Detection of Brain Tumors from Fusion of Different Imaging Techniques

Presented by:

- Soumyabroto Banerjee
- Sneha Roy

Under guidance of
Dr. Arpita Das

In today's world,

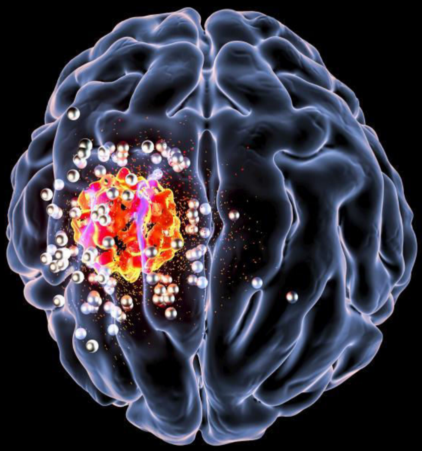
one of the reasons in rise in mortality amongst the people is *BRAIN TUMOR.*

So brain tumors have to be detected as early as possible

Here we present an efficient method for processing the MRI image as well as Brain tumor detection

What is Brain tumor?

- A brain tumor is a collection, or mass, of abnormal cells in the brain.This mass of abnormal cells grows within the skull due to which normal brain activity is hampered.



Brain tumors can be cancerous (malignant) or non cancerous (benign)

- Brain tumors are categorized as primary or secondary.

- ❑ Primary brain tumor - originates in the brain. Many primary brain tumors are benign.
- ❑ Secondary brain tumor - also known as a metastatic brain tumor, occurs when cancer cells spread to the brain from another organ, such as lung or breast.

MRI Images

- An **MRI** (magnetic resonance imaging) uses magnetic fields, not x-rays, to produce detailed images of the body. MRI can be used to measure the tumor's size.
- MRIs create more detailed pictures than CT scans and are the preferred way to diagnose a brain tumor.

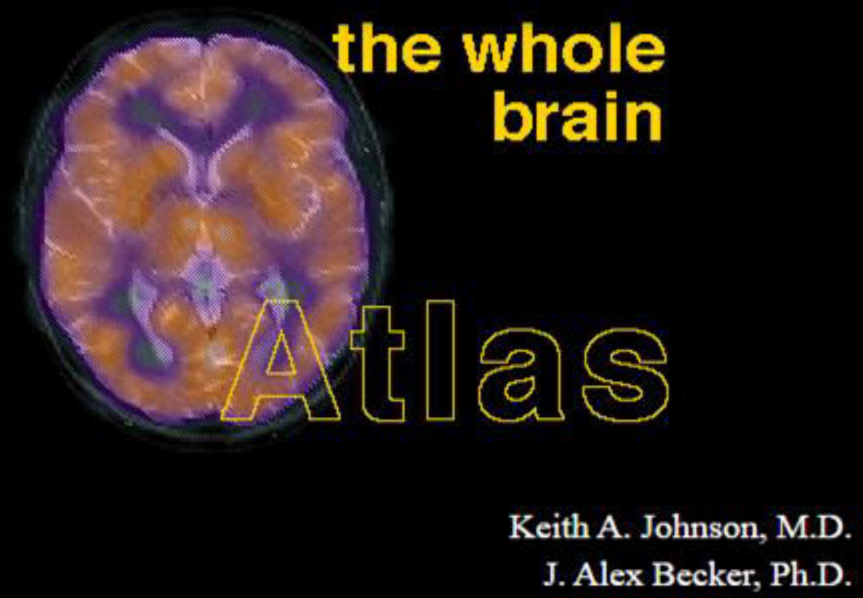
There are different types of MRI -

- Intravenous (IV) gadolinium-enhanced MRI is typically used to help create a clearer picture of a brain tumor.
- An MRI technique called "diffusion weighted imaging" helps show the cellular structure of the brain.
- Another technique called "perfusion imaging" shows how much blood is reaching the tumor.

These methods may help doctors predict how well treatment will work.

Data Preparation

- The Data Images has been collected from
  - http://www.med.harvard.edu/AANLIB/o



Data Preparation

- We concentrate our study on Glioma

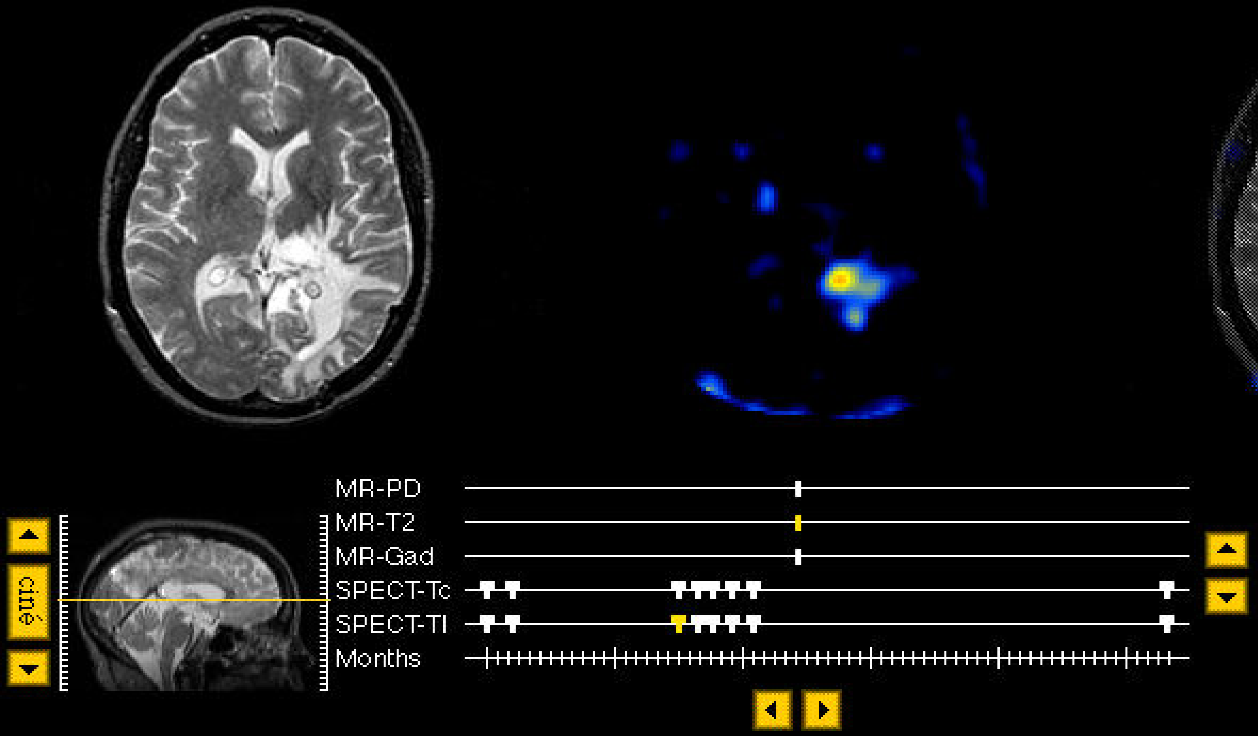
- **Neoplastic Disease (brain tumor):**
  - Glioma, TlTc-SPECT with a Tour
  - Glioma, FDG-PET
  - Glioma, FDG-PET

Glioma

Glioma is a type of tumor that occurs in the brain and spinal cord.

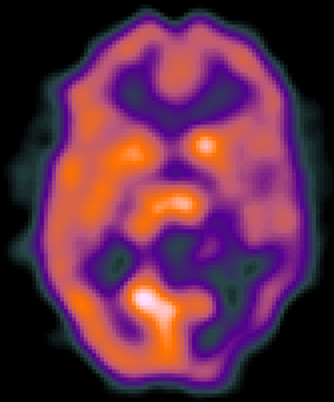
Data Preparation

- The downloaded Images have been collected by slicing the Brain at various slices and by changing the month.

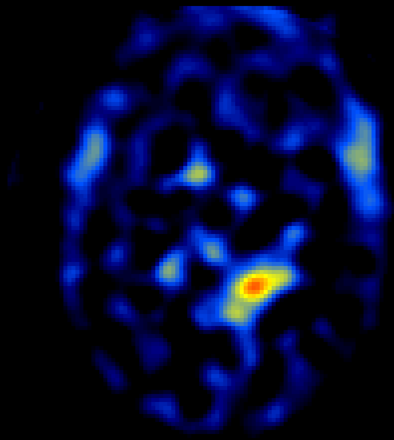


Data Preparation

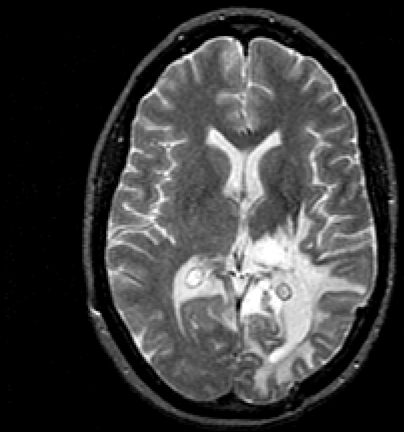
- We download the 3 Images primarily:
  - **MR-T2**
  - **SPECT-TC**
  - **SPECT-T1**



SPECT-TC



SPECT-T1



MR-T2

- **MR-T2 images -** 2 tissue types are bright – **FAT** and **WATER**

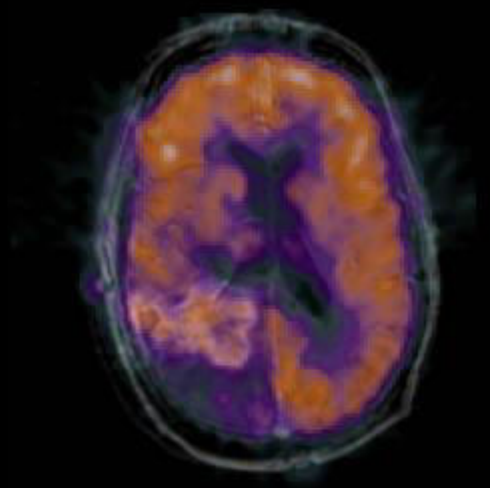
- **SPECT images - Single-photon emission computed tomography** a nuclear medicine tomographic

Imaging technique using gamma rays.

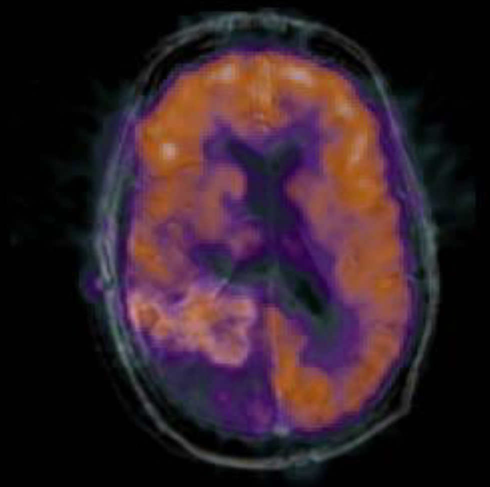
Data Preparation

- We blend the Images as shown in the above Site to confirm the coefficients.

Coefficients:
Image 1 =0.5
Image 2 =0.5



Downloaded Image



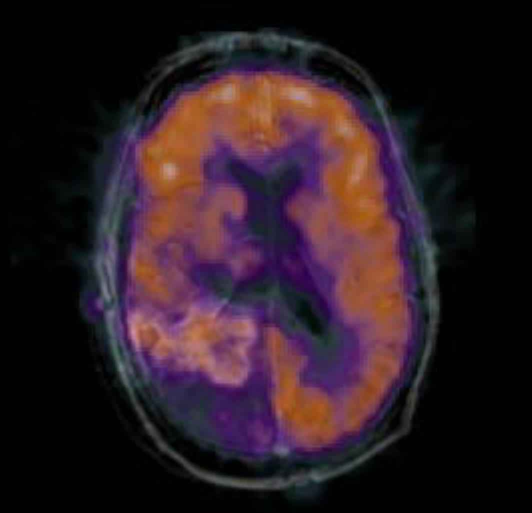
Blended Image

Data Preparation

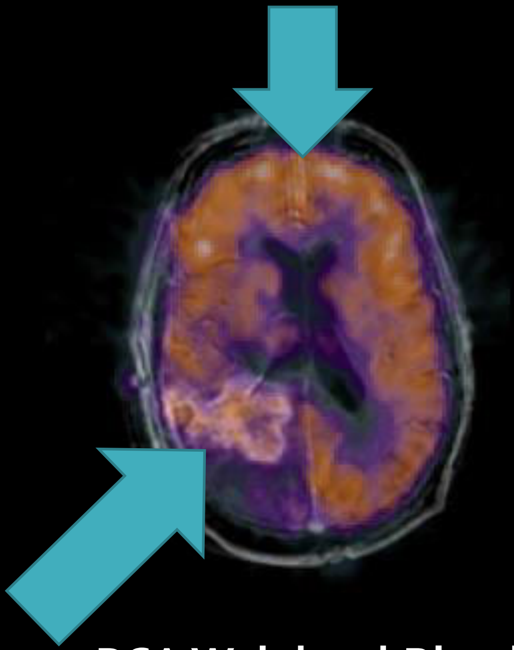
- What could be a better coefficient for Blending?
  - *i.e., a better and scientific approach to form the blend by automatically weighting the images on the info they hold.*
  - *Solution: Use Principal Component Analysis (PCA) to extract the **Essential Information (Eigen Info)** from the Image Vectors and use the Weight Matrix Formed for the Blending.*

