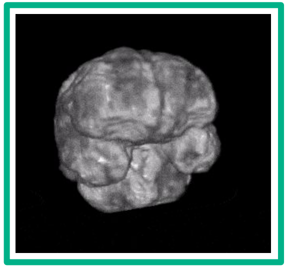


Brain Tumor Segmentation

Hewlett Packard Enterprise

- Team 6 Capstone Project

Introduction.

- Brain tumor – One of the most difficult problems in medicine.
- If not detected, the tumor may grow in size and hamper brain functioning.
- May put pressure, cause damage and can lead to organ failure.
- Prompt diagnosis and treatment planning for tumors is of utmost importance.

Business Problem.

Radiologist use MRI Scans for Detecting Brain Tumors.

Statistically overworked and stressed.

65%

Could Lead to Misdiagnosis of Cancer Cells.

Such Mistakes could lead to Fatal Consequences.

Solution Statement.

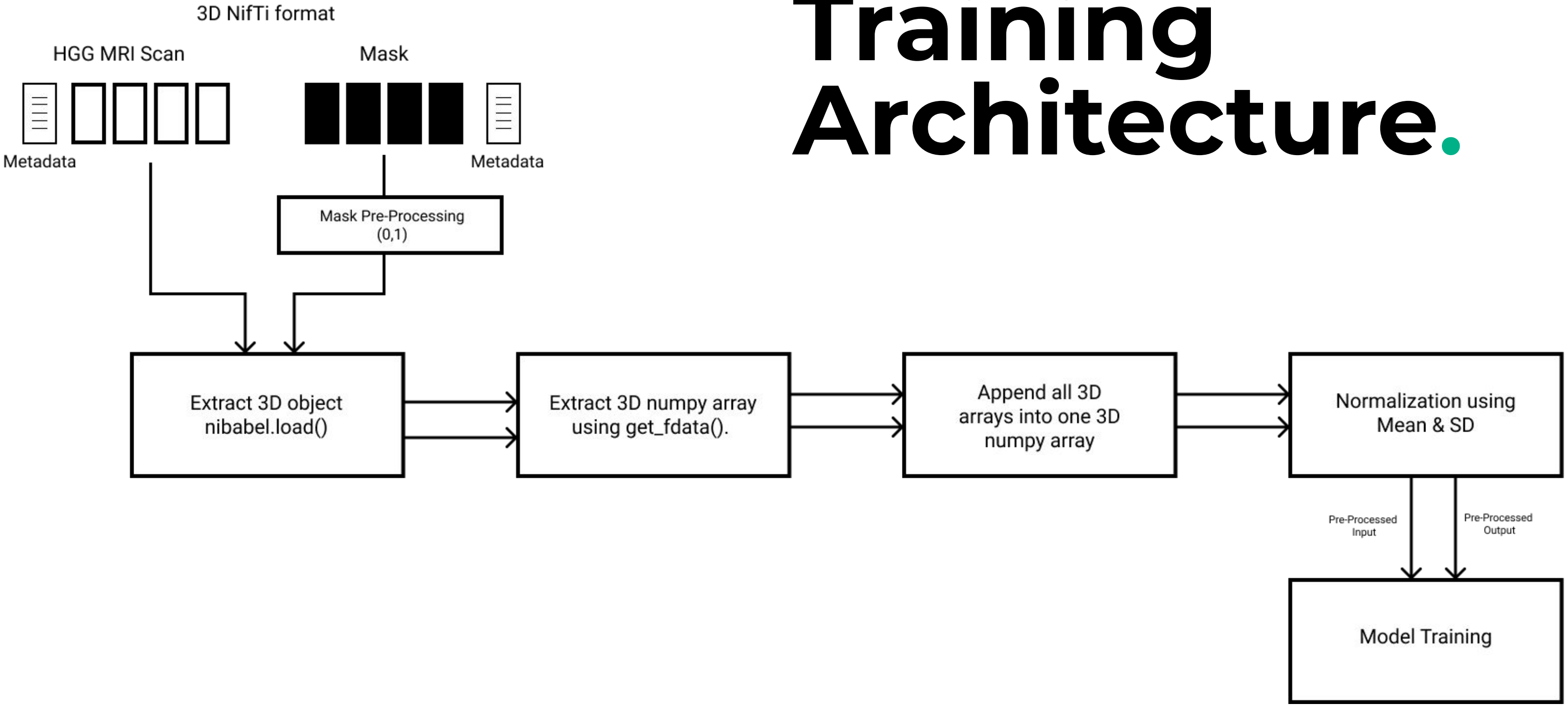
- Performs brain tumor segmentation using brain MRI scans.
- Used data from University of Pennsylvania BraTS Challenge 2019.
- Highlights tumor position using UNet deep learning model.
- Aims at contributing to the medical sector, especially radiologists.
- Provides useful information for diagnosis and treatment planning through web app.

Deep
Learning:

Data Collection and Preprocessing.

