
matrix=confusion_matrix(test_label,SVMPredict, labels=['Any','Normal','Epidural','Subdural','Intraparenchyma
print(matrix)
```

Step 17. Classification report for precision, recall and f1 score

```
[48]: report = classification_report(test_label,SVMPredict,labels=['Any','Normal','Epidural','Subdural','Intraparenchymal','Intraventr'
print('Classification report : \n',report)
```

### 6 Executing the PCA and SVM Model

Step 1: Open the PCA and SVM.ipynb from Thesis code folder.

Step 2: Set the path in the code from where data labels and PNG images are read.



Step 3: Reading the train label file and creating a dataframe.

n [4]: import pandas as pd import numpy as np train\_df = pd.read\_csv(csv\_path) train\_df['filename'] = train\_df['ID'].apply(lambda st: "ID\_" + st.split('\_')[1] + ".png") train\_df['type'] = train\_df['ID'].apply(lambda st: st.split('\_')[2]) train\_df.head()

. .

Step 4: Creating a dataframe by checking the filename from labels dataframe and checking if the file exists for 50,000 images in the Image folder



Step 5: Splitting the label data frame created into test and train labels data frame.

```
In [93]: from sklearn.model_selection import train_test_split
train_label, test_label = train_test_split(sample_df, test_size=0.2)
print(train_label.shape)
print(test_label.shape)
arraylist1=list(train_label.index.values)
print(len(arraylist1))
arraylist2=list(test_label.index.values)
print(len(arraylist2))
```

Step 6: Import libraries for loading training image and convert to numpy array

```
: from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img import PIL
```

Step 6: After splitting the label dataframe, read filename and from numpy array for training images.

```
In [95]: from keras.preprocessing import image
all_images=[]
for i in arraylist1:
    file = train_label.loc[i]["filename"]
    if(os.path.isfile(imgpath+file)):
        img = image.load_img(imgpath+file, target_size=(128,128))
        img = image.img_to_array(img)
        all_images.append(img)
print(len(all_images))
print(len(train_label))
```

Trainimagearray=np.array(all\_images)

Step 7: Loading image for testing data and converting to numpy array

```
5]: test_images=[]
for j in arraylist2:
    file =test_label.loc[j]["filename"]
    if(os.path.isfile(imgpath+ file)):
        img = image.load_img(imgpath+file, target_size=(128,128))
        img = image.img_to_array(img)
        test_images.append(img)
    print(len(test_images))
    print(len(test_label))
```

In [99]: Testimagearray=np.array(test\_images)

Step 8: One hot encoding for train and test labels

```
#converting encoded data to categories, one hot encoded data provided in the datas
def label_img(label_arr):
   labels=[]
    for i in range(len(label_arr)):
        if (label_arr[i][0]==0):
            labels.append("Normal")
        elif (label_arr[i][1]==1):
           labels.append("Epidural")
        elif (label_arr[i][2]==1):
            labels.append("Intraparenchymal")
        elif (label_arr[i][3]==1):
            labels.append("Intraventricular")
        elif (label_arr[i][4]==1):
            labels.append("Subarachnoid")
        else:
            labels.append("Subdural")
    return labels
```

```
train_label=np.array(train_label)
train_label= label_img(train_label)
```

Step 9: Reshaping to one dimension vector for 128\*128\*3= 49152



Step 10:

Applying the model:

```
: from sklearn.decomposition import PCA
pca = PCA(n_components=300)
pca.fit(Trainimagearray)
```

: data\_pca = pca.transform(Trainimagearray)
 test\_pca=pca.transform(Testimagearray)

```
: test_label=np.array(test_label)
  train_label=np.array(train_label)
```

```
: train_label
from sklearn.svm import SVC
svm = SVC(kernel='linear', probability=True, random_state=42)
# fit model
SVMmodel=svm.fit(data_pca, train_label)
```

: SVMPredict=SVMmodel.predict(test\_pca)

#### Step 11: Accuracy, confusion matrix and classification report:

```
SWMPredict=SVMmodel.predict(test_pca)

from sklearn.metrics import accuracy_score
# calculate accuracy
accuracy = accuracy score(test_label, SVMPredict)
print('Model accuracy is: ', accuracy)

from sklearn.metrics import confusion_matrix,classification_report
matrix=confusion_matrix(test_label, SVMPredict, labels=['Normal','Epidural','Subdural','Intraparenchymal','Intraventricular','Sub
print(matrix)

report = classification_report(test_label,SVMPredict, labels=['Normal','Epidural','Subdural','Intraparenchymal','Intraventricular
print('Classification report : \n',report)
```

### 7 Executing CNN Model

Step 1: Repeat step 1 to step 7 from previous section 5.

Step 2: Perform one hot encoding for labels.

```
In [11]:
         #converting encoded data to categories, one hot encoded data provided in the dataset
         import pdb
         def label_img(label_arr):
             labels=[]
             for i in range(len(label_arr)):
                 if (label_arr[i][0]==0):
                     labels.append(0)
                  elif (label_arr[i][1]==1):
                     labels.append(1)
                  elif (label_arr[i][2]==1):
                     labels.append(2)
                 elif (label_arr[i][3]==1):
                     labels.append(3)
                 elif (label_arr[i][4]==1):
                     labels.append(4)
                  else:
                      labels.append(5)
             return labels
In [12]: import sys
         test_label= label_img(np.array(test_label))
         train_label= label_img(np.array(train_label))
         train_label=np.array(train_label)
         test_label=np.array(test_label)
```

Step 3: Normalize the pixels in an image.



Step 4: import libraries for CNN model

: # import libraries for CNN model from keras.layers import MaxPool2D import keras from keras.layers import MaxPooling2D from keras.models import Sequential from tensorflow import keras from keras.layers import Conv2D

Step 5: Building the CNN Model.

```
:
  #Building the model
  model = Sequential([
      Conv2D(16, (3, 3), activation='sigmoid', input_shape=(224,224, 3)),
      MaxPooling2D(2, 2),
      Conv2D(32, (3, 3), activation='sigmoid'),
      MaxPooling2D(2, 2),
      Conv2D(64, (3, 3), activation='sigmoid'),
      Conv2D(64, (3, 3), activation='sigmoid'),
      MaxPooling2D(2, 2),
      Conv2D(128, (3, 3), activation='sigmoid'),
      Conv2D(128, (3, 3), activation='relu'),
      MaxPooling2D(2, 2),
      Dropout(0.50),
      Flatten(),
      Dense(512, activation='relu'),
      Dense(6, activation='softmax')
  ])
  model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics =['accuracy'])
```

Step 6: Fitting the model.

```
3]: # fitting the model
Model=model.fit(Trainimagearray, train_label, epochs=50,verbose=1, batch_size=28, validation_data=(Testimagearray, test_label))
```

## 8 Executing the Random Forest Classifier.

Step 1: Follow the steps from 1 to 9 used in Section 5. Step 2: Generate the random forest classifier

```
: from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=10)
rf.fit(Trainimagearray,train_label)
# Predictions on training and validation
y_pred_train = rf.predict(Testimagearray)
# training metrics
from sklearn.metrics import accuracy_score
# calculate accuracy
accuracy = accuracy_score(test_label, y_pred_train)
print('Model accuracy is: ', accuracy)
```

Step 4. Generate the confusion matrix using sk learn library

```
from sklearn.metrics import confusion_matrix,classification_report
matrix=confusion_matrix(test_label,y_pred, labels=['Normal','Epidural','Subdural','Intraparenchymal','Intraventricular','Subaracl
print(matrix)
```

Step 4. Create the classification report for evaluation metrics like precision, recall, f1score.

:	<pre>from sklearn.metrics import classification_report report = classification_report(test_label,y_pred_train,labels=['Any','Normal','Epidural','Subdural','Intraparenchymal','Intraven print('Classification report : \n',report)</pre>	-