Data Preparation

Result Comparison



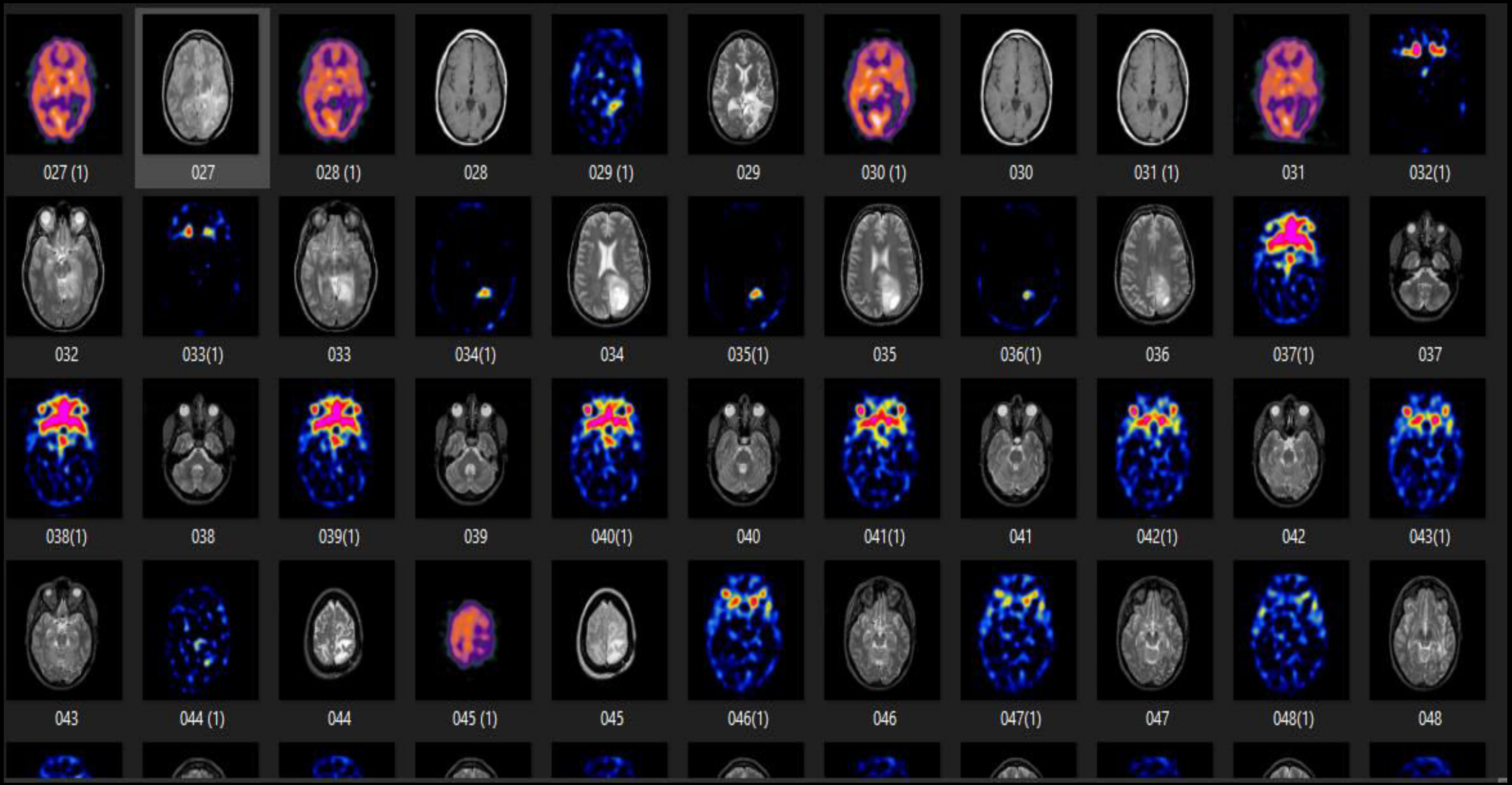
Auto 0.5 weighted Image



PCA Weighted Blend

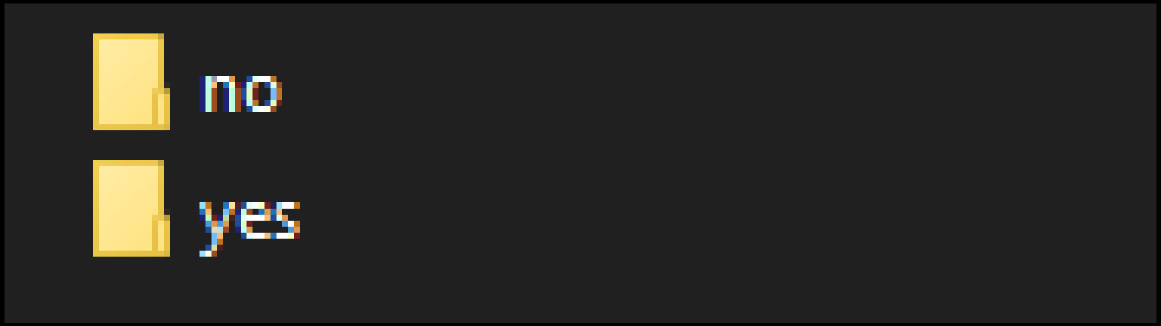
Data Preparation

- Due to lack of time we could manage to collect about 100 or so images from the Internet.



Data Preparation

- Next, we blend the Collected Images and Store them into two Folders
  - Yes (Indication Presence of Brain Tumour)
  - No (Indication No Presence of Brain Tumour)

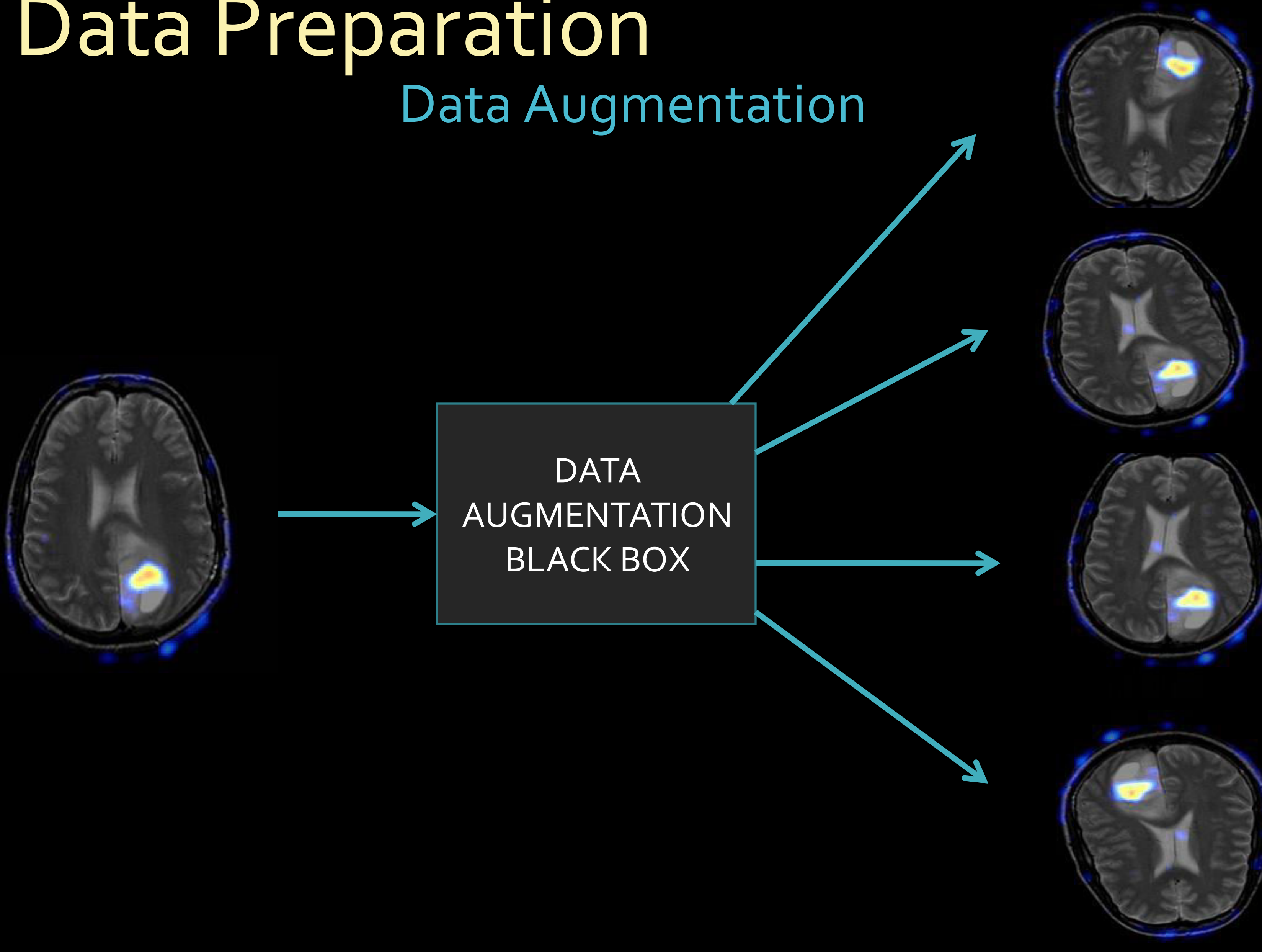


Data Preparation

- We then Hand Segregate the Images into the **Yes** and **No** Folders by identifying Tumour or No Tumour in the Images.
  - The Total Images in **YES** : 25
  - The Total Images in **NO** : 35
    - **We find the Dataset is very small. And the YES Folder contains fewer images.**
- We Increase our Dataset by Data Augmentation.
  - Controlled Random Rotations and Alterations to the Image Matrix
    - Without Damaging the Image.

Data Preparation

Data Augmentation



Data Preparation

- Total Images after Data Augmentation :
  - YES : 734
  - NO : 476

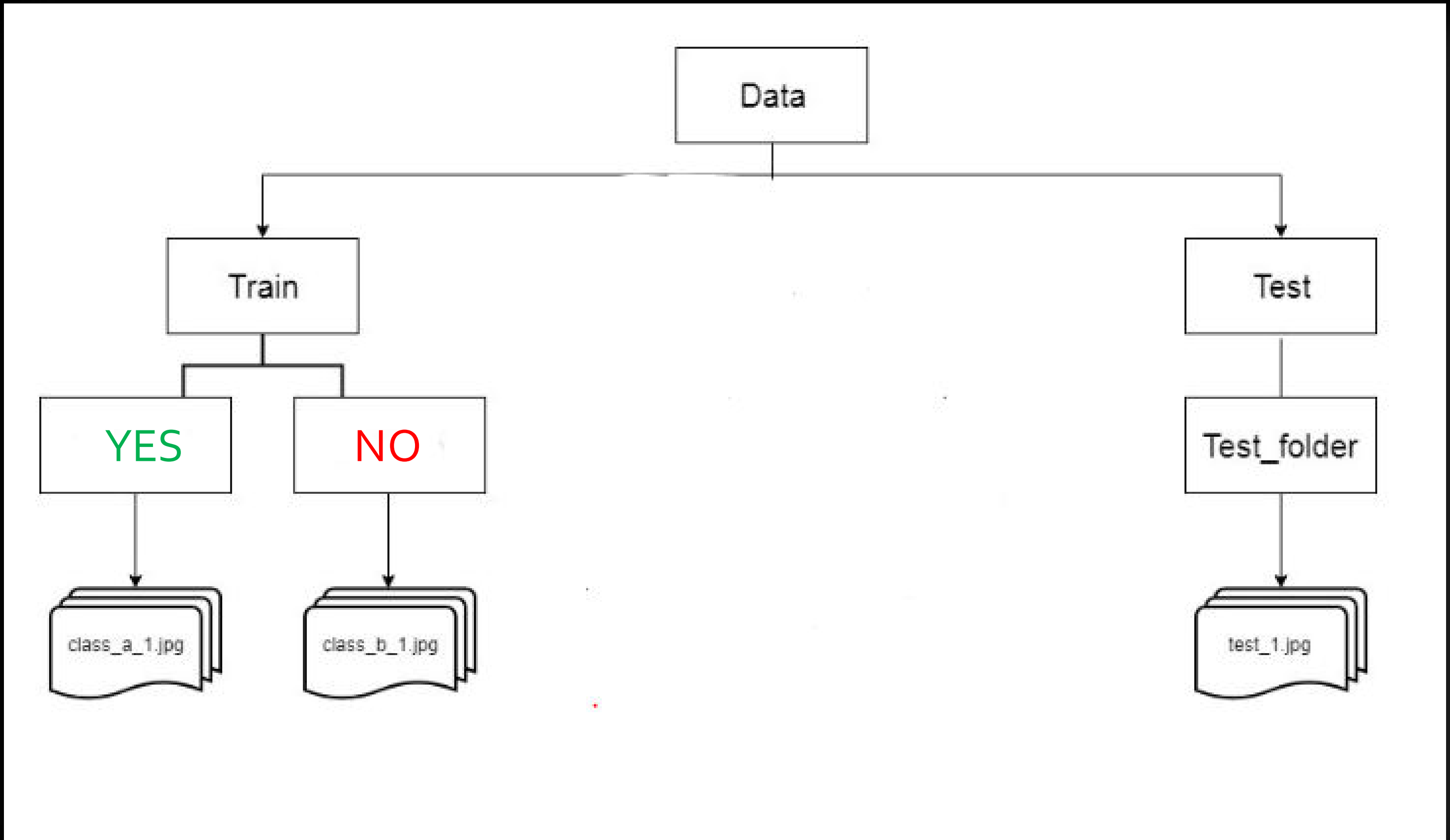
```
{'no': 0, 'yes': 1} (array([0, 1], dtype=int32), array([476, 734]))
```