- Get NifTi 3D object from NifTi files using nibabel.load().
- Extract 3D numpy array from NifTi object using get_fdata().
- Convert all mask values greater than 0 to 1.
- Append all 2D scan arrays into one 3D numpy array.
- Perform normalisation using mean and std dev.

Training Architecture.



Models Used.

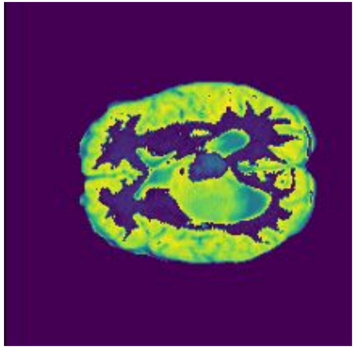
Initial run done to pick better model

Model Name	Train Dice Coefficient	Validation Dice Coefficient
UNet	19.30	5.85
ResUNet	56.77	3.30

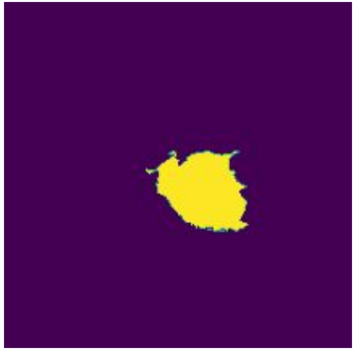
- Initial run was done on entire dataset for 2 epochs.
- From the above table, it is clear that U-Net is the better model.

Results.

Model Name	Dice Coefficient	IoU	Localization Error
UNet	72.777	0.670	12.032



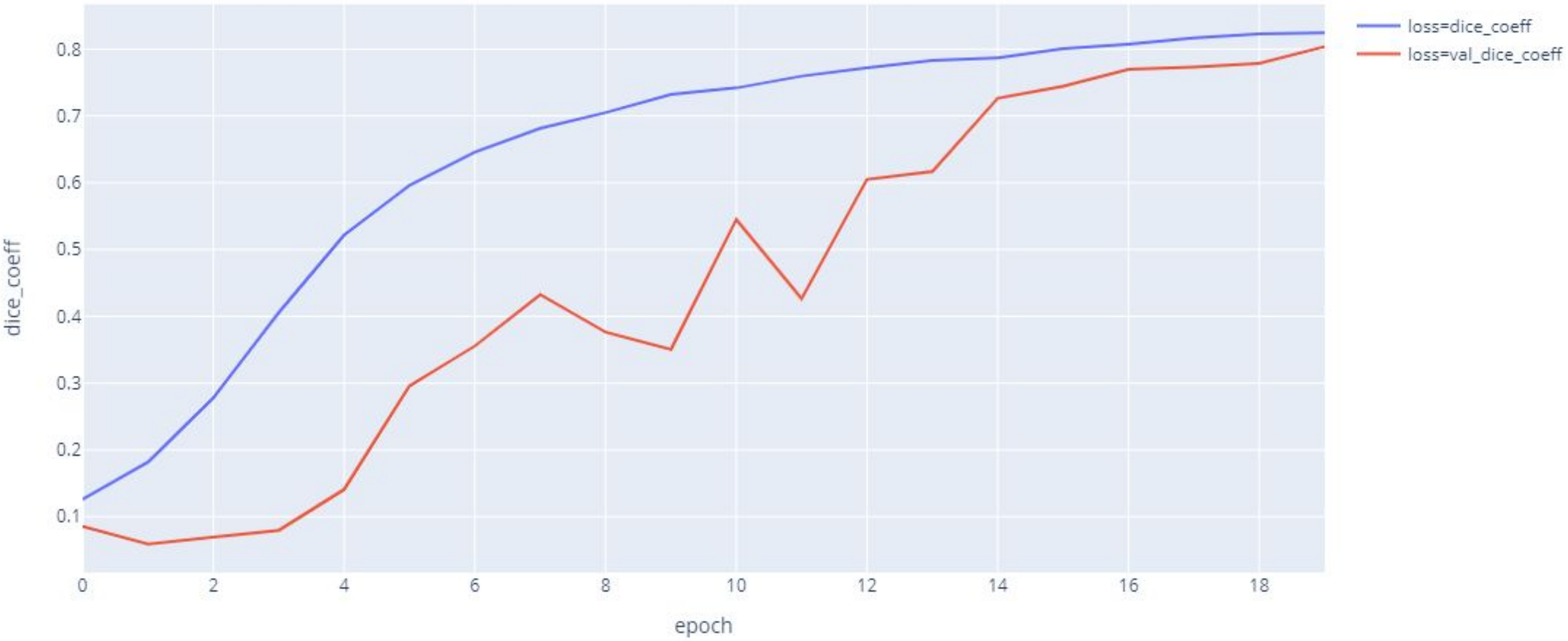
Source Image



Ground Truth

Prediction

Charts.

