A Generalized Method for Cancer Detection: A CNN Based Approach

Authors:

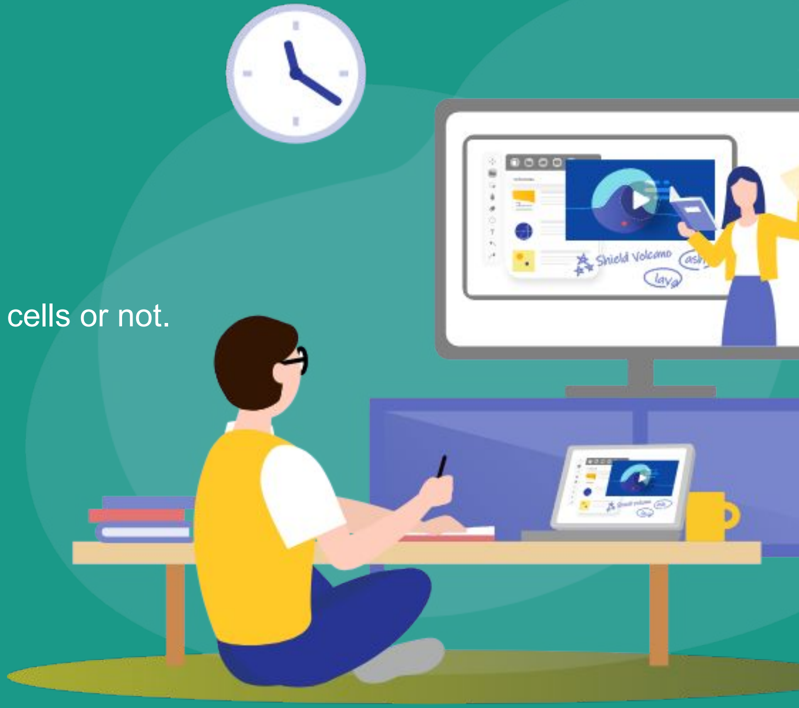
Akash Kalluvilayil Venugopalan

Sreedev D

Vishnu Mohan M S

Content

1. Overview of Project working



- Here we upload an Image to the site that either contain cancerous cells or not.
- Then we choose the model from dropdown box.
- The models we designed are:

  1.Skin Cancer-M1

  2.Brain Cancer-M2

  3.Combined Model-(M1 & M2)

  4.Generalized Model

2. Literature Survey

- Paper Title:

  Big data analysis for brain tumor detection: Deep convolutional neural networks

- Authors:

  Javeria Amin, Muhammad Sharifa, Mussarat Yasmina and Steven Lawrence Fernandes

- Publisher and Date of publication:

  Elsevier, Future Generation Computer Systems, Volume 87, October 2018,Pages 290-297

2. Literature Survey

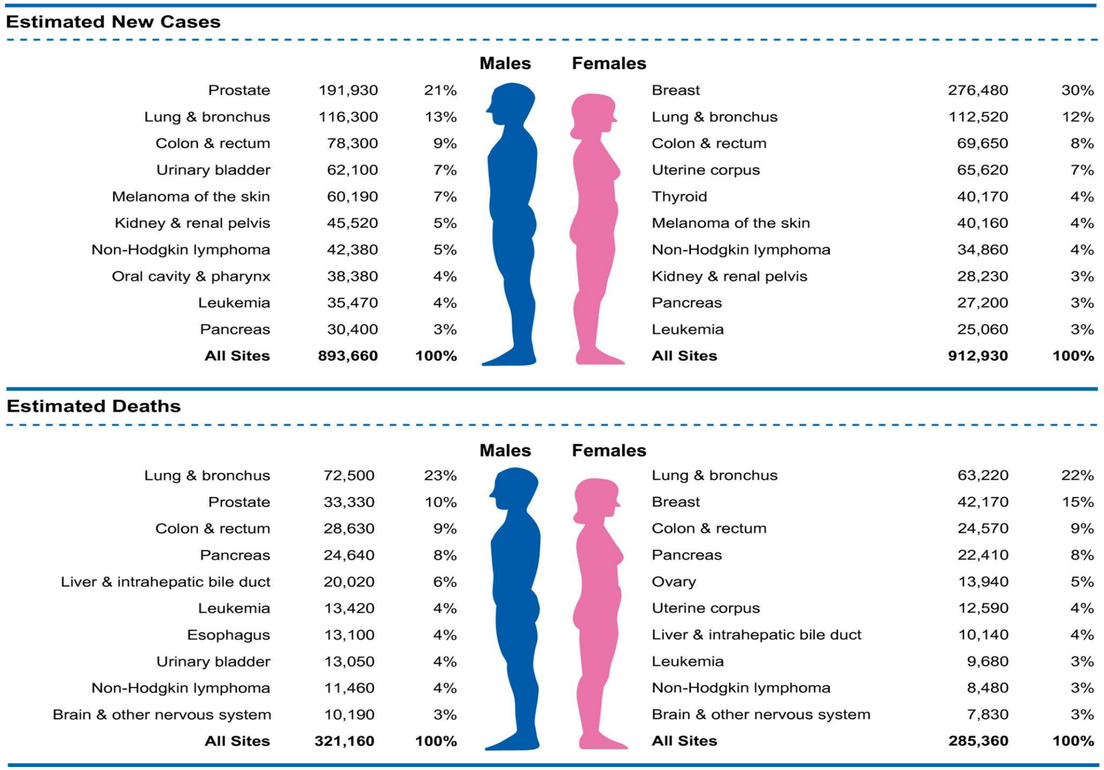
Title: Big data analysis for brain tumor detection: Deep convolutional neural networks

- The paper explains about detection and segmentation of brain tumor from MRI images.
- The dataset used:
    1. BRATS (image dataset and synthetic dataset) 2012 , 2013 , 2014, 2015 and,
    2. ISLES (Ischemic stroke lesion segmentation) 2015 and 2017
- They have used 7 layers, that is 3 convolution layer, 3 ReLU layer and output layer which uses a softmax function.
- The overall experiment is performed on Core i7, 3.4 GHz CPU with 32 GB RAM and Nvidia K40 GPU running on the MATLAB toolbox

3. Introduction to the Problem

- Cancer is one of the most common diseases we can see around the world.
- Which is second case of leading cause of death around the world. (Whereas first is heart disease)
- In 2020, 1,806,590 new cancer cases and 606,520 cancer deaths are projected to occur in the United States alone.
- Prostate, lung and bronchus, and colorectal cancers, account for 43% of all cases in men, with prostate cancer alone accounting for more than 1 in 5 new diagnoses.
- For women, the 3 most common cancers are breast, lung, and colorectal, accounting for 50% of all new diagnoses; breast cancer alone accounts for 30% of female cancers.