These are split into Training and Testing separately :

Train : 70%

Test : 30%

Final Dataset



Classification

BEGIN ->

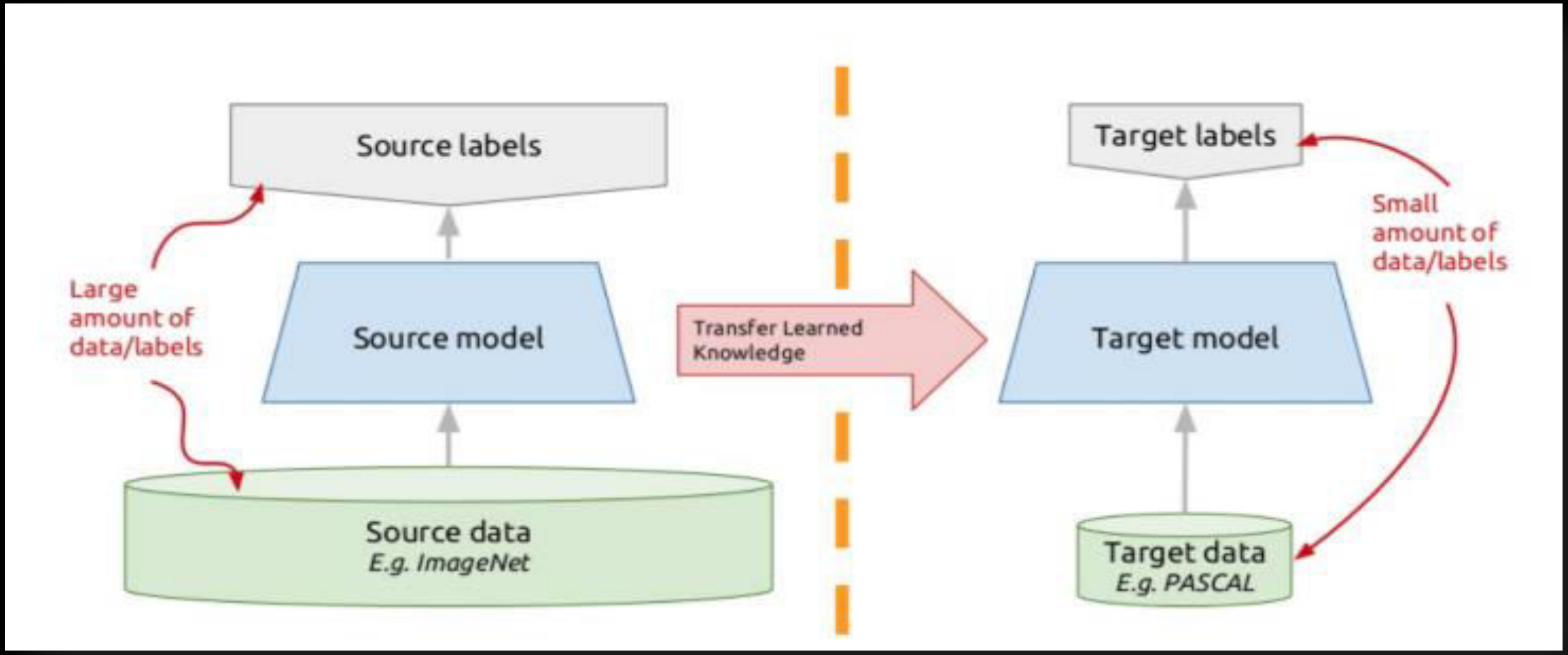
Technique

- We have used a **Deep Neural Network** consisting of **CONVNETS and FCNs**.

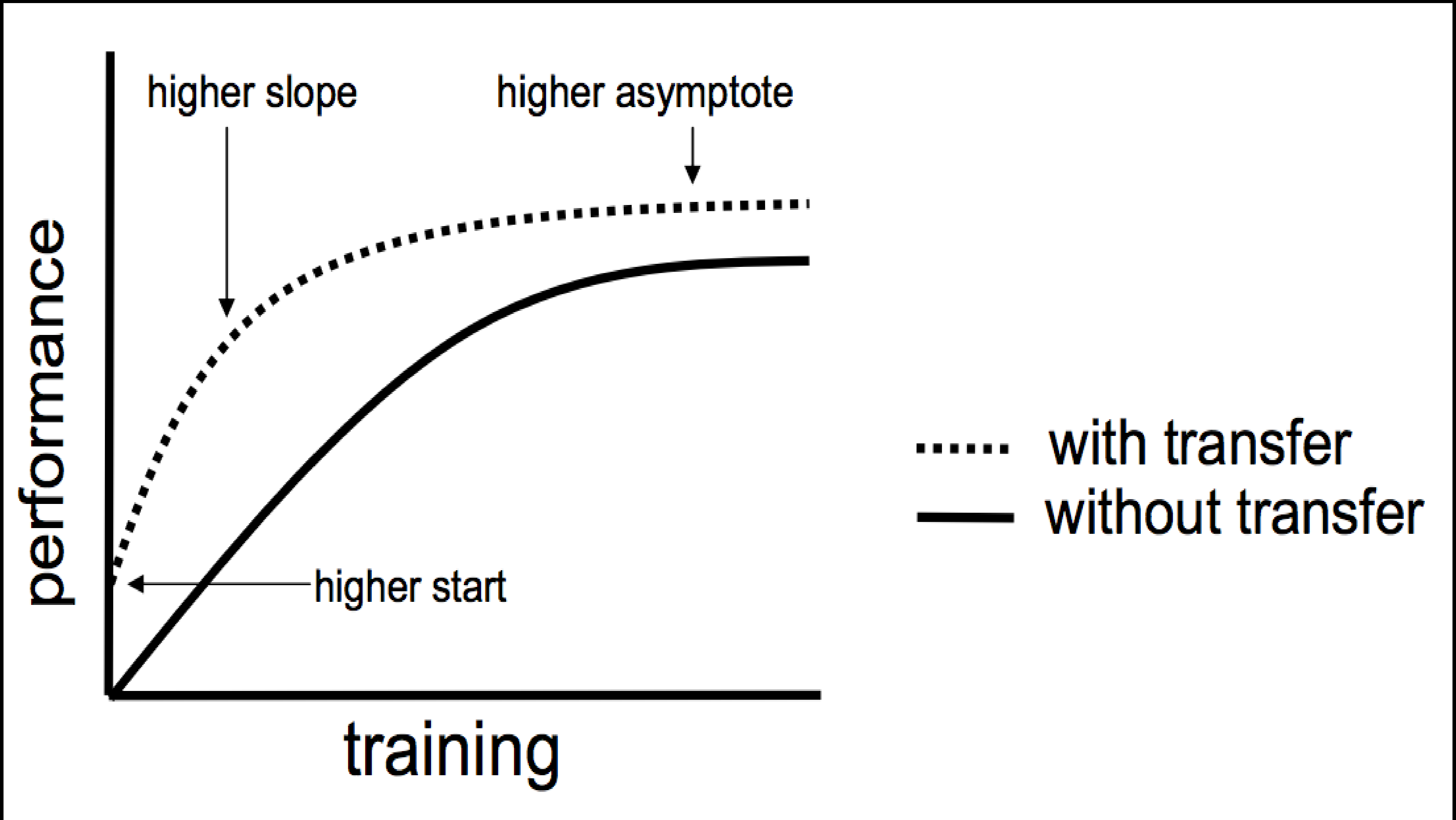
- The Technique we have used is called **Transfer Learning**

What is Transfer Learning ?

- Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.



Growth perspective of TL



Architecture used ?

VGG-16

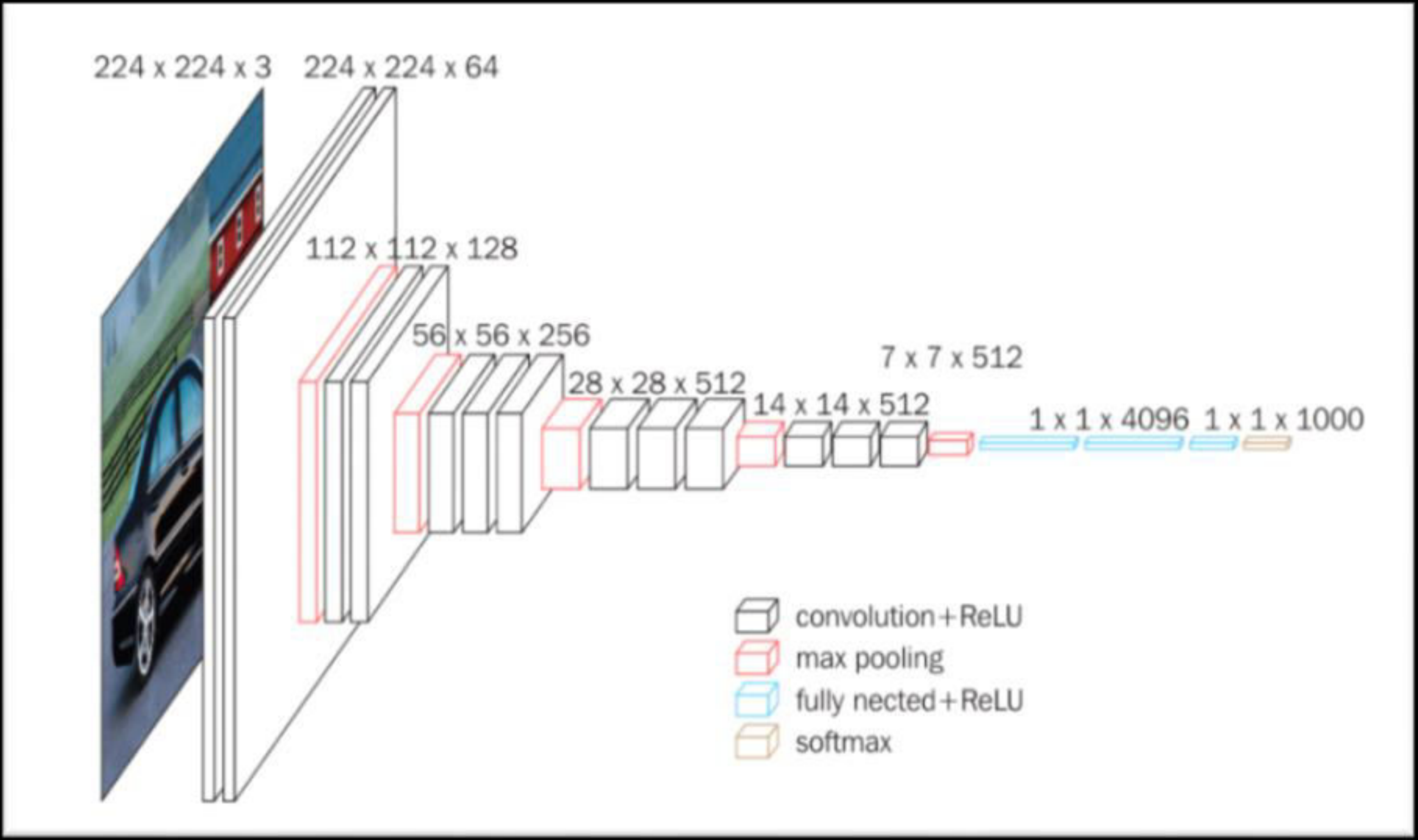
VGG16 is a Convolutional Neural Network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition".

The model achieves **92.7% top-5 test accuracy** in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014.

It makes the improvement over Alex Net by replacing large kernel-sized filters (11 and 5 in the first and second Convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another.

VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's.

How does it look like ?