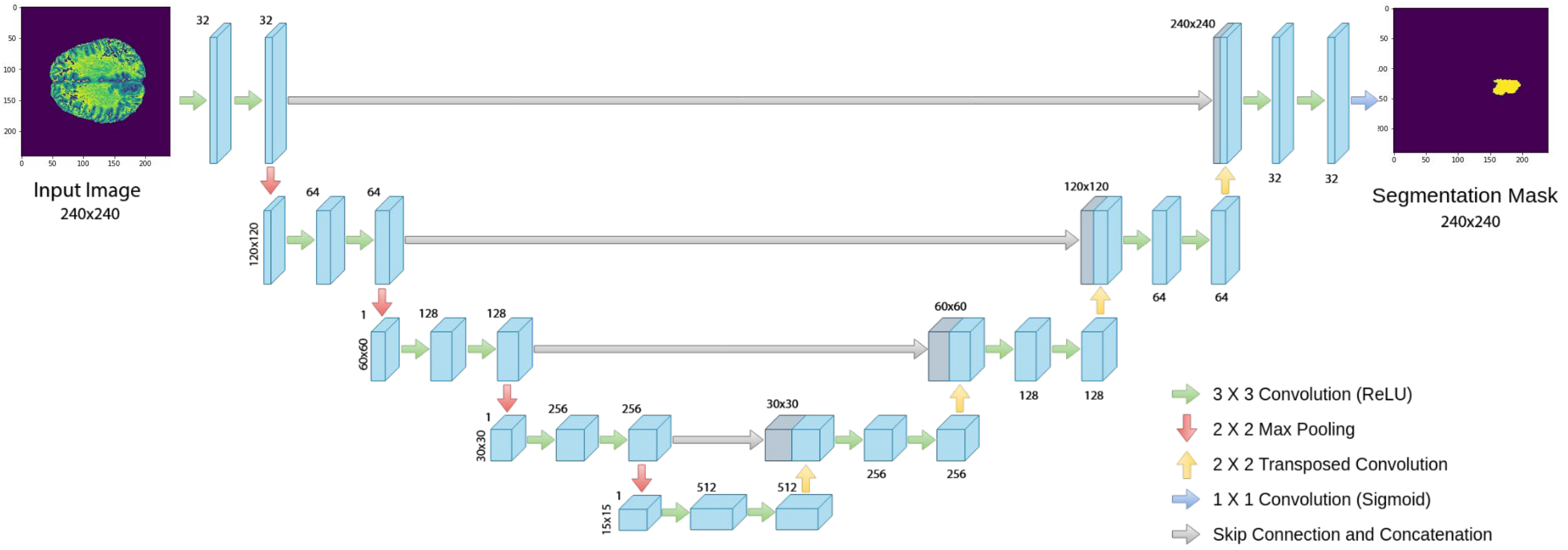


loss=dice_coeff
loss=val_dice_coeff
dice_coeff
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8
10
12
14
16
18
epoch

Best Model - U-Net.

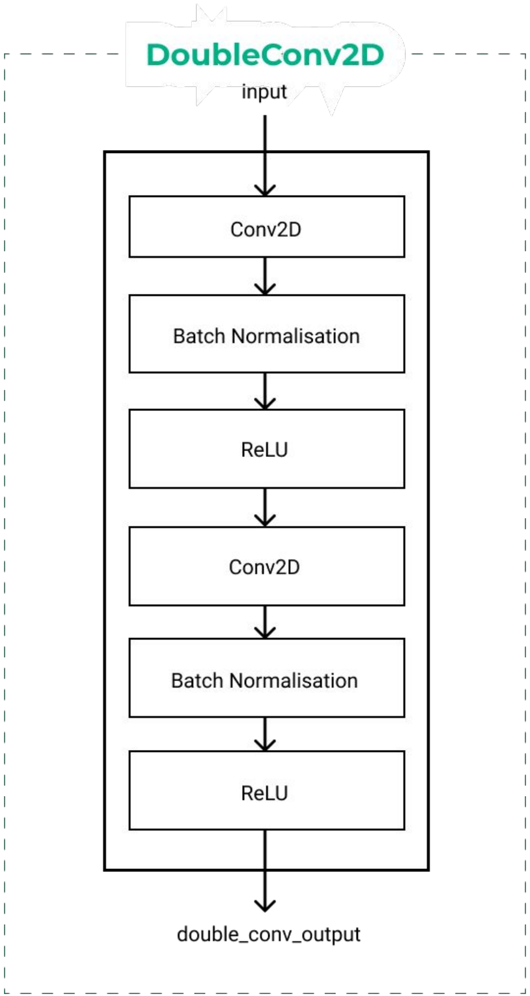
- U-Net follows a symmetric Encoder-Decoder architecture.
- Encoder (contracting path) reduces spatial dimensions, while increasing number of feature maps.
- The Decoder performs the inverse of the above.
- Concatenation ensures the use of lost information in reconstruction of features.
- Batch Normalization for regularization.
- Used Early Stopping, Model Checkpointing, CSV Logging