3. Introduction to the Problem



Estimated New Cases

Males			Females		
Prostate	191,930	21%	Breast	276,480	30%
Lung & bronchus	116,300	13%	Lung & bronchus	112,520	12%
Colon & rectum	78,300	9%	Colon & rectum	69,650	8%
Urinary bladder	62,100	7%	Uterine corpus	65,620	7%
Melanoma of the skin	60,190	7%	Thyroid	40,170	4%
Kidney & renal pelvis	45,520	5%	Melanoma of the skin	40,160	4%
Non-Hodgkin lymphoma	42,380	5%	Non-Hodgkin lymphoma	34,860	4%
Oral cavity & pharynx	38,380	4%	Kidney & renal pelvis	28,230	3%
Leukemia	35,470	4%	Pancreas	27,200	3%
Pancreas	30,400	3%	Leukemia	25,060	3%
All Sites	**893,660**	**100%**	**All Sites**	**912,930**	**100%**

Estimated Deaths

Males			Females		
Lung & bronchus	72,500	23%	Lung & bronchus	63,220	22%
Prostate	33,330	10%	Breast	42,170	15%
Colon & rectum	28,630	9%	Colon & rectum	24,570	9%
Pancreas	24,640	8%	Pancreas	22,410	8%
Liver & intrahepatic bile duct	20,020	6%	Ovary	13,940	5%
Leukemia	13,420	4%	Uterine corpus	12,590	4%
Esophagus	13,100	4%	Liver & intrahepatic bile duct	10,140	4%
Urinary bladder	13,050	4%	Leukemia	9,680	3%
Non-Hodgkin lymphoma	11,460	4%	Non-Hodgkin lymphoma	8,480	3%
Brain & other nervous system	10,190	3%	Brain & other nervous system	7,830	3%
All Sites	**321,160**	**100%**	**All Sites**	**285,360**	**100%**

Ref: ACS Journals, Cancer statistics, 2020, Doi: https://doi.org/10.3322/caac.21590

4. Our Approach

- In our project, we have developed a web framework where doctors or patients can upload image of a particular area of body (area can be image of skin or CT scan of lung or brain)and check whether this part contain any cancerous cell.
- The website take an image as an input and user should choose a relevant deep learning model for cancer detection.
- Then the server take this image and pass it to the model as per the users choice.
- Finally model run in the server side and after model arrived at a result, this result will be then send back to website. So user can see the result at their screen.

4. Our Approach

The model mainly classified into 4 types in the site:

1. Skin Cancer-M1: This model only capable of detecting skin cancer from uploaded image. So user should only give skin image to this model and model make prediction with an accuracy of 82.5%.
2. Brain Cancer-M2: Similarly, this model can be used for detect brain cancer only. And the prediction accuracy is 93%.
3. Combined Model-(M1 & M2): Model that can detect cancer from skin, brain or lung. It cannot take image of any other portion of body. Accuracy of prediction will be 86%.
4. Generalized Model: This is the major model we have developed. The model is capable of detect cancer from any part of the body. So user can upload scanned image of liver, lung, skin, brain, bronchus etc. to model to detect cancer. Model has an accuracy of 82.2% in prediction.

5. Implementation

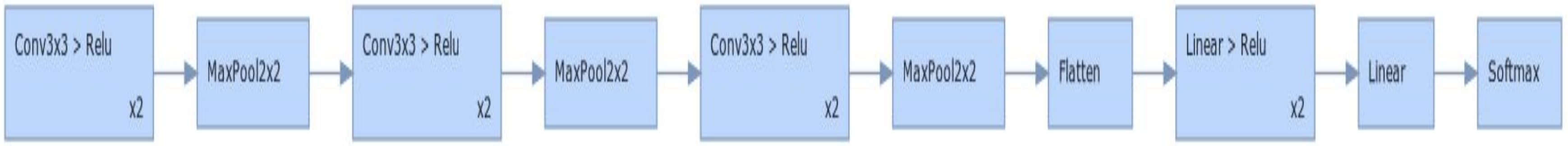
Technology used for Implementation of models

- Pytorch
- Tensorflow
- Dask - Dask is a flexible parallel computing library for analytics
- Scikeras- Creates a wrapper around keras model.

Models

Model	Usage
Model 1- Combined	Classify between 4 class: Lung, Brain, Skin, No cancer
Model 2 - Brain only	Malignant, Benign(Brain Cancer)
Model 3 - Skin Cancer	Malignant, Benign(Skin cancer)
Model 4 - Generalized Model	Malignant, Benign(All class of cancer)

Models - General Architecture



Model 1,4 : Same
Model 2,3 : Excluded some CNN layers

Data Preprocessing

```
TRANSFORM_IMG = transforms.Compose([
    transforms.Grayscale(num_output_channels=1),
    transforms.Resize((320,320)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0],std=[1] )])
```

Model 1 Label Encoding:
{'Brain': 0, 'Lung': 1, 'Skin': 2, 'no': 3}

Model 2,3,4 Label Encoding:
{Benign: 0, Malignant: 1}

Model 2,3 : Resize (128,128)
(Grayscale transformation not used in skin only Model)
Model 4 : Resize (200,200)

Optimizer and Loss Function

```
loss_fn=nn.CrossEntropyLoss()

optimizer=optim.SGD(model.parameters(),lr=0.001,momentum=0.9)
```

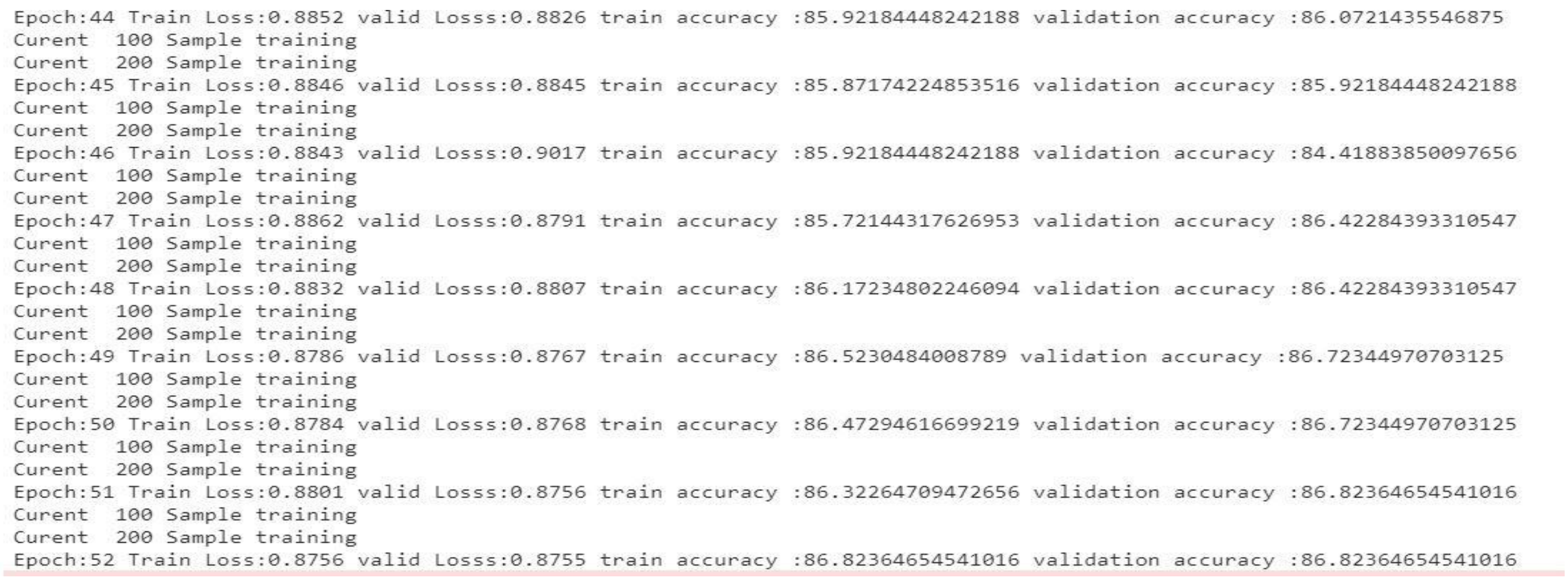
$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i) \quad \theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Metrics used