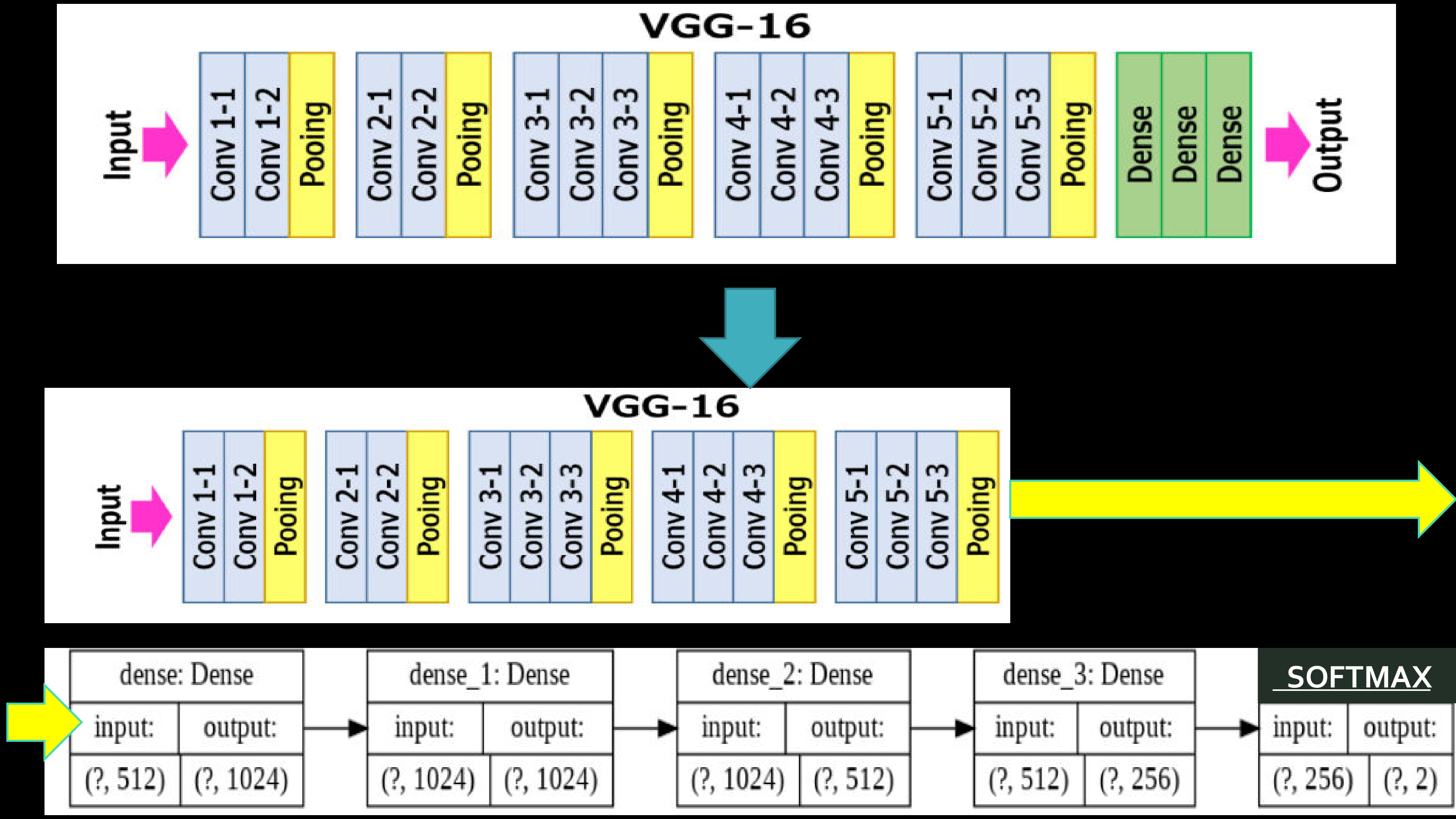


Modified Transfer Learning (Our Approach)

- Training over those huge number of Layers is both time consuming and redundant for our small task.

  - **Solution**: We create our own TOP, *i.e., the Output Layer including the Dense Layers(FCN) are all replaced by a Neural Network we designed.*

Modified Architecture



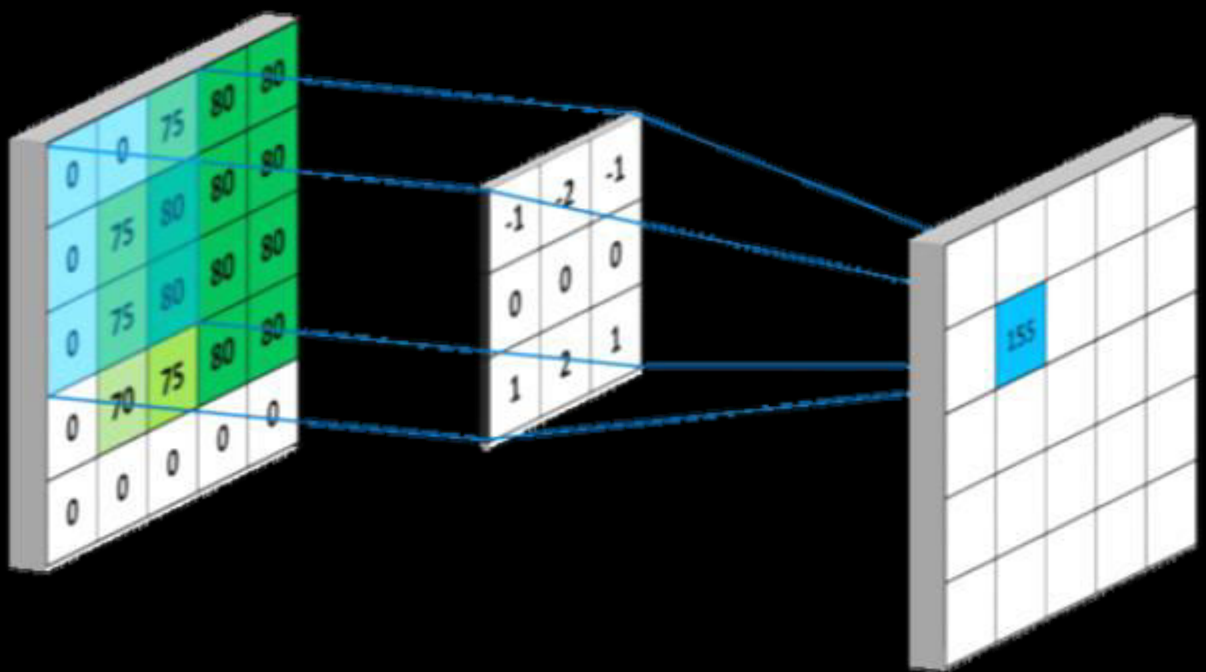
Let's Look into these Layers

→

Convolutional Neural Network

- A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.
- The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.
- The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

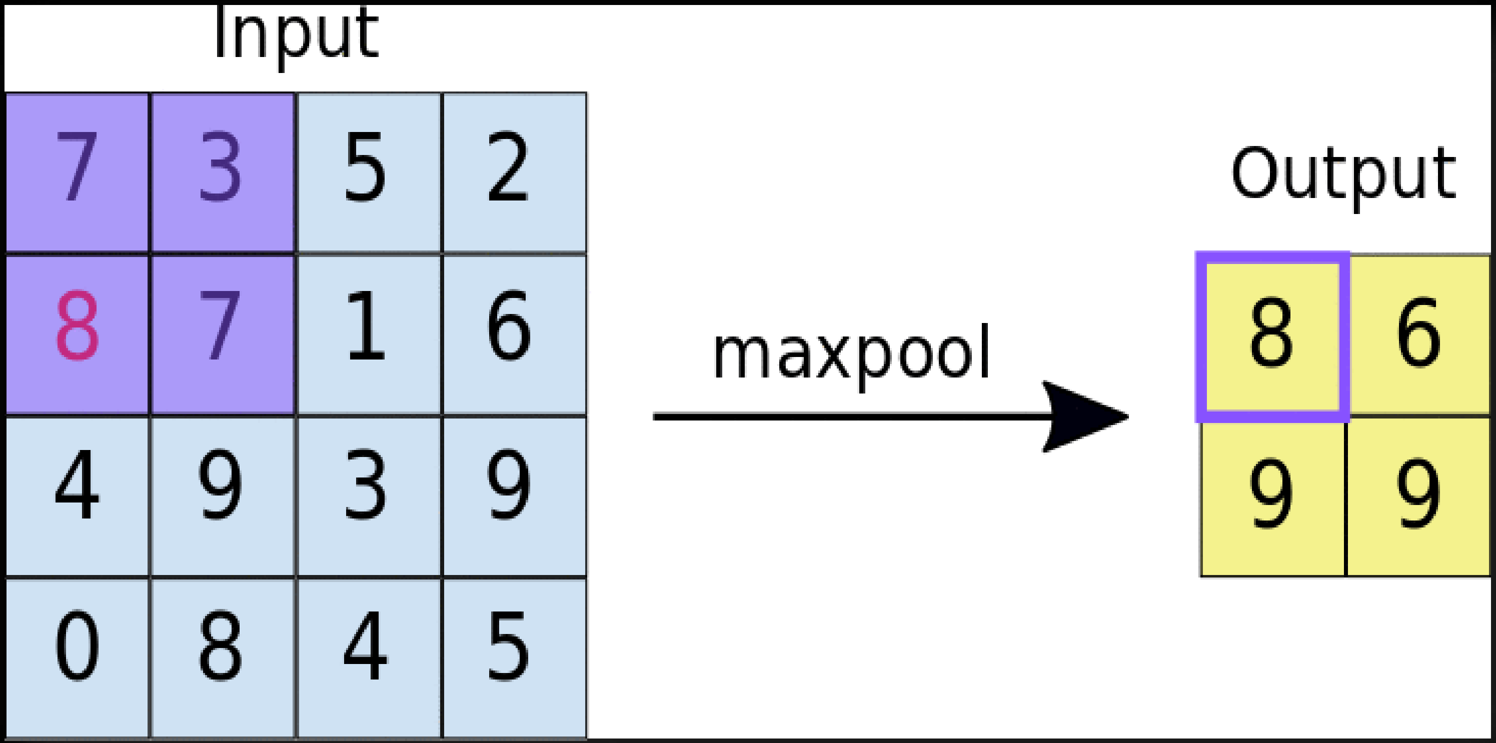
Convolutional Neural Network



Pooling (Max-Pool)

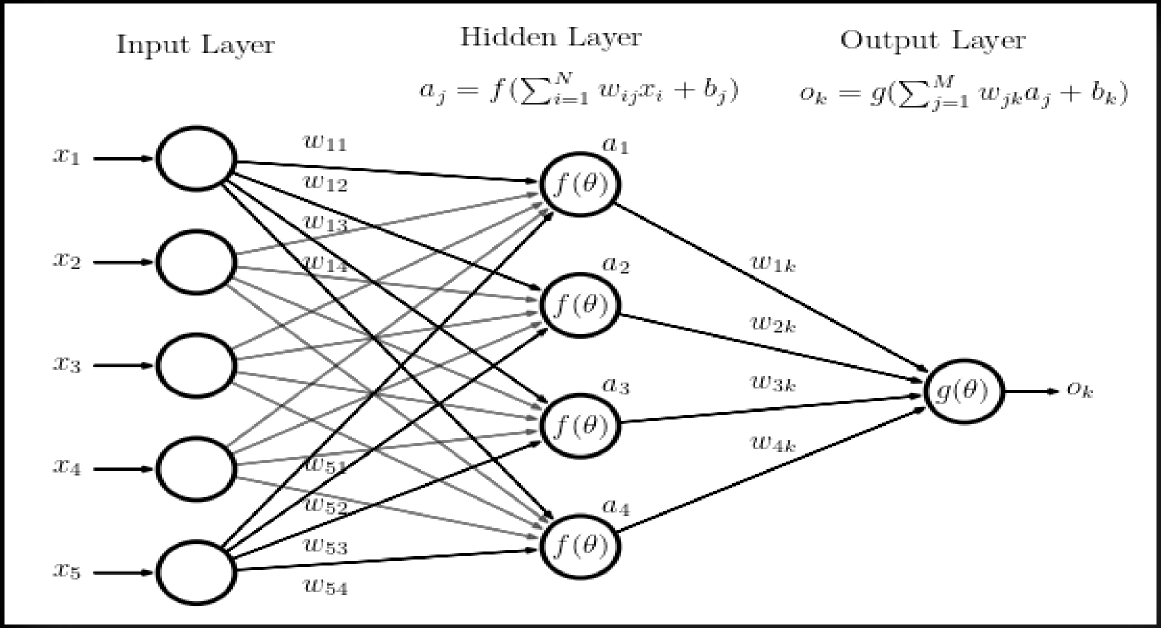
- Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to **decrease the computational power required to process the data** through dimensionality reduction. Furthermore, it is useful for **extracting dominant features** which are rotational and positional invariant, thus maintaining the process of effectively training of the model.
- Max Pooling also performs as a **Noise Suppressant**. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction

Max Pooling



Fully Connected Networks

- **Fully connected** layers **connect** every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural **network** (MLP).



Forming the Model

→

How we do it ?

- The Entire Model is designed in Keras.