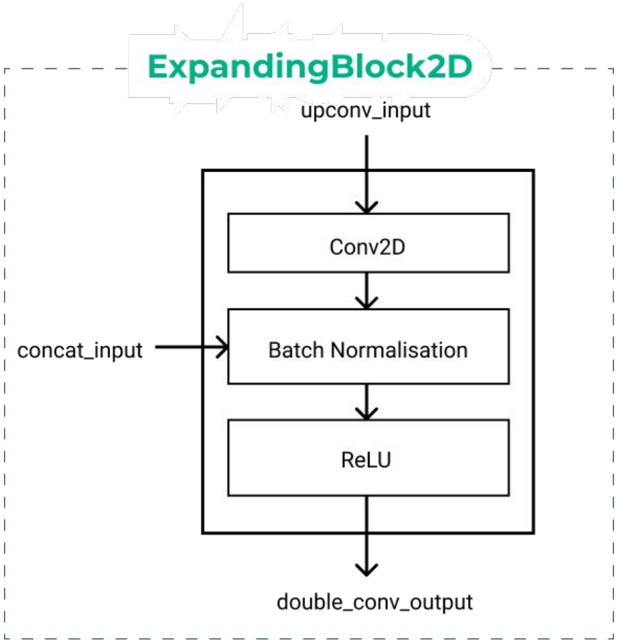
U-Net.

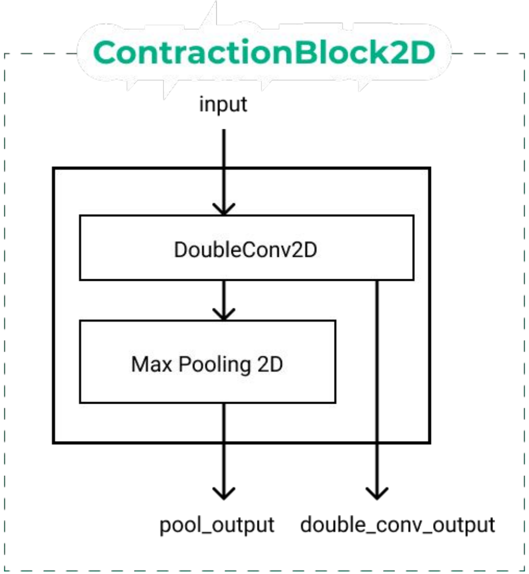


Neural Network Design.

U-Net Architecture.







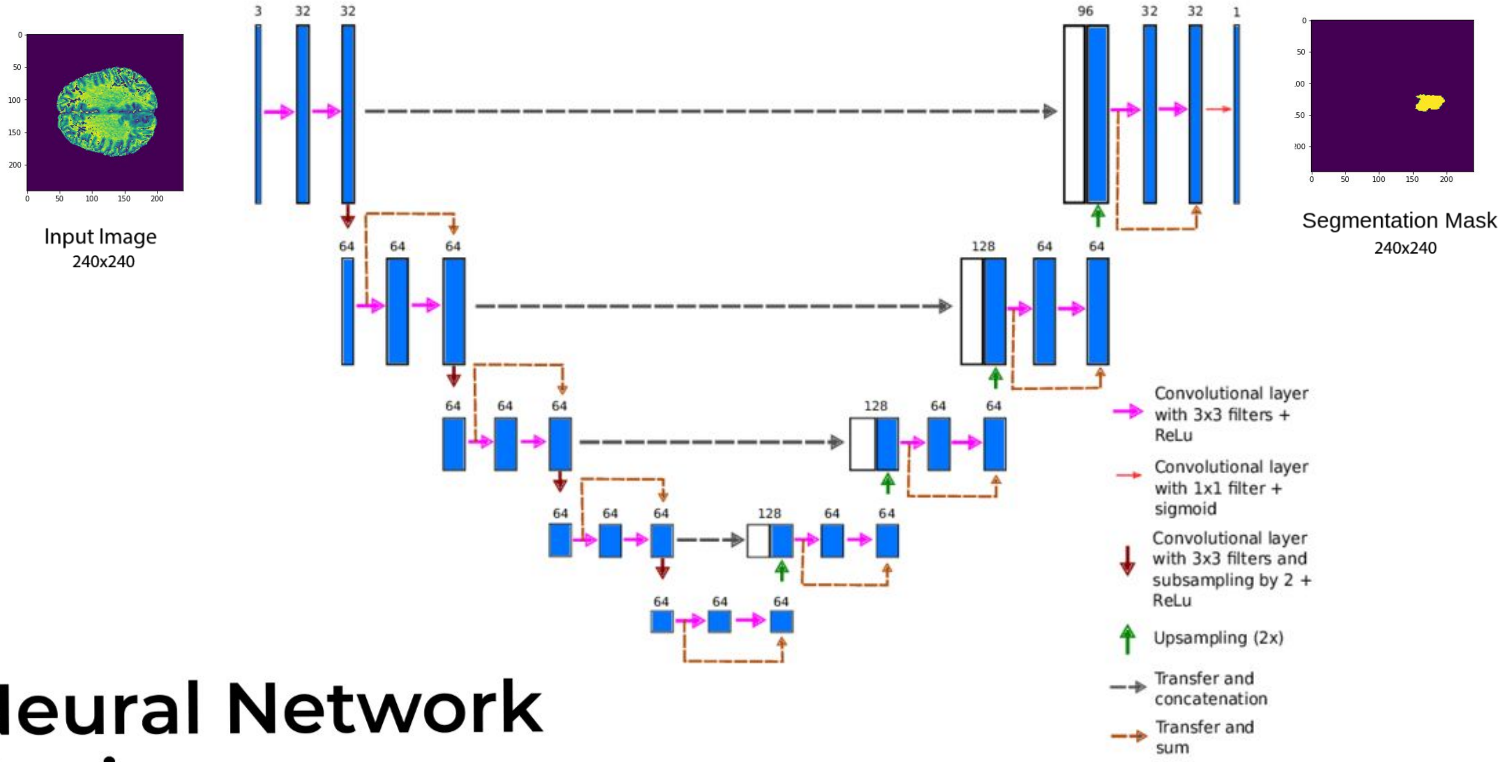
Best Model - U-Net.

```
def DoubleConv2D(n_filters, input, activation='relu', padding='same'):
    conv_output_a = Conv2D(n_filters, (3, 3), padding = padding)(input)
    bnorm_a = BatchNormalization()(conv_output_a)
    activated_a = Activation(activation)(bnorm_a)
    conv_output_b = Conv2D(n_filters, (3, 3), padding = padding)(activated_a)
    bnorm_b = BatchNormalization()(conv_output_b)
    activated_b = Activation(activation)(bnorm_b)
    return activated_b

def ContractingBlock2D(n_filters, input):
    double_conv_output = DoubleConv2D(n_filters, input)
    pool = MaxPooling2D(pool_size=(2, 2))(double_conv_output)
    return double_conv_output, pool

def ExpandingBlock2D(n_filters, upconv_input, concat_input):
    upsamp = Conv2DTranspose(n_filters, (2, 2), strides=(2, 2), padding = 'same')(upconv_input)
    conv_input = concatenate([upsamp, concat_input], axis=3)
    double_conv_output = DoubleConv2D(n_filters, conv_input)
    return double_conv_output
```