● Sensitivity	● TPR = TP / (TP + FN)
● Precision	● PPV = TP / (TP + FP)
● Accuracy	● ACC = (TP + TN) / (P + N)
● F1 Score	● F1 = 2TP / (2TP + FP + FN)
● ROC Curve	● Sensitivity Vs 1-Specificity (TPR vs FPR)
● Specificity	● SPC = TN / (FP + TN)

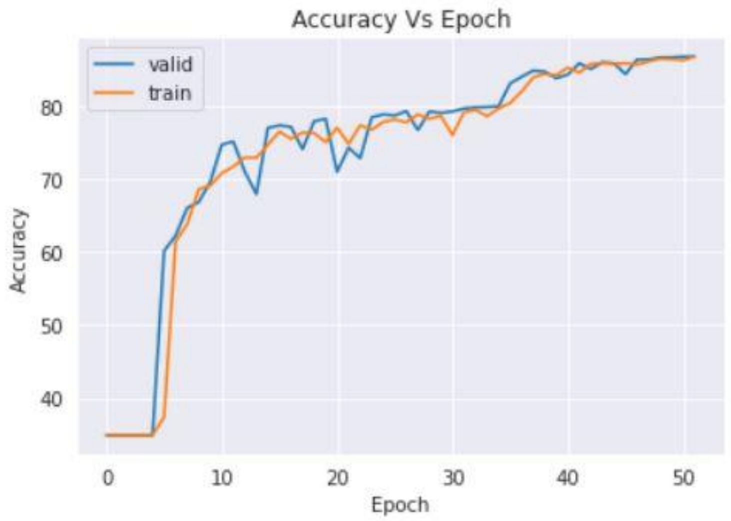
6. Results

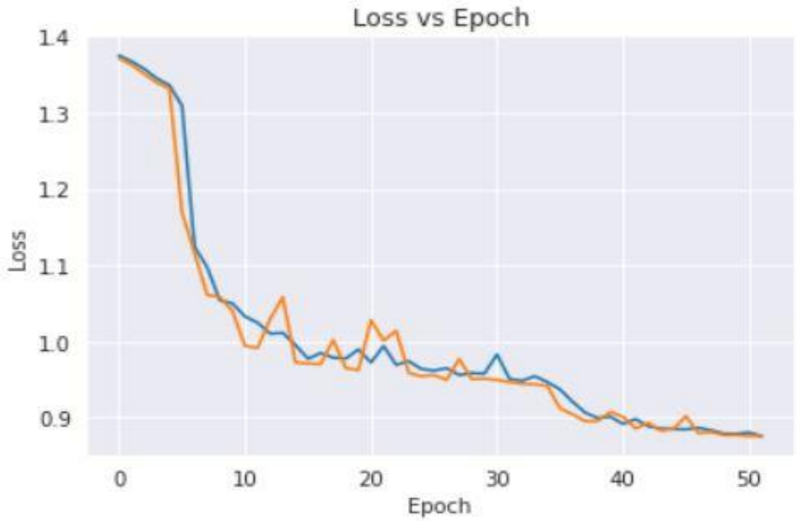
Model 1(Combined) : Accuracy,Loss Vs Epoch



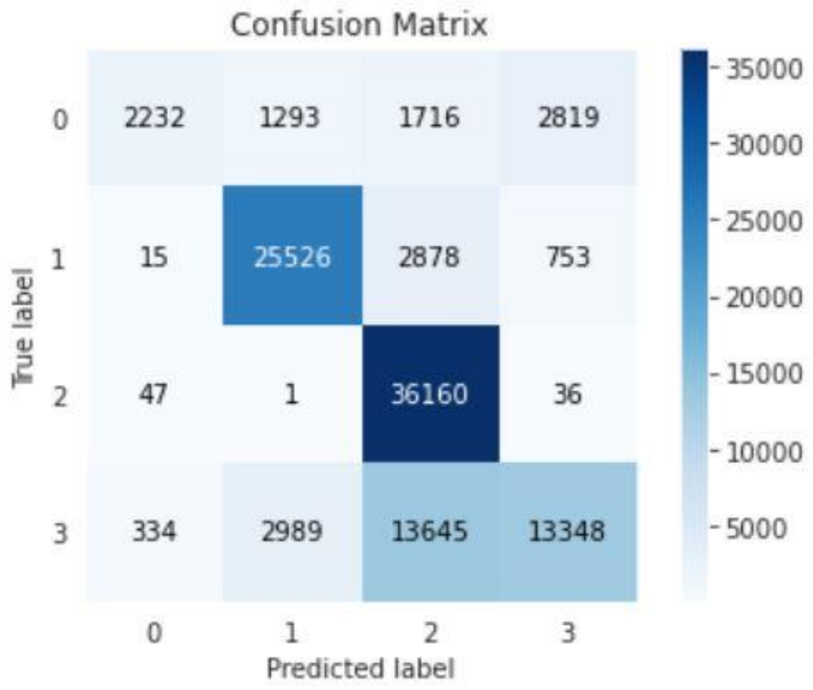
```
Epoch:44 Train Loss:0.8852 valid Losss:0.8826 train accuracy :85.92184448242188 validation accuracy :86.0721435546875
Curent  100 Sample training
Curent  200 Sample training
Epoch:45 Train Loss:0.8846 valid Losss:0.8845 train accuracy :85.87174224853516 validation accuracy :85.92184448242188
Curent  100 Sample training
Curent  200 Sample training
Epoch:46 Train Loss:0.8843 valid Losss:0.9017 train accuracy :85.92184448242188 validation accuracy :84.41883850097656
Curent  100 Sample training
Curent  200 Sample training
Epoch:47 Train Loss:0.8862 valid Losss:0.8791 train accuracy :85.72144317626953 validation accuracy :86.42284393310547
Curent  100 Sample training
Curent  200 Sample training
Epoch:48 Train Loss:0.8832 valid Losss:0.8807 train accuracy :86.17234802246094 validation accuracy :86.42284393310547
Curent  100 Sample training
Curent  200 Sample training
Epoch:49 Train Loss:0.8786 valid Losss:0.8767 train accuracy :86.5230484008789 validation accuracy :86.72344970703125
Curent  100 Sample training
Curent  200 Sample training
Epoch:50 Train Loss:0.8784 valid Losss:0.8768 train accuracy :86.47294616699219 validation accuracy :86.72344970703125
Curent  100 Sample training
Curent  200 Sample training
Epoch:51 Train Loss:0.8801 valid Losss:0.8756 train accuracy :86.32264709472656 validation accuracy :86.82364654541016
Curent  100 Sample training
Curent  200 Sample training
Epoch:52 Train Loss:0.8756 valid Losss:0.8755 train accuracy :86.82364654541016 validation accuracy :86.82364654541016
```