  (NOT tf.keras and definitely not tf 2.0)
- Keras is a High Level API which uses Tensorflow in the backend.
- VGG-16 is a sequential() model

The Algorithm bit

```
x = vgg.output
```

```
[ ] x = Dense(1024,activation='relu')(x)
    x = Dense(1024,activation='relu')(x)
    x = Dense(512, activation='relu')(x)
    x= Dense(256,activation='relu')(x)
```

```
[ ] preds = Dense(2,activation='softmax')(x)
```

```
[ ] model = Model(inputs = vgg.input,outputs=preds)
```

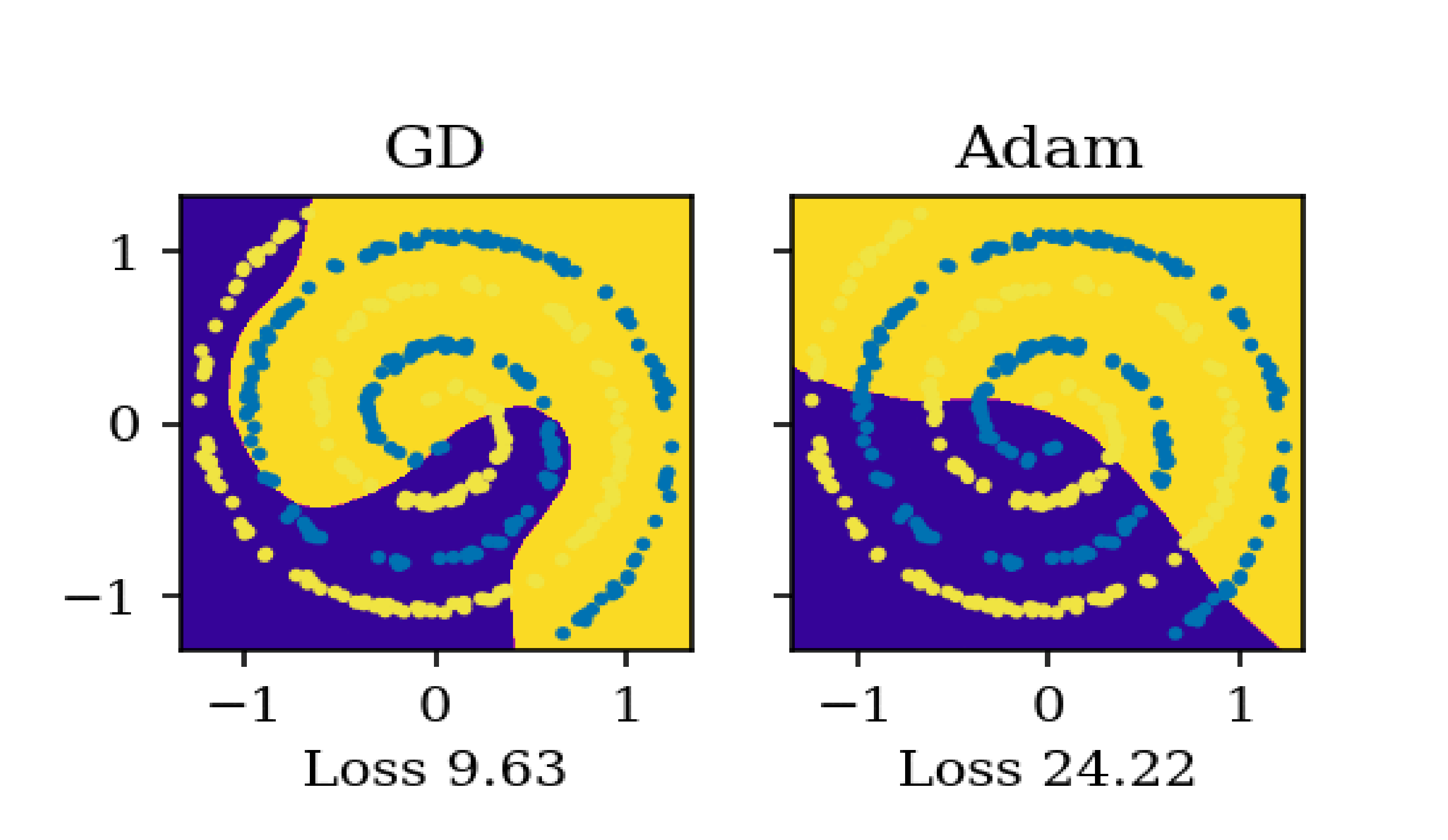
The Entire Model

```
model.compile(optimizer='Adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Categorical Cross Entropy:

$$-(y\log(p) + (1-y)\log(1-p))$$

What is *Adam* optimizer ?

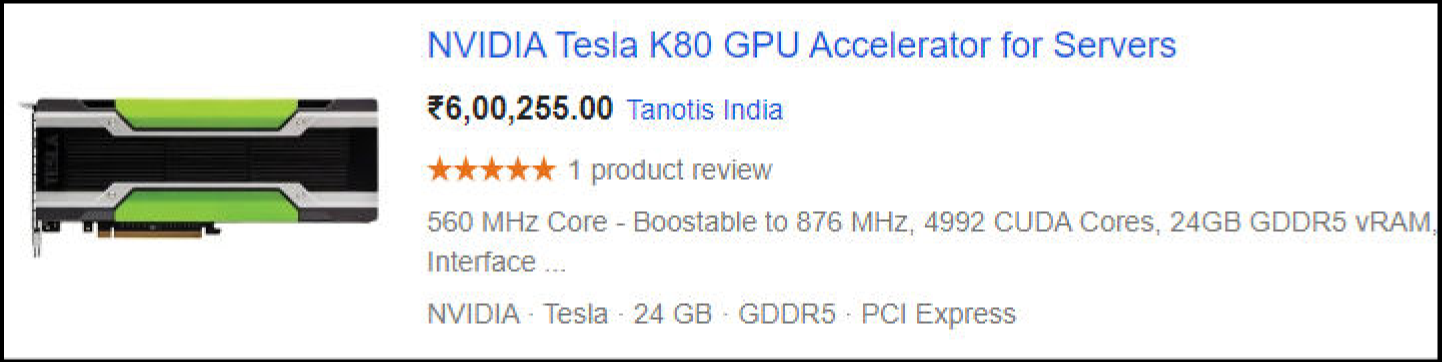


Now the Training Phase

->

Training our Model

- The Training is done in Google Colab using Tesla K80 GPU.



- The Notebook is connected to a Hosted Runtime and provides 12 hours of usage for free.

Training

- We upload the Dataset in Google Drive and Read the Data from there using a DataGenerator.

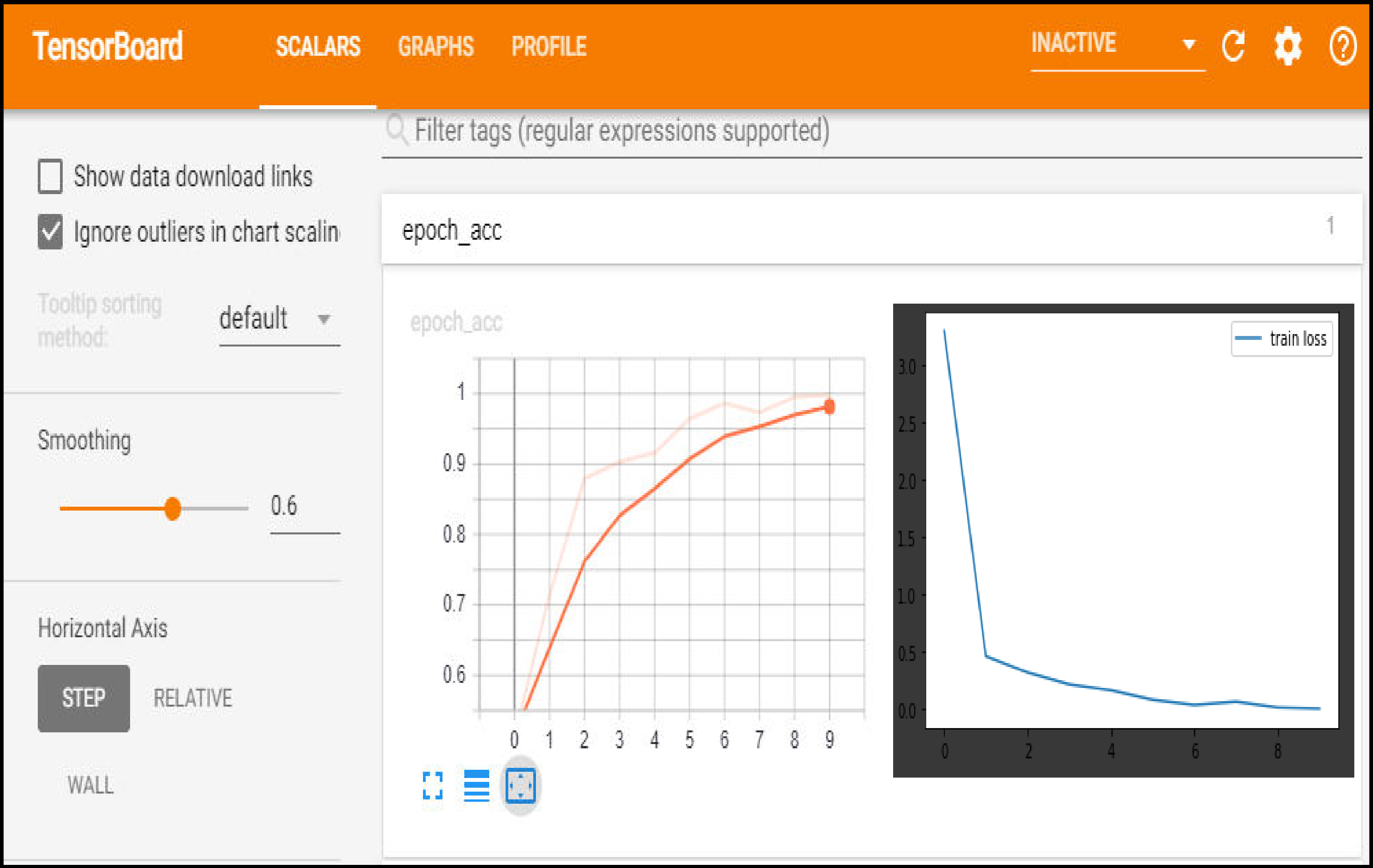
```
train_datagen=ImageDataGenerator(preprocessing_function=preprocess_input) #included in our dependencies

train_generator=train_datagen.flow_from_directory('/content/drive/My Drive/Dataset/Augmented Data/Train/'
                                                 target_size=(224,224),
                                                 color_mode='rgb',
                                                 batch_size=55,
                                                 class_mode='categorical',
                                                 shuffle=False)
```

- Similarly, we do the same for Test Data.

Training Flow