Best Model - U-Net.

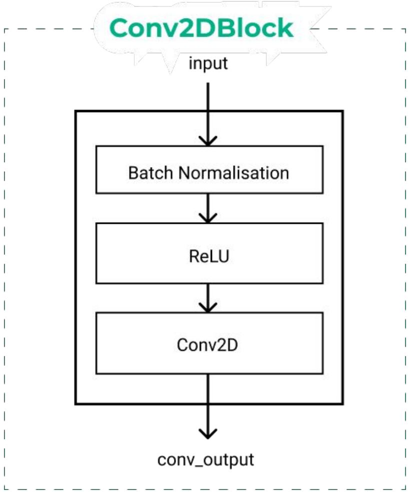
```
def UNet():
    input_tensor = Input((img_rows, img_cols, 1))

    conv_1, pool_1 = ContractingBlock2D(32, input_tensor)
    conv_2, pool_2 = ContractingBlock2D(64, pool_1)
    conv_3, pool_3 = ContractingBlock2D(128, pool_2)
    conv_4, pool_4 = ContractingBlock2D(256, pool_3)

    conv_5 = DoubleConv2D(512, pool_4)

    conv_6 = ExpandingBlock2D(256, conv_5, conv_4)
    conv_7 = ExpandingBlock2D(128, conv_6, conv_3)
    conv_8 = ExpandingBlock2D(64, conv_7, conv_2)
    conv_9 = ExpandingBlock2D(32, conv_8, conv_1)

    conv_10 = Conv2D(1, (1, 1), activation = 'sigmoid')(conv_9)

    model = Model(inputs = [input_tensor], outputs = [conv_10])
    return model
```

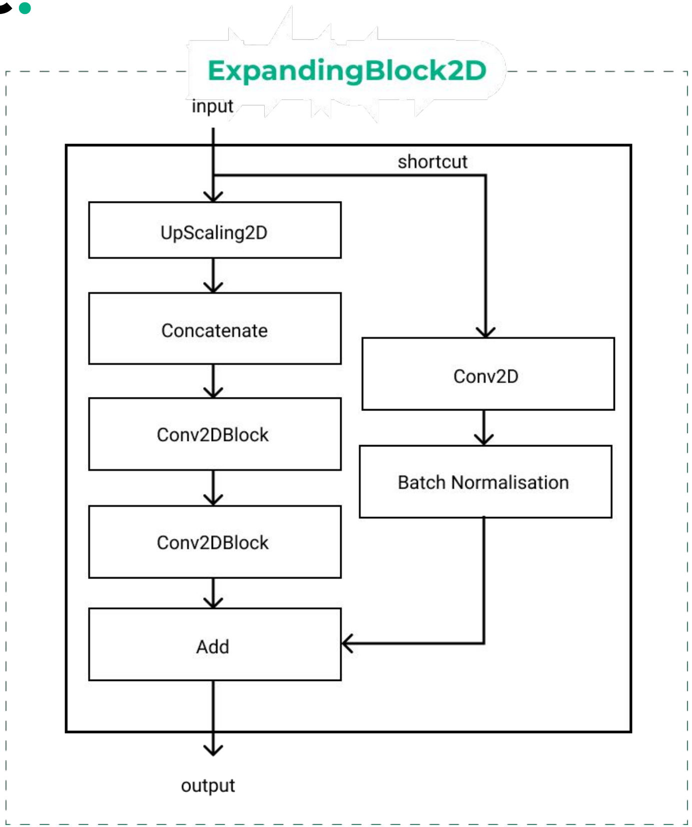
ResUNet.