Model 1(Combined) : Accuracy,Loss Vs Epoch





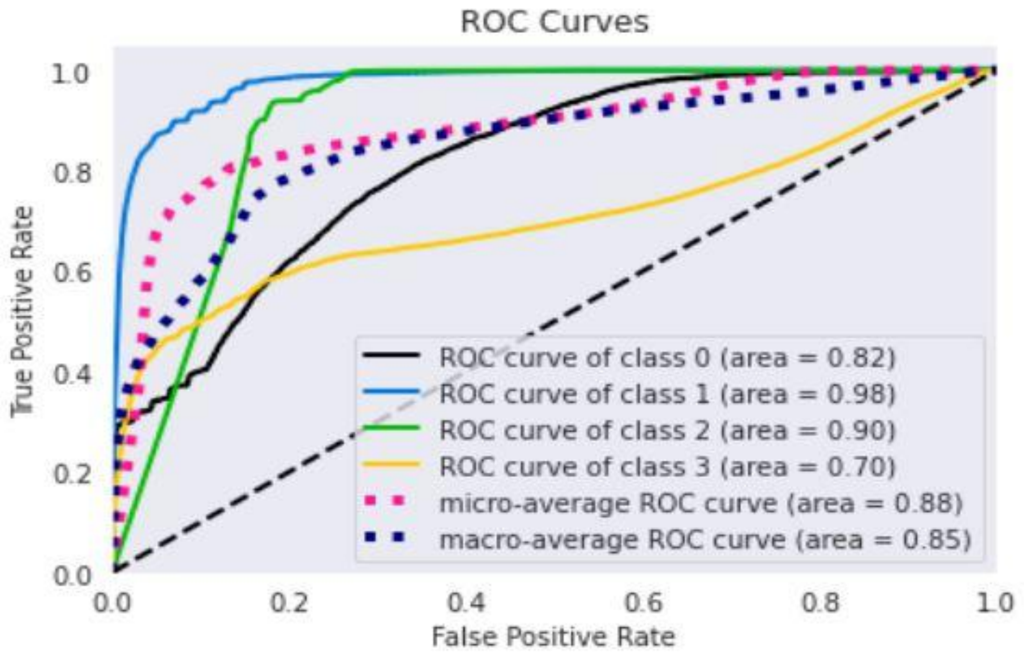
Model 1 (Combined) : Confusion Matrix



True label \ Predicted label	0	1	2	3
0	2232	1293	1716	2819
1	15	25526	2878	753
2	47	1	36160	36
3	334	2989	13645	13348

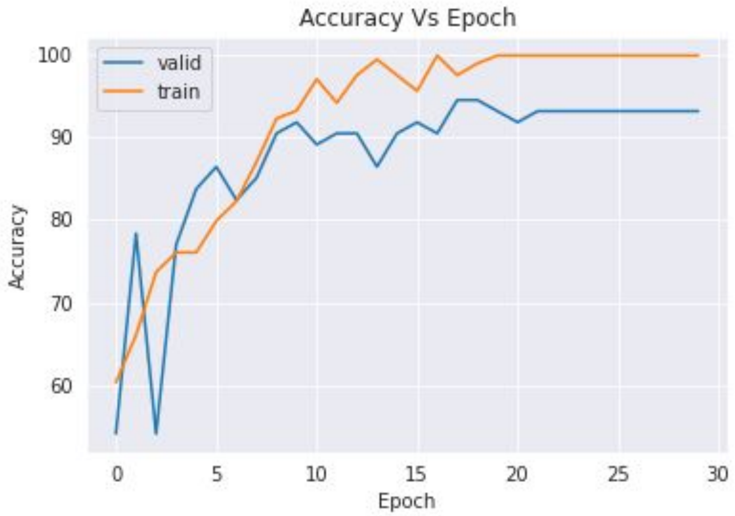
{'Brain': 0, 'Lung': 1, 'Skin': 2, 'no': 3}

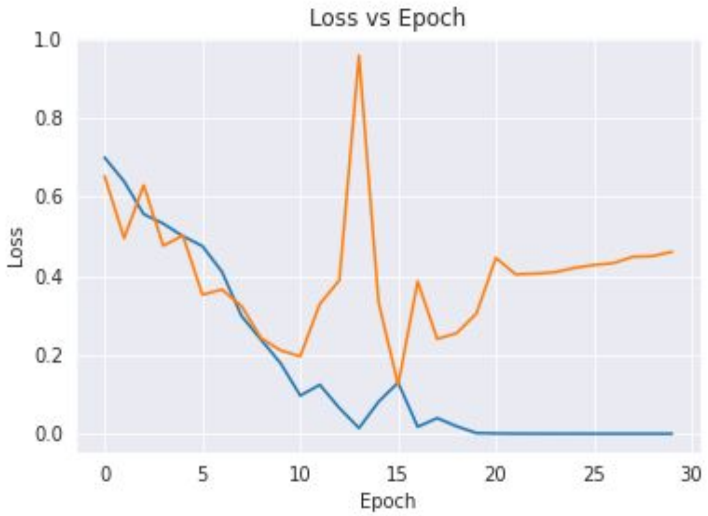
Model 1 (Combined) : ROC Curve



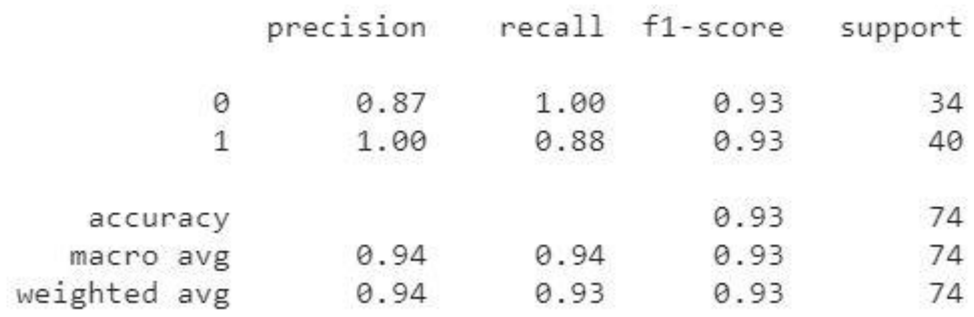
{'Brain': 0, 'Lung': 1, 'Skin': 2, 'no': 3}