```
Epoch 1/10
22/22 [==============================] - 423s 19s/step - loss: 3.3038 - acc: 0.5132
Epoch 2/10
22/22 [==============================] - 7s 302ms/step - loss: 0.4621 - acc: 0.7132
Epoch 3/10
22/22 [==============================] - 7s 310ms/step - loss: 0.3208 - acc: 0.8793
Epoch 4/10
22/22 [==============================] - 7s 319ms/step - loss: 0.2179 - acc: 0.9033
Epoch 5/10
22/22 [==============================] - 7s 321ms/step - loss: 0.1653 - acc: 0.9157
Epoch 6/10
22/22 [==============================] - 7s 312ms/step - loss: 0.0829 - acc: 0.9645
Epoch 7/10
22/22 [==============================] - 7s 305ms/step - loss: 0.0382 - acc: 0.9860
Epoch 8/10
22/22 [==============================] - 7s 303ms/step - loss: 0.0667 - acc: 0.9736
Epoch 9/10
22/22 [==============================] - 7s 299ms/step - loss: 0.0151 - acc: 0.9950
Epoch 10/10
22/22 [==============================] - 7s 297ms/step - loss: 0.0053 - acc: 0.9983
```

Loss and Accuracy Plot



Now the Test Phase

->

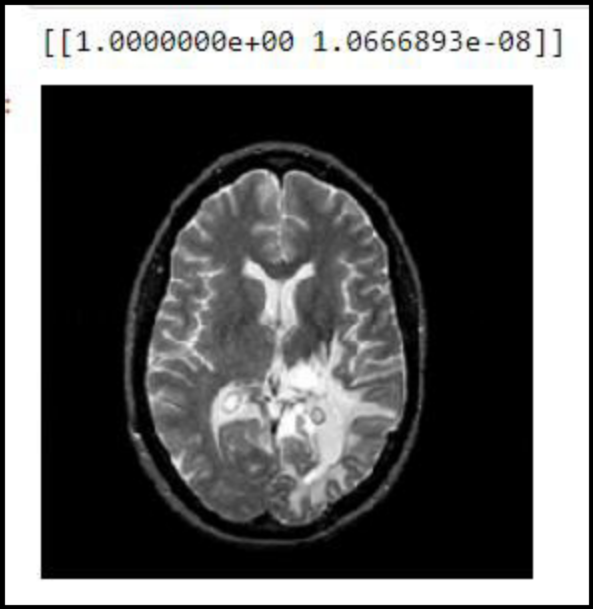
Testing

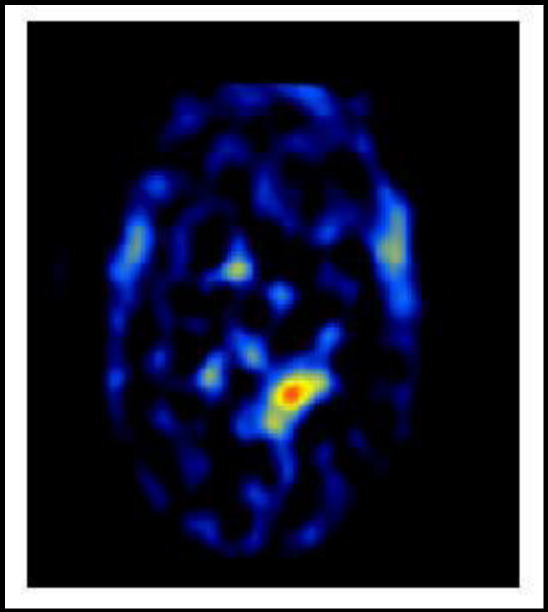
- We prepare the Data just like the Training Phase and ultimately use ImageDataGenerator to load the data.

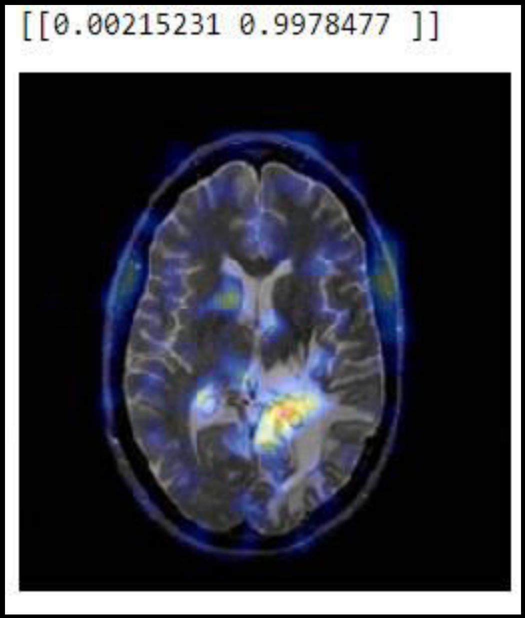
```
{'no': 0, 'yes': 1} (array([0, 1], dtype=int32), array([109, 126]))
```

YES : 126 Images
NO : 109 Images

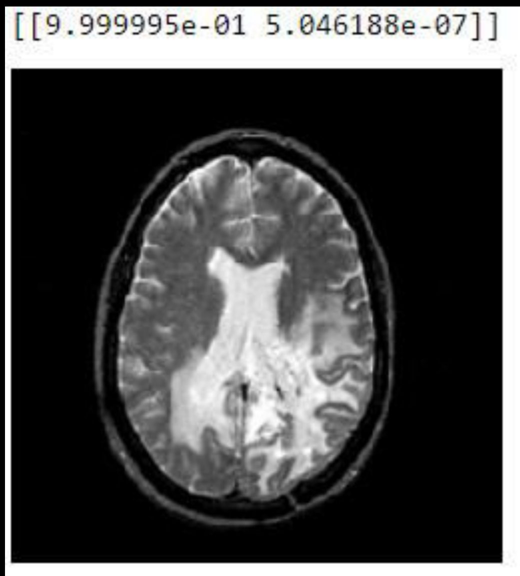
Results

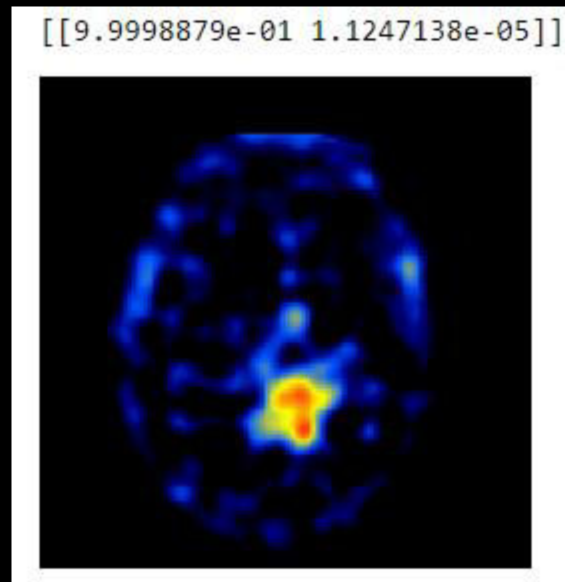


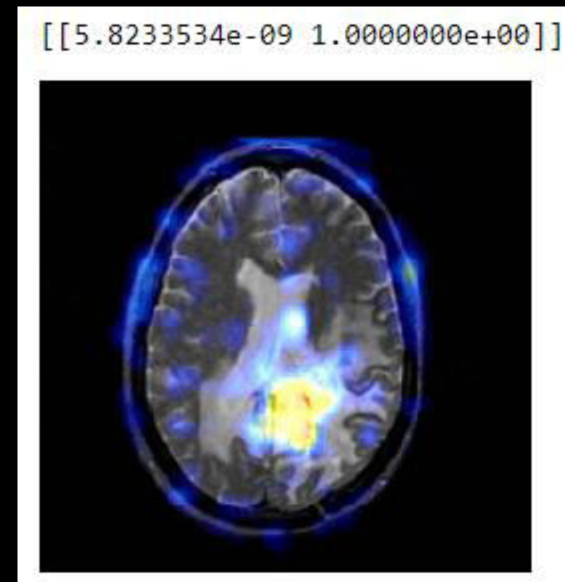




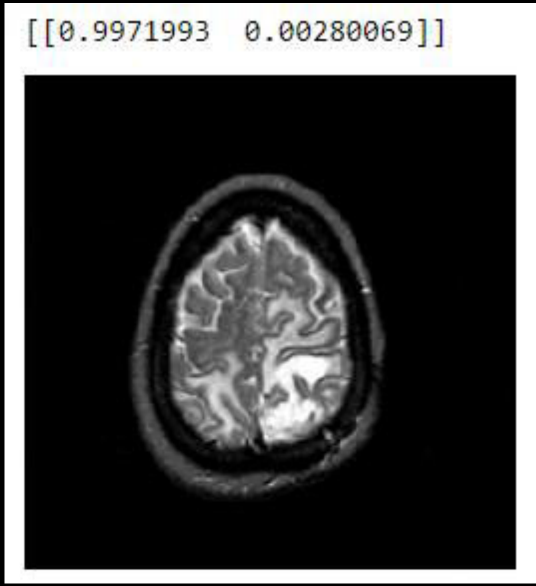
Results

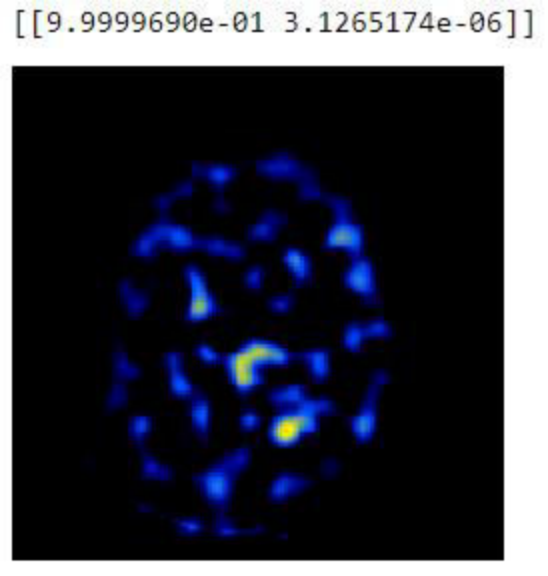

[[9.999995e-01 5.046188e-07]]

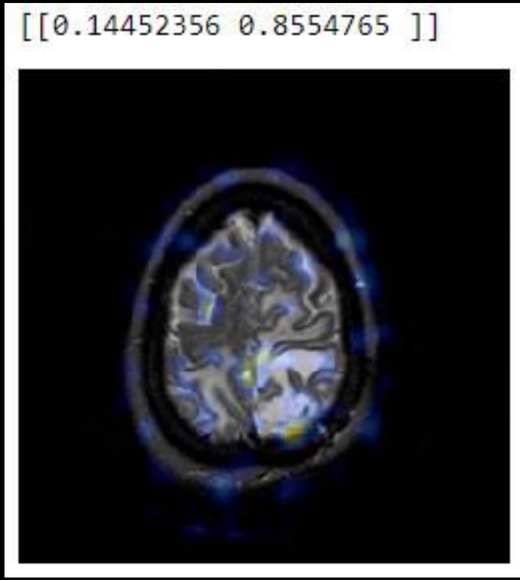

[[9.9998879e-01 1.1247138e-05]]


[[5.8233534e-09 1.0000000e+00]]

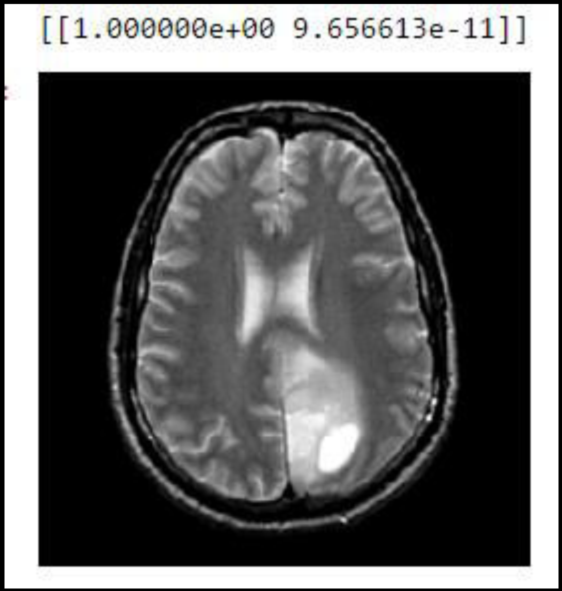
Results

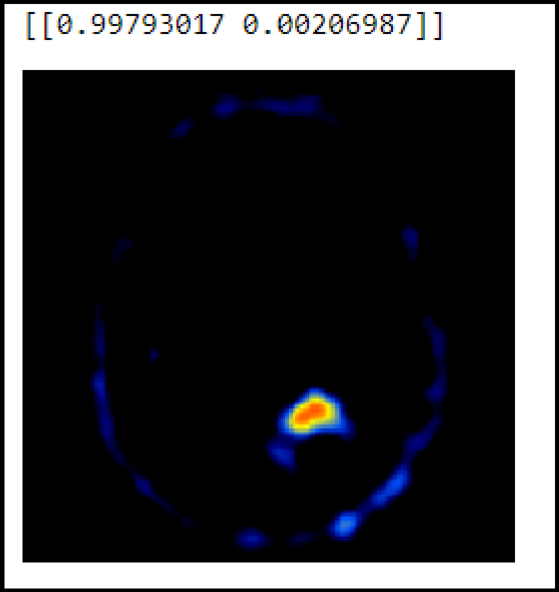

[[0.9971993 0.00280069]]

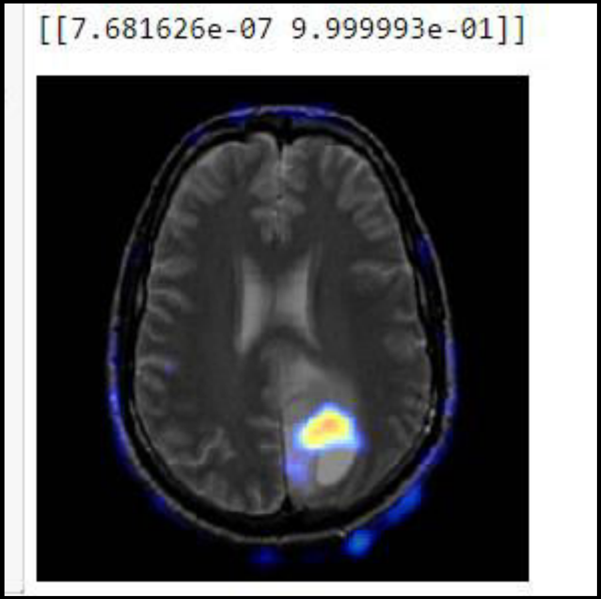

[[9.9999690e-01 3.1265174e-06]]


[[0.14452356 0.8554765]]

Results

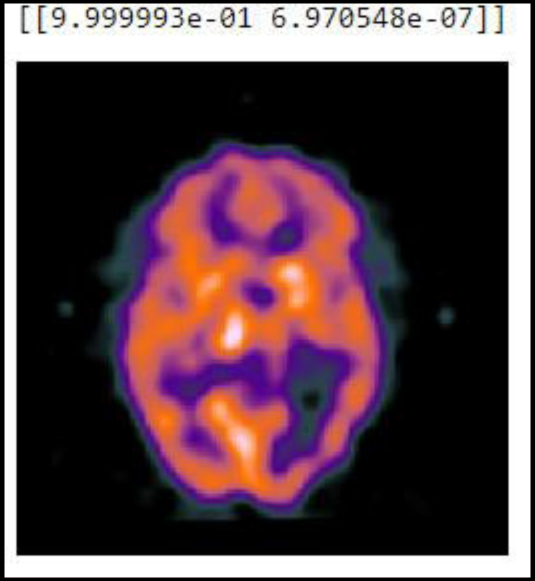

[[1.000000e+00 9.656613e-11]]

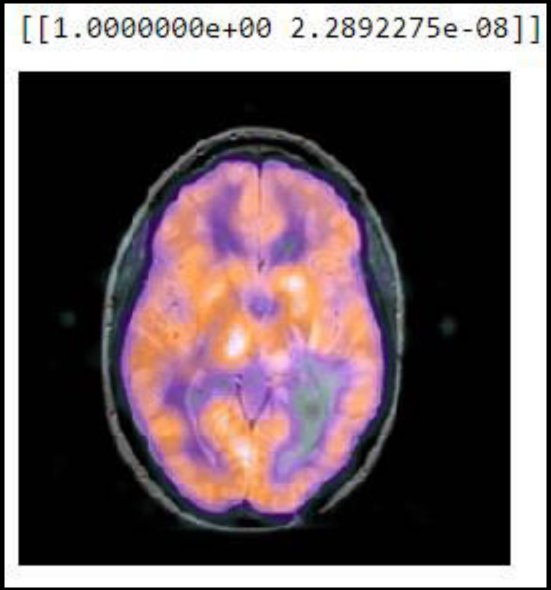

[[0.99793017 0.00206987]]


[[7.681626e-07 9.999993e-01]]