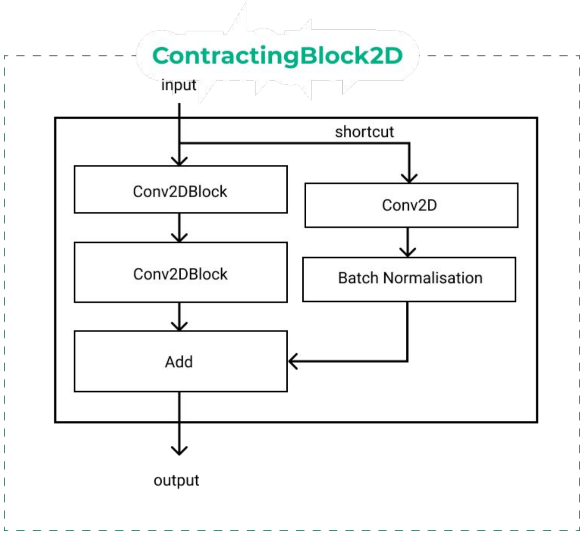


Neural Network Design.

ResUNet Architecture.



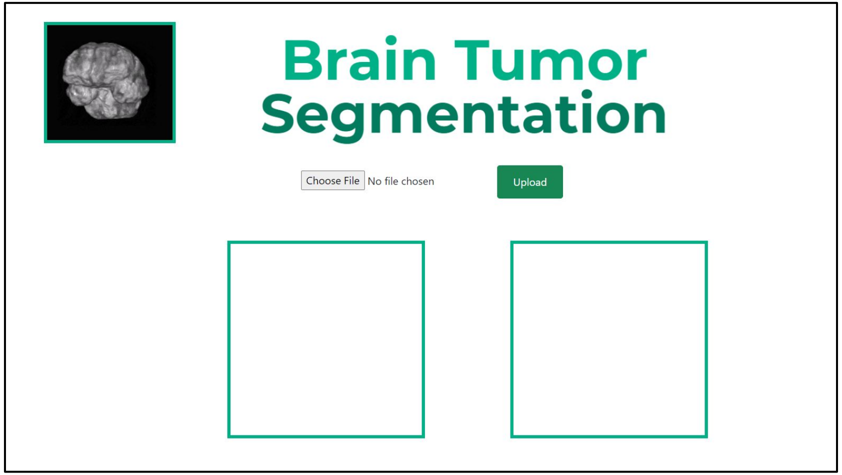


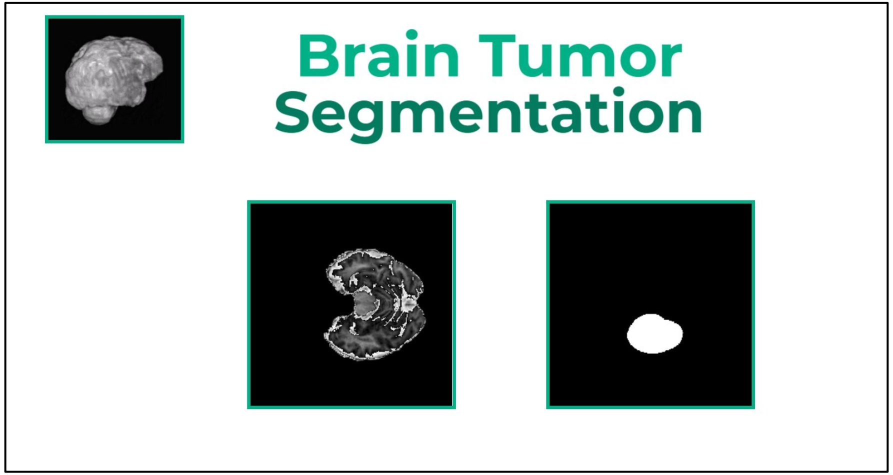


App Development.

- Developed FrontEnd User Interface using HTML, CSS and JavaScript
- Used Flask Framework for backend.
- Created API endpoints.
- Processed the input image.
- Connected Deep Learning Model.

App Development.



