Deep Learning for Detection of Glioblastoma

Prerna Luthra(PL2243), Dhairya Upadhyay(DAU7593)

Problem Statement

The detection and genomic identification of Brain tumor highly relies on invasive procedures for which biopsies are performed to extract tissue samples which then undergo histopathology for prognosis. The malignant brain tumor also known as glioblastoma has a survival of roughly 12 months with the prognosis being costly, dangerous and time consuming. There have been numerous studies outlining a specific approach of using medical imaging to identify the presence of a specific genomic sequence known as MGMT promoter methylation being a strong predictor of responsiveness to chemotherapy. In this project we aspire to successfully detect the presence of this genomic sequence with the help images procured through Magnetic Resonance Imaging (MRI) which will help the patients to receive a less invasive diagnosis and treatment.

Literature Survey

A brain tumor is an unintended growth of cells in a human's brain which can be benign or malignant. Its signs and symptoms vary greatly depending on its shape, size, location and rate of growth because of which the diagnosis and treatment can be delayed and changes from person to person. Glioblastoma is one type of cancer that can occur in brain or spinal cord and is found in cells called astrocytes that supports nerve cells. The diagnosis of this ranges from neurological exams to invasive biopsies. One of the treatment options include a surgical process to remove as much of the tumor as possible, but since it grows in the normal brain tissue complete removal of the tumor is not possible.

It is also important to note that it can take a few weeks to determine the results of these invasive biopsies. It's only after the initial results have arrived that a treatment can be started. Brain cancer is known to be aggressive in nature. This means that the longer a patient waits to get the treatment, the higher the likelihood is for the tumor to grow and spread.

If a method can be developed that is non-invasive in nature and minimizes the overall time spent on analysis of the genetic characterization of the tumor, then the treatment can be initiated a bit more quickly thereby increasing the survival chances of a particular patient. For this the researchers, have been trying to use medical imaging to quick prognosis by identifying the presence of MGMT promoter methylation. Initial work in this field was done using a single type of MRI scan which was later changed to use Multiparameter MRI (mpMRI), which is used to obtain a better 3-D image by combining T2-weighted(T2WI), diffusion weighted (DWI), dynamic contrast enhanced (DCEI) and if desired MR spectroscopy (MRSI) images. Machine Learning and deep learning appraoches have produced valuable results in identifying Glioblastoma.

The presence of MGMT promoter methylation in the MRI scans has proven to be good in prognosis of responsiveness of a patient to chemotherapy. This correlation has resulted in a non-invasive and quick detection of the tumor thereby reducing the response time to the condition.

Preliminary Investigation

For this project we primarily had access to BraTS 2021 Dataset. Details for this dataset can be found in 'Final Methodology and Implementation' Section below. For detection of MGMT promoter status, we created a simple 3D CNN model using this dataset. The description of the model is provided in Figure 1. We used Leaky ReLU, a type of activation function based on a ReLU, but it has a small slope for negative values instead of a flat slope. The slope coefficient is determined before training, i.e. it is not learnt during training which in our case is to 0.01. We trained our model using Adam's optimizer with a binary cross entropy loss function. After training the model for 30 epochs, after which there is little to no accuracy