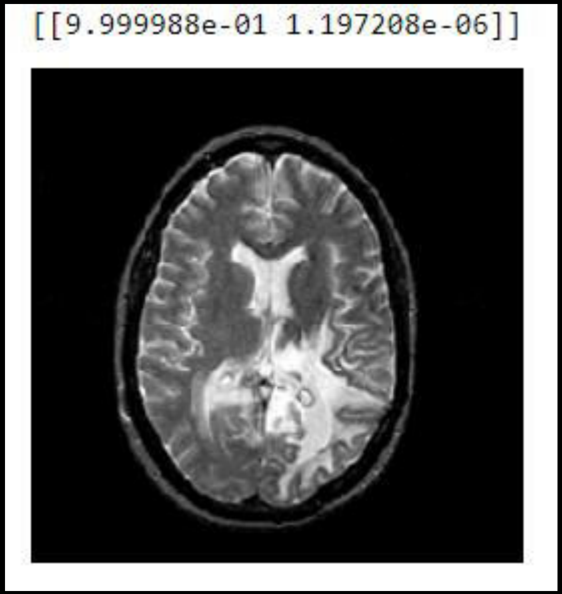
Results


[[4.3620853e-04 9.9956375e-01]]


[[9.999993e-01 6.970548e-07]]

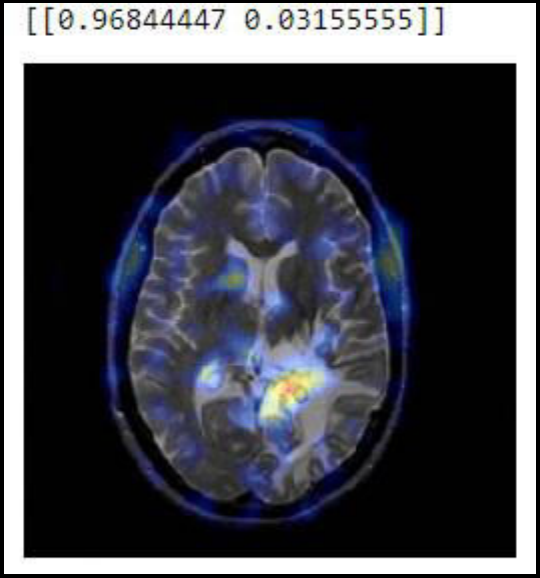

[[1.0000000e+00 2.2892275e-08]]

Results

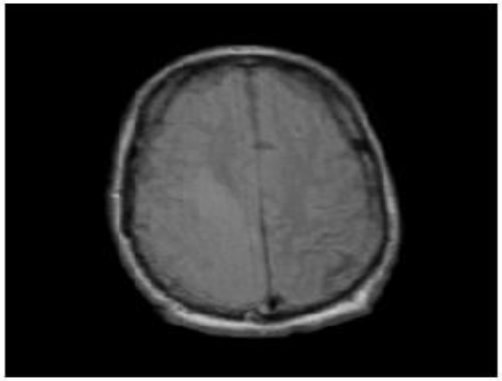

[[9.999988e-01 1.197208e-06]]


[[1.7385905e-06 9.9999821e-01]]

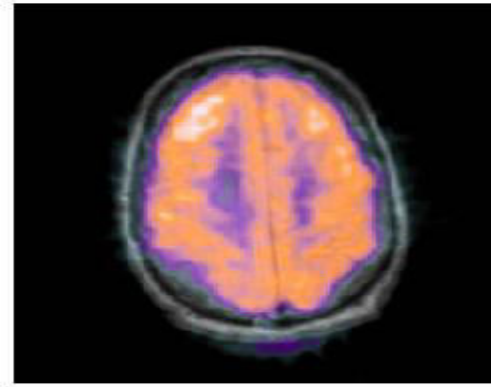

[[0.91874635 0.08125366]]


[[0.96844447 0.03155555]]

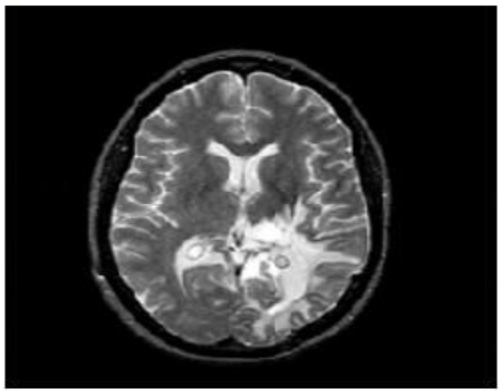
[[0.2709756 0.72902435]]



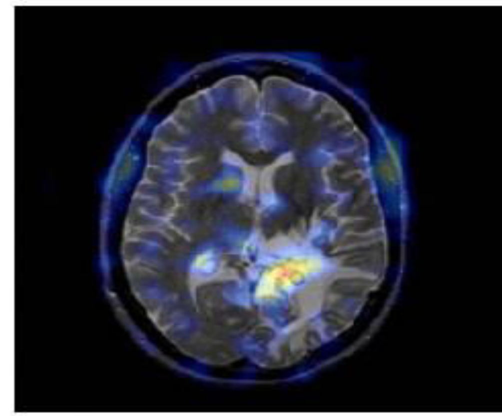
[[1.0000000e+00 2.8779454e-09]]



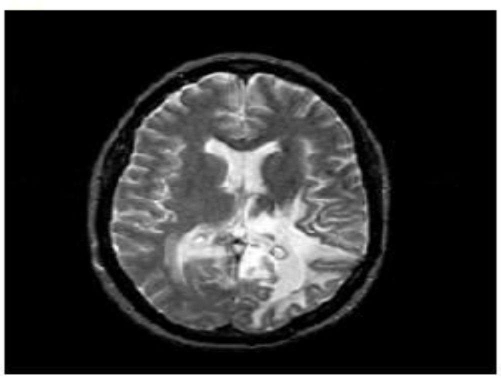
[[0.91874635 0.08125366]]



[[0.96844447 0.03155555]]



[[9.999988e-01 1.197208e-06]]



[[1.7385905e-06 9.9999821e-01]]

ROC-AUC Curve

(**Area Under the Receiver Operating Characteristics Curve**)

- AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. **ROC is a probability curve and AUC represents degree or measure of separability**. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. **By analogy, Higher the AUC, better the model is at distinguishing between patients with disease and no disease.**

- The ROC curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis.

How to evaluate this ?

- We have used ROC-AUC Curve for evaluation

- BUT… What is that ?
  - Let’s see

ROC-AUC Basic understanding

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

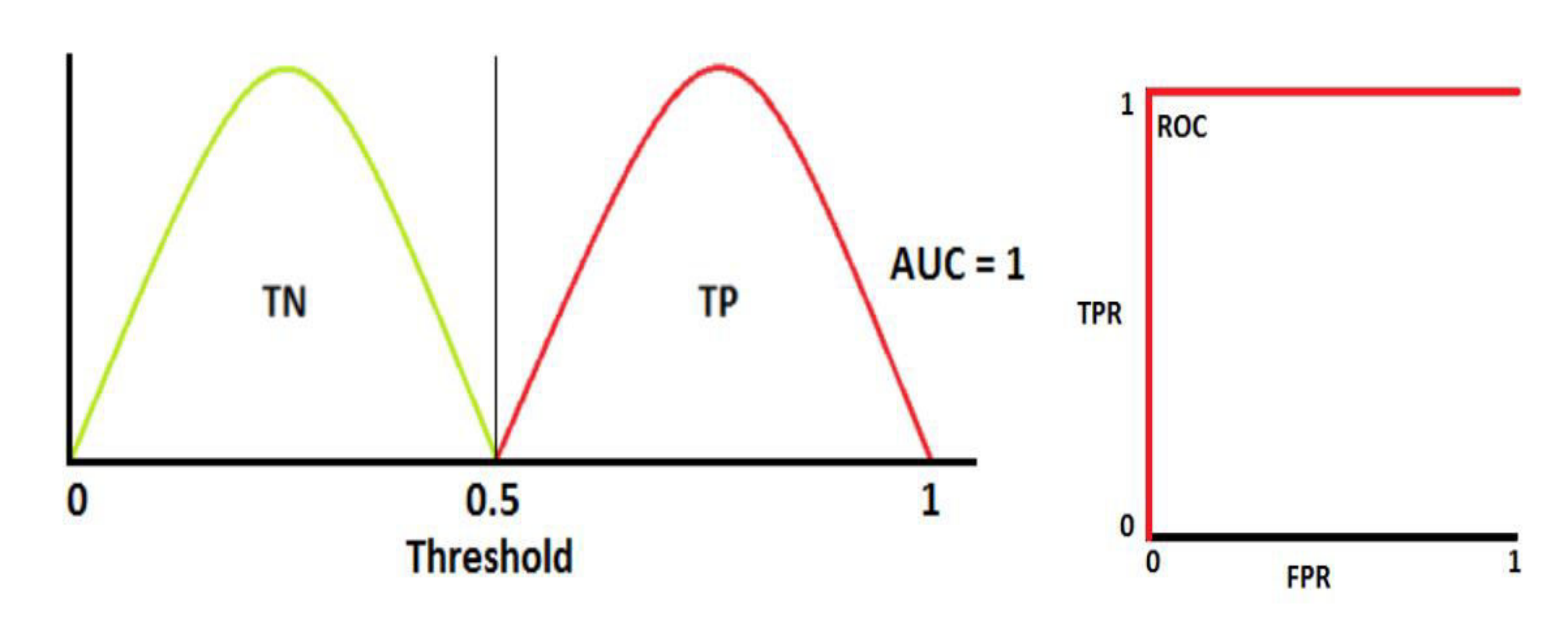
$$\text{FPR} = 1 - \text{Specificity} = \frac{FP}{TN + FP}$$

Sensitivity↑, Specificity↓ and Sensitivity↓, Specificity↑

TPR↑, FPR↑ and TPR↓, FPR↓

When we decrease the threshold, we get more positive values thus it increases the sensitivity and decreasing the specificity.
Similarly, when we increase the threshold, we get more negative values thus we get higher specificity and lower sensitivity.

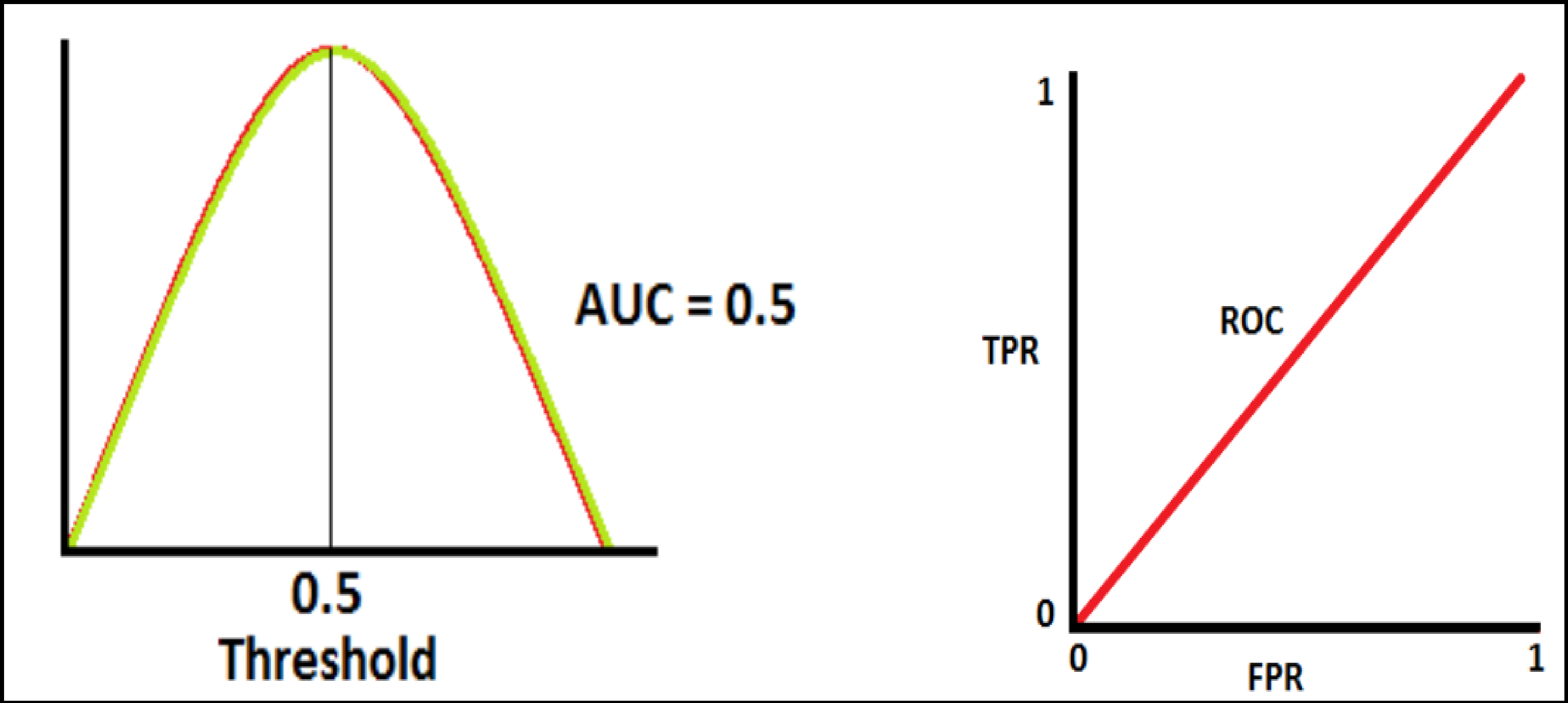
ROC-AUC Ideal Case



This is an ideal situation. When two curves don't overlap at all means model has an ideal measure of separability.

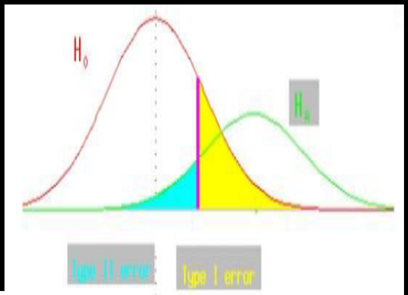
It is perfectly able to distinguish between positive class and negative class.