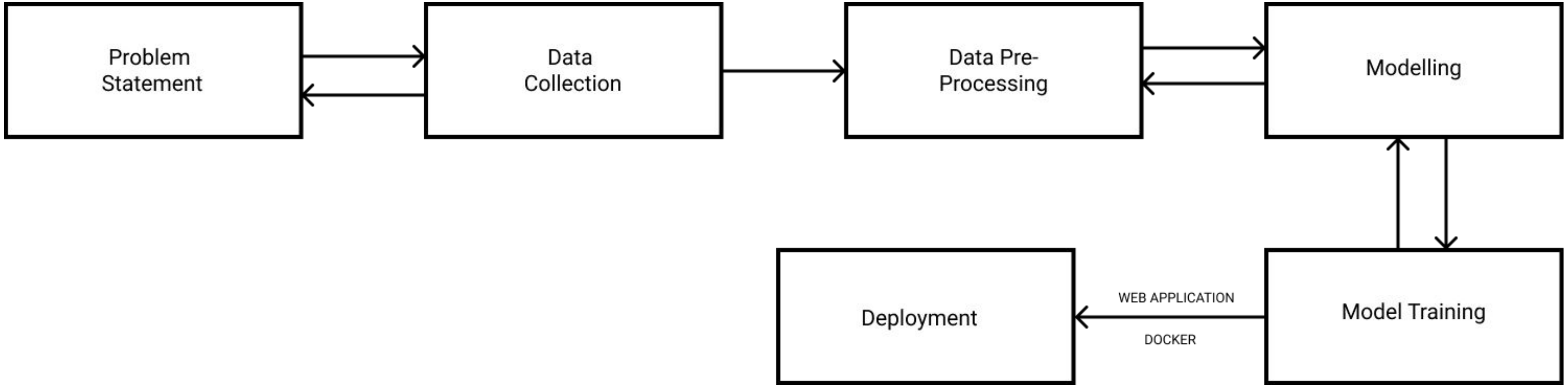


Cloud Architecture:

Cloud Resources.

- **Data Preprocessing and Modelling** -
  - Azure Notebook
- **Model Training** -
  - Azure Compute Instances
- **Deployment and Docker** -
  - Azure Container Registry, Azure Container Instances and Azure App Service

System Design.



- Pre-processing ⇒ Pixel intensity Normalization(Output_channel = 255 * (Input_channel - min) / (max-min)
- Modelling ⇒ UNet
- Using the Azure cloud services makes the training and storage very effortless.

Benefits of using

Azure.

1. Using machine as a service.
2. Easy and flexible building interface.
3. Wide range of supported algorithms.
4. Easy implementation of web services.
5. Extensibility.

Deployment:

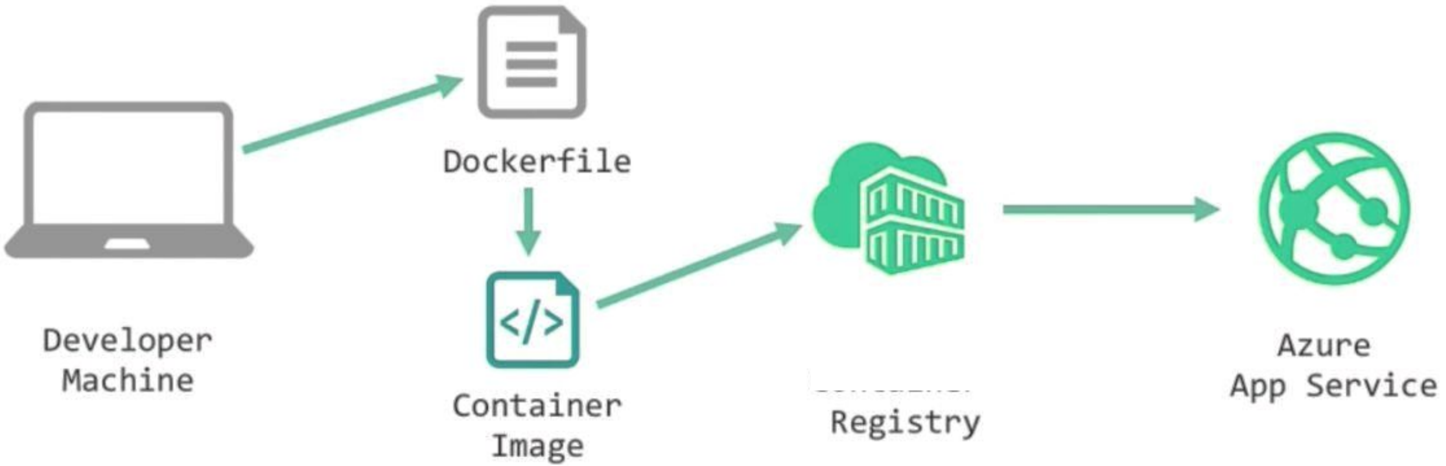
Flask & Docker.