Model 2(Brain Only): Accuracy,Loss Vs Epoch

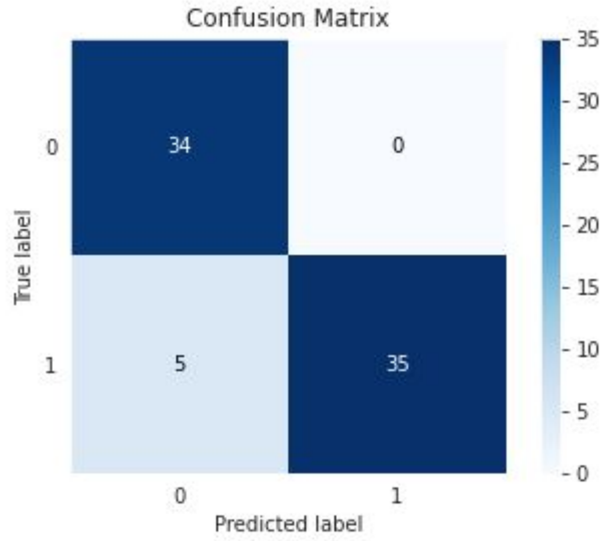




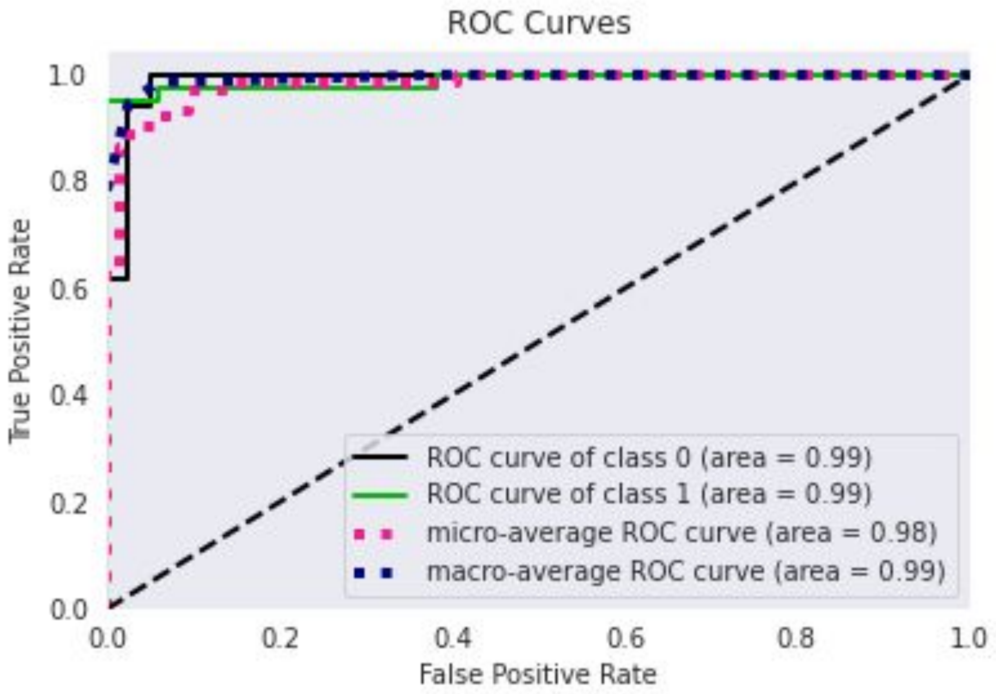
Model 2(Brain Only) :Confusion Matrix



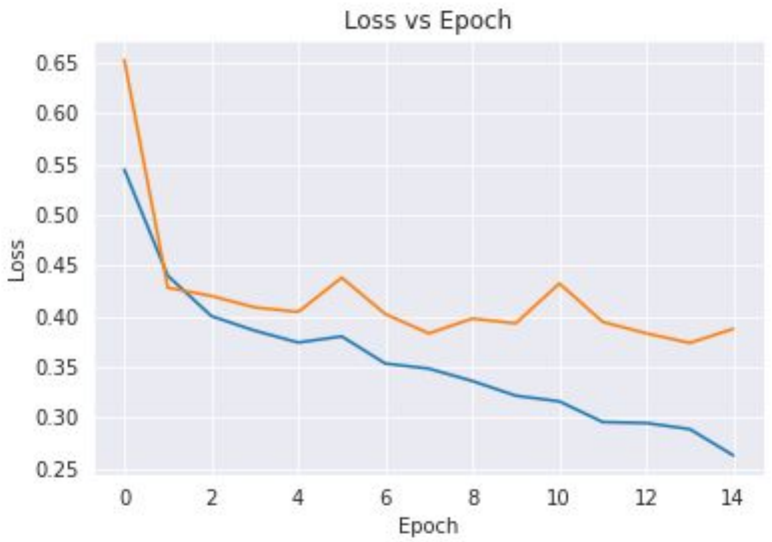
```
              precision    recall  f1-score   support

           0       0.87      1.00      0.93        34
           1       1.00      0.88      0.93        40

    accuracy                           0.93        74
   macro avg       0.94      0.94      0.93        74
weighted avg       0.94      0.93      0.93        74
```