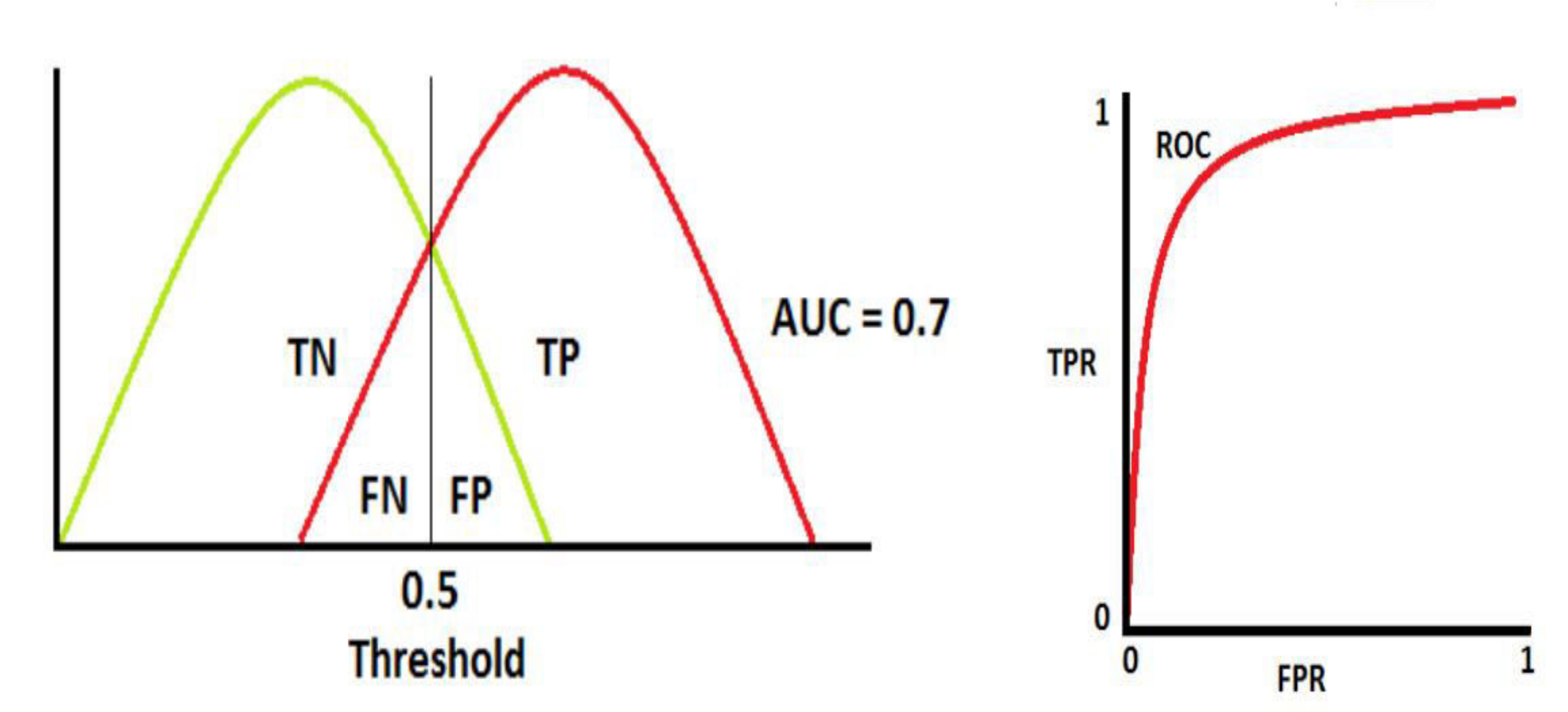
ROC-AUC Worst Case



This is the worst situation. When AUC is approximately 0.5, model has no discrimination capacity to distinguish between positive class and negative class.

ROC-AUC Practical Case

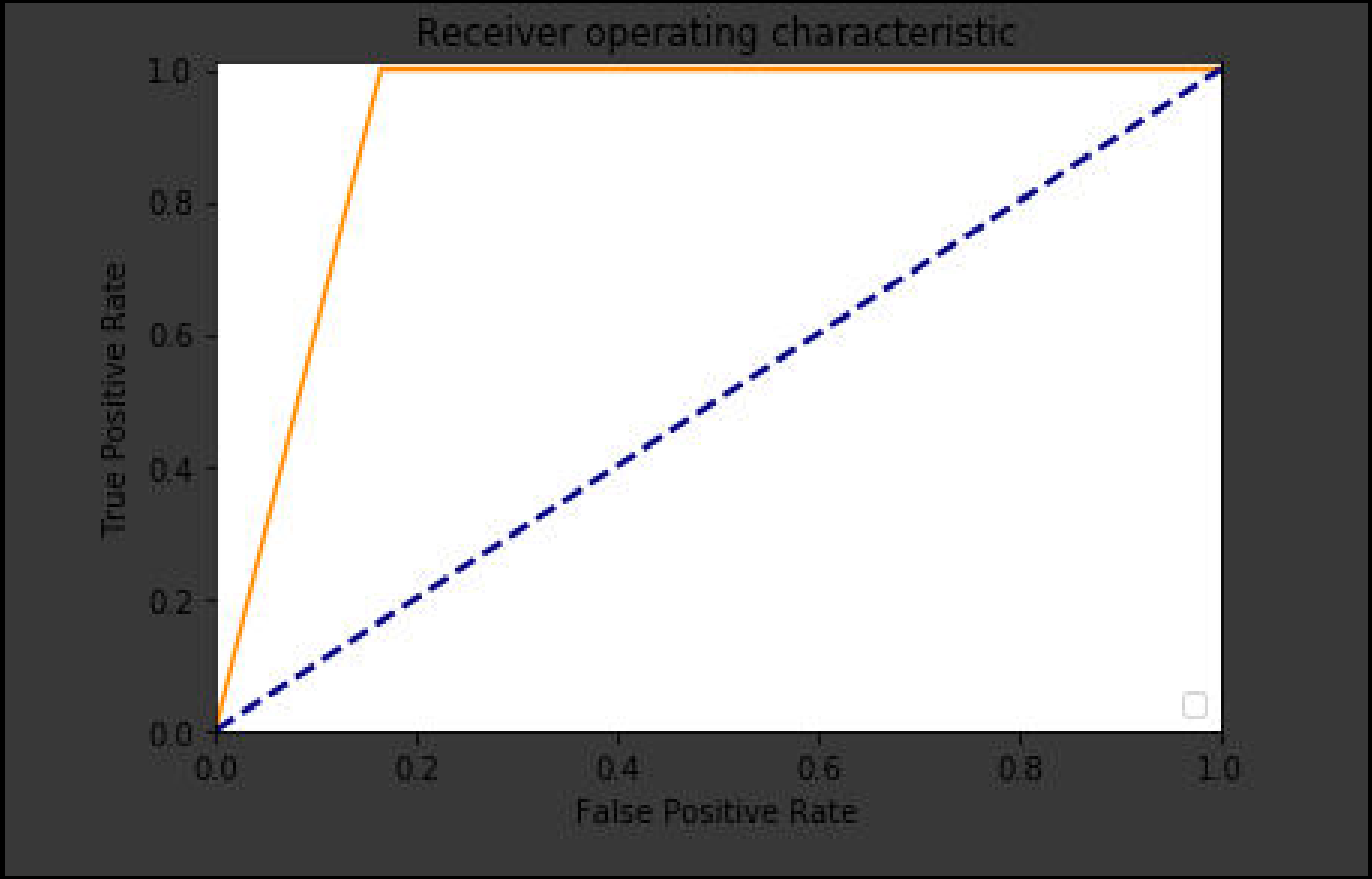




When two distributions overlap, we introduce type 1 and type 2 error. Depending upon the threshold, we can minimize or maximize them.

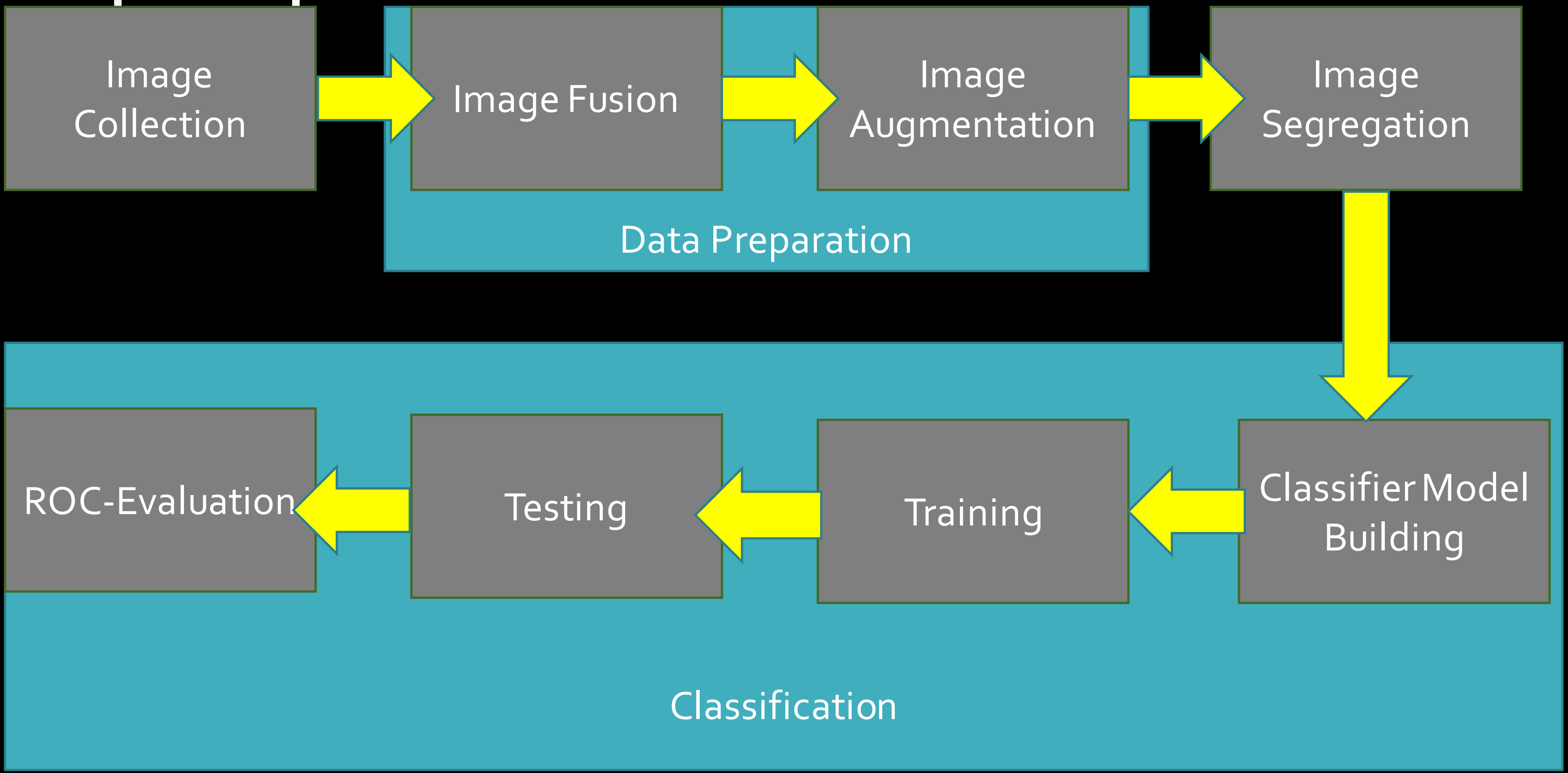
When AUC is 0.7, it means there is 70% chance that model will be able to distinguish between positive class and negative class.

Our ROC-AUC



Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

Finally, to put it all into perspective



Future Works

- Learn to work with DCIM scans(3D scans) instead of 2D JPEG Images.

- We want to Build our own supervised Learning Model which specializes in Brain Tumour Detection.

- Once we are able to Collect more data we can segregate the Output Classes into Other Labels to detect the stage of the Tumour.

- We aim for an efficient and a faster Algorithm specialised to serve our purpose.

References

- Brain Tumor Extraction from MRI Images Using MATLAB
  - By Rajesh C. Patel, Dr. A. S. Bhalchandra
  - International Journal of Electronics, Communication & Soft Computing Science and Engineering ISSN: 2277-9477, Volume 2, Issue 1
  -
- **Automatic Brain Tumour Detection and Segmentation Using U-Net Based Fully Convolutional Networks**
  - By H. Dong, G. Yang and F. Liu
  - Annual Conference on Medical Image Understanding and Analysis
- A survey of MRI-based medical image analysis for brain tumor studies.
  - Bauer, S., Wiest, R., Nolte, L.-P., Reyes, M.:
  - Phys. Med. Biol. **58**, R97–R129 (2013)
- **Classification using deep learning neural networks for brain tumors**
  - HebaMohsen et. al.
  - Future Computing and Informatics Journal Elsevier
- **Brain Tumor Type Classification via Capsule Networks**
  - Parnian Afshar ; Arash Mohammadi ; Konstantinos N. Plataniotis
  - 2018 25th IEEE International Conference on Image Processing (ICIP)
- https://med.Harvard.edu (Data Collected from here)

THANK YOU