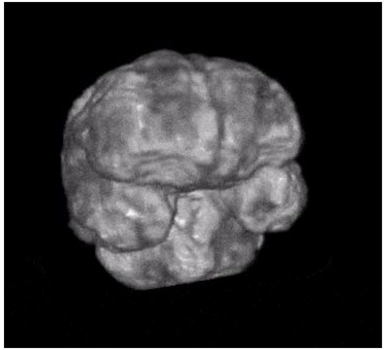
- A container is a type of software that packages up an application and all its dependencies so the application runs reliably from one computing environment to another.

- We made the web app using Flask and then containerised it using Docker containers.

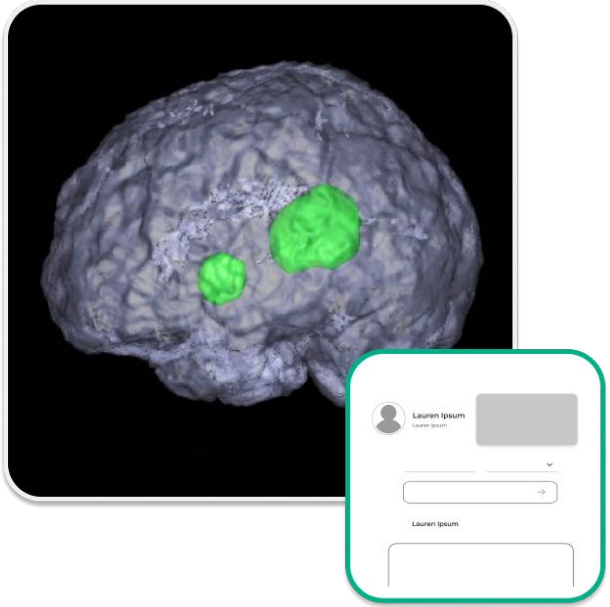
- Unlike VM, containers do not have guest OS which makes them lightweight, fast and portable.

Deployment Architecture.





Project Demo.

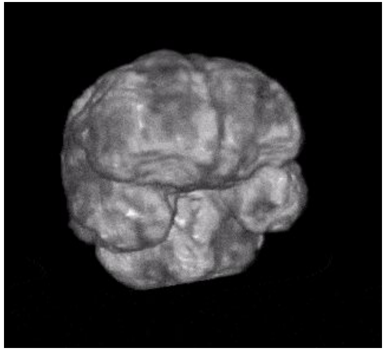


Future Enhancements.

- Provide 3D visualisation of output.
- Building a 3D UNet model.
- Functional dashboard with patient details.
- Develop models for other tumor types.

References.

- Dataset Details: https://www.med.upenn.edu/cbica/brats-2019/
- UNet Model: https://towardsdatascience.com/medical-images-segmentation-using-keras-7dc3be5a8524
- ResUNet Model: https://github.com/DuFanXin/deep_residual_unet/blob/master/res_unet.py
- For Measurement Of IoU and Localization Error Metrics: https://github.com/quantumjot/unet_segmentation_metrics
- Azure Docker Containers: https://cloudskills.io/blog/azure-docker-containers
- Azure DevOps KanBan: https://docs.microsoft.com/en-us/azure/devops/boards/boards/kanban-basics?view=azure-devops



The Team.

Jack Praveen Raj Ilango.

- Worked as Team Lead.
- Overlooked Azure DevOps and Team Organization.
- Designed KanBan Boards, Presentation and Architecture Diagrams.
- Defining Qualities: Leadership Skills, Detail Oriented, Strategic & Critical Thinking.

Shriya Dutta.

- Worked on Documentation & Presentation.
- Collaborated with team members to gather details of their respective tasks.
- Presentation and Wiki for Documentation.
- Defining Qualities: Operational Excellence, Stakeholder Engagement and Analytical Skills.

Aditya G.

- Worked as Programmer.
- Built the Image Segmentation model.
- Understanding the U-Net Model, performing model training on Azure.
- Defining Qualities: Problem-solving skills, Persistence, Meticulousness.

Tanmay Garg.

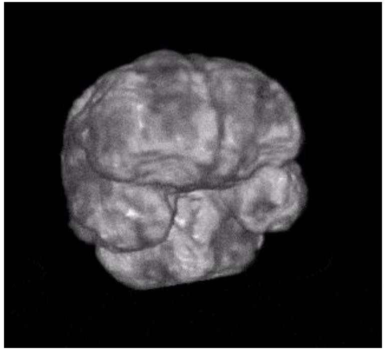
- Worked as Programmer.
- Did all the data cleaning and preprocessing.
- Built the WebApp (Frontend and Backend)
- Defining Qualities: Problem-solving skills, debugging skills, analytical skills

Megha N.

- Worked as Cloud Architect.
- Planning and generating the system Design.
- Setting up the cloud resources required.
- Defining Qualities: Effective Communicator, critical thinking, Analytical Skill.

Rishit Puri.

- Worked on Deployment.
- Containerised the web app using Docker Containers.
- Deployed the Docker Containers on Azure Container Instances.
- Defining Qualities: Quick learner, Eye for detailing and Ability to work as a team player.



Thank You.