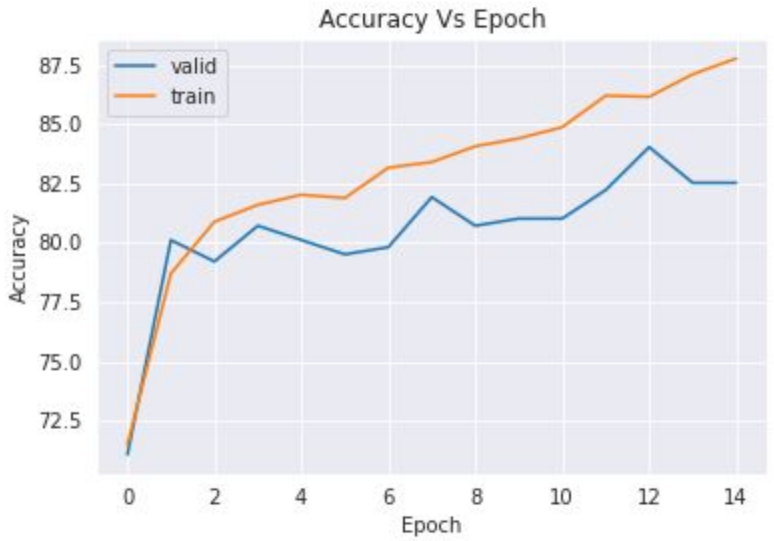


Model 2 (Brain Only): ROC Curve

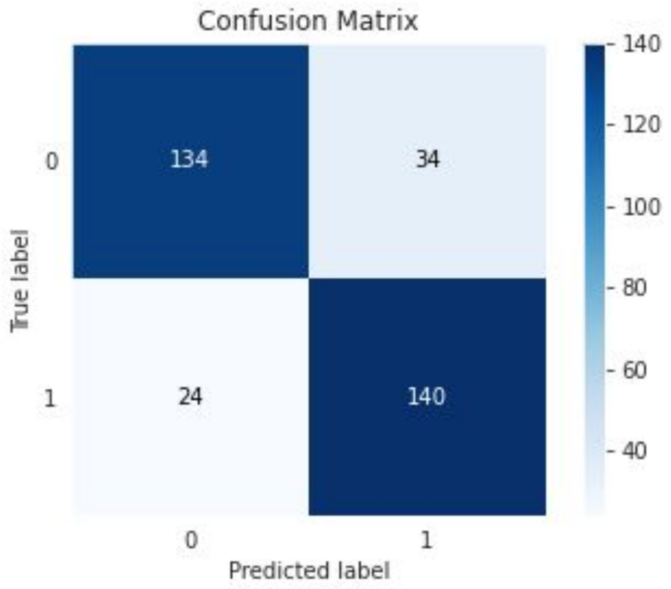


Model 3 (Skin Only): Loss,Accuracy Vs Epoch

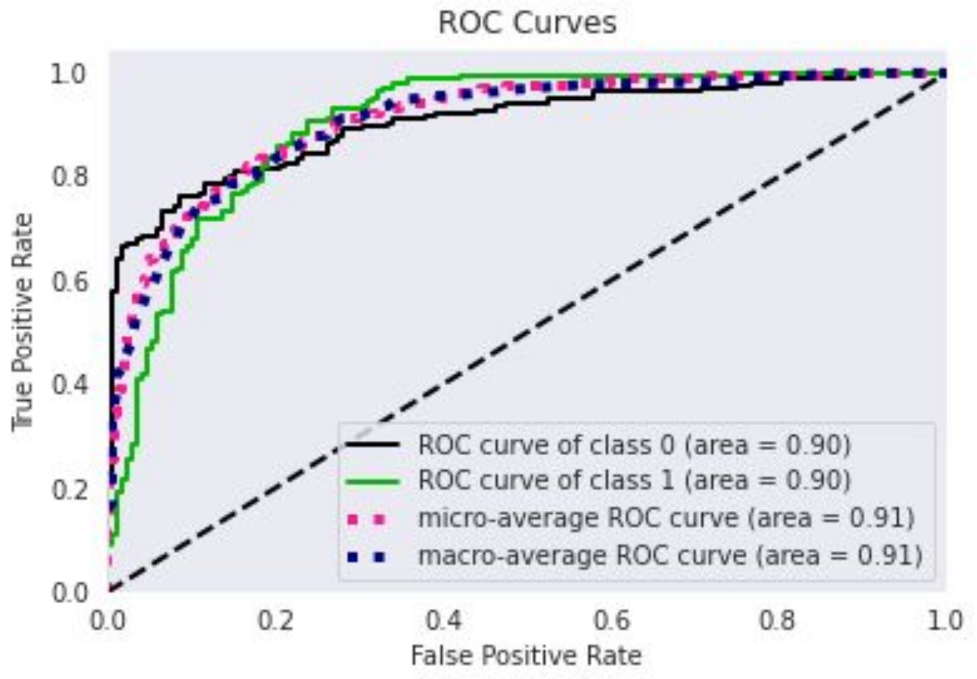




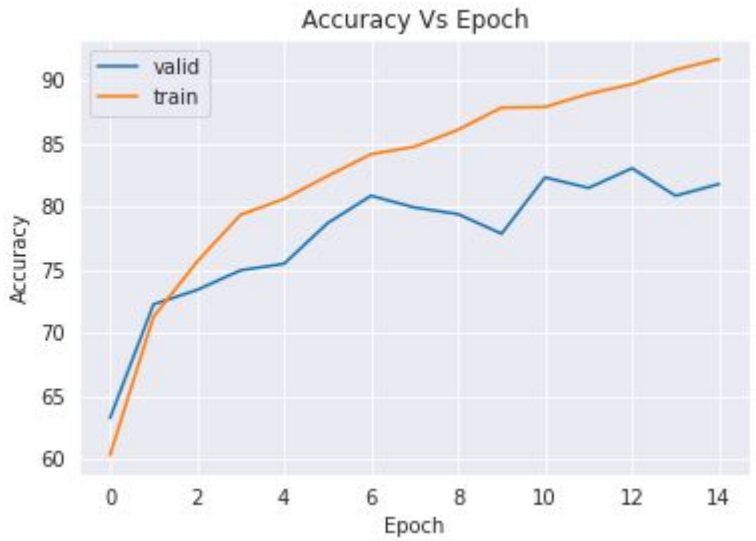
Model 3 (Skin Only): CM

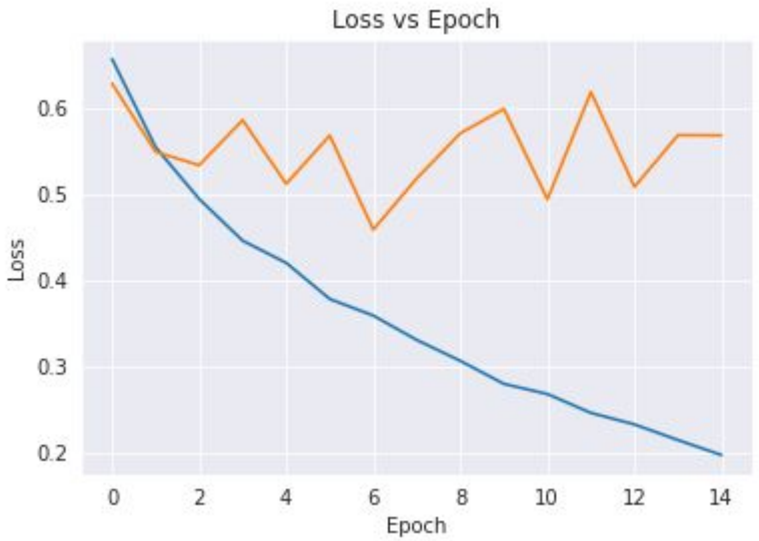


Model 3 (Skin Only): ROC Curve

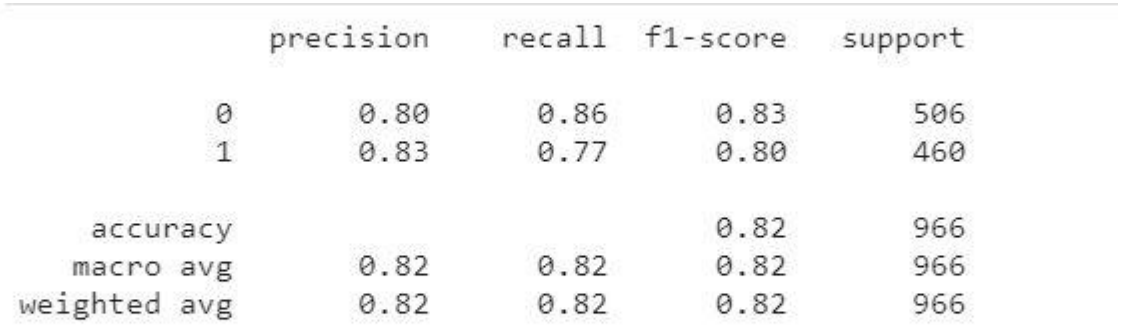


Model 4 (Combined-1): Loss,Accuray Vs Epoch

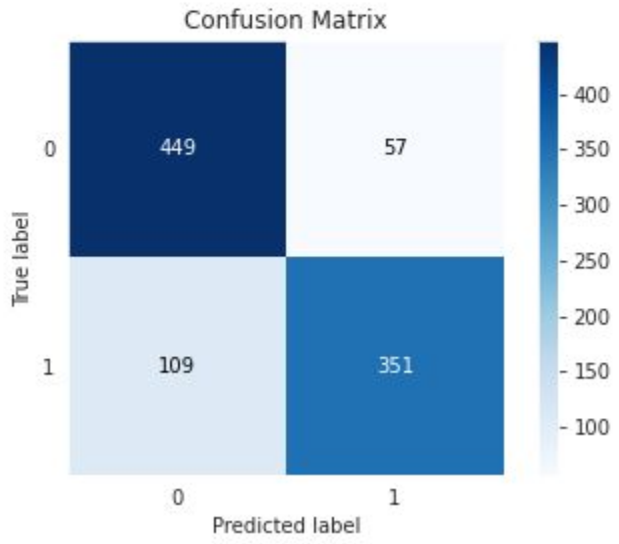




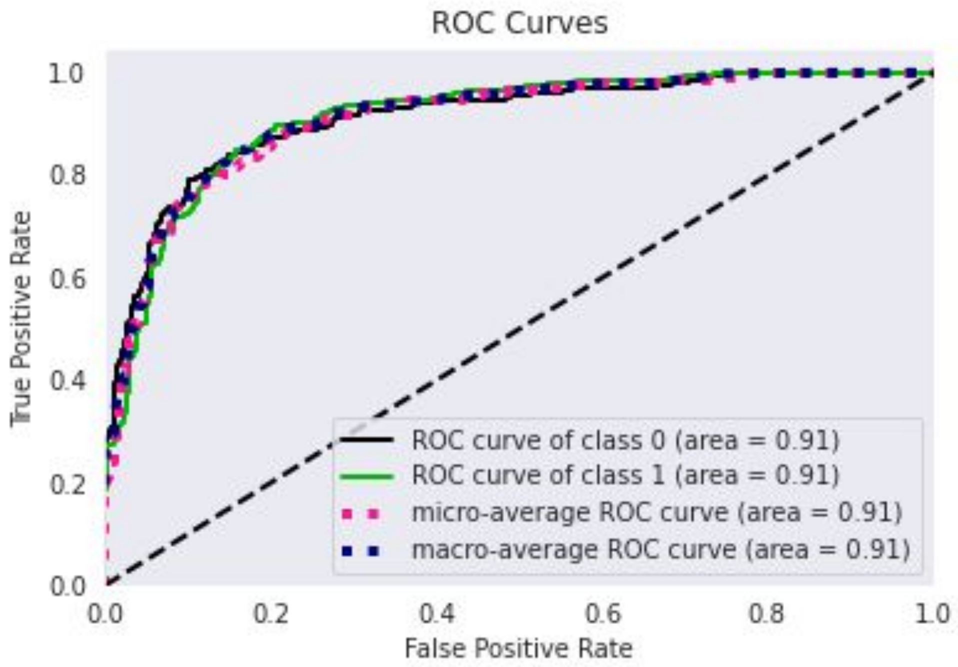
Model 4 (Combined-1): CR and CM



```
              precision    recall  f1-score   support

           0       0.80      0.86      0.83       506
           1       0.83      0.77      0.80       460

    accuracy                           0.82       966
   macro avg       0.82      0.82      0.82       966
weighted avg       0.82      0.82      0.82       966
```



Model 4(Combined-1): ROC Curve



Accuracy score for each model

Models	**Accuracy**
Model 1	86%
Model 2	93%
Model 3	82.5%
Model 4	82.8%

7. Flask

- Flask is a micro-framework for Python
- Flask won’t make many decisions for you, such as what database to use.

```python
# Import Flask library
from flask import Flask
# Initialize the app from Flask
app = Flask(__name__)
# Define a route to hello world funtion
@app.route('/')
def hello_world():
    return 'Hello World!'
# Run the app on http://localhost:8085
app.run(debug=True,port=8085)
```

- It is easy to code, easy to configure, testable and restful.
- A Flask application is started by calling the run() method.
- As a result, if a user visits for example http://localhost:5000/ file name.
- There are two primary coding environments for the whole web app ecosystem.
  1. Client-side scripting – The code executed on the user's browser visible to anyone who has access to the system, generating the first results.
  2. Server-side scripting – This type of code is run on the backend on a web server. To enable developers to design, build, maintain, host web apps over the internet, a web framework is required.

```python
rom flask import Flask, render_template,
redirect, url_for, request, make_response,
session
from models import *
types = None
app = Flask(__name__)
# two decorators, same function
@app.route('/')
@app.route('/home')
def index():
    return render_template('index.html')
@app.route('/test')
def test():
    return render_template('test.html')
@app.route('/samples1')
def sample_select_1():
    session['path'] = None
    path="static\\images\\L1.jpeg"
    session["path"] = path
    return render_template('test.html')
```

- This is the sample flask program in our project named routes.py
- The route() decorator in Flask is used to bind URL to a function.
- As a result, if a user visits http://localhost:5000/.
- First redirect the browser in to home page as index.html.

Amazon EC2

Amazon Machine Image (AMI)

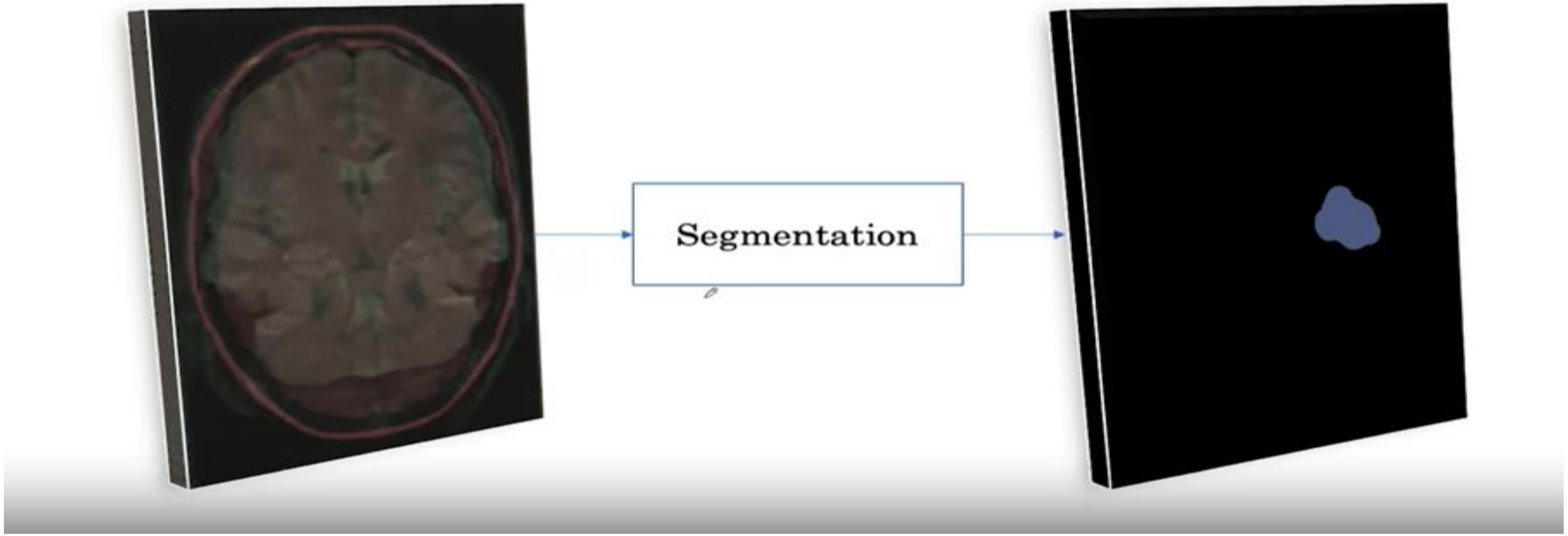
Amazon EC2

- Amazon Machine Images (**AMIs**) are the basic building blocks of Amazon EC2.
- An AMI is a template that contains a software configuration (operating system, application server and applications) that can run on Amazon’s computing environment
- AMIs can be used to launch an ***instance***, which is a copy of the AMI running as a virtual server in the cloud.

EC2 Instance

- Serving the created flask using EC2 instance.
- The flask/source is copied to the linux server created in the EC2
- Created a python virtual environment there.
- Made inbound rule changes to the EC2 instance for public access.
- Launched flask production server.

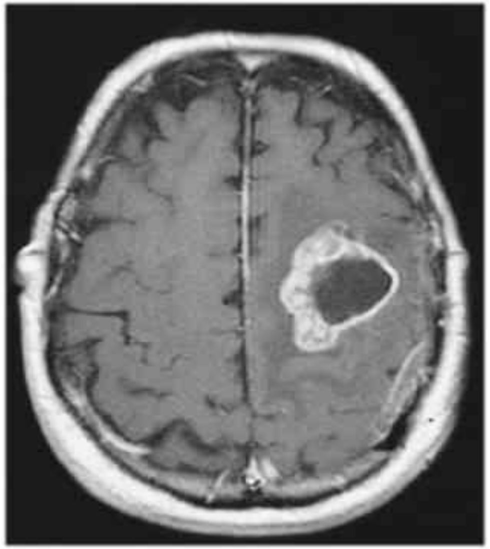
8. Segmentation and Detection



Source : Deeplearning.ai

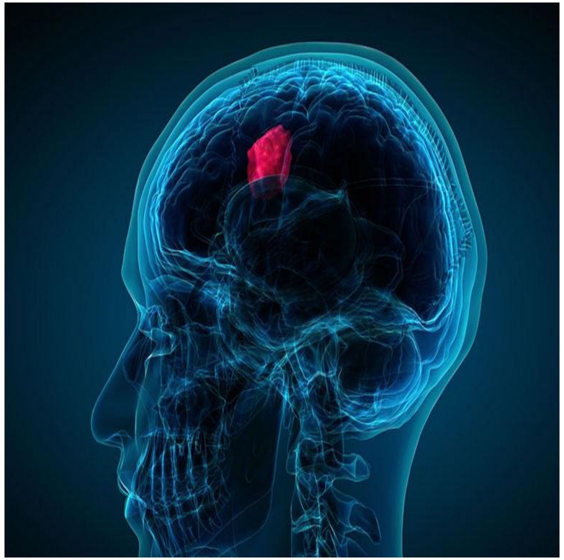
Problems: using MRI Scan locally for Deep Learning

- Each sample has a dimension of 250*250*155*4
- Loading this into memory for training is difficult
- ImageLoader class in pytorch and tensorflow doesn't support this type of Data.
- For training purpose we only used a slice of MRI scan, which is a grayscale image



9. Future Scope

- We are creating the ml model for only testing skin, lung and brain cancer.
- Different classifiers can be used to increase the accuracy combining more efficient segmentation and feature extraction techniques.
- In future create a model for testing in MRI image.
- Segmentation
    - It can Detect the location of the cancer cells.
    - The output will be image



Conclusion

- Cancer is the one of common problem whose incidence is increasing in the world.
- With the help of the website that we have developed can help in the diagnosis of the cancer.
- This project can detect cancer from any part of the body with the help of generalized model. The user should only upload the image to the site and test it.
- We can improve on this project and make much more accurate models.

Reference

[1] Javeria Amin, Muhammad Sharifa, Mussarat Yasmina, Steven Lawrence Fernandes, “Big data analysis for brain tumor detection: Deep convolutional neural networks”, *Elsevier, Future Generation Computer Systems*, Volume 87, October 2018,Pages 290-297

[2] Rebecca L. Siegel, Kimberly D. Miller, Ahmedin Jemal, ACS JOURNALS, Cancer statistics, 2020, Volume 70, issue 1, February 2020, Pages 7-30

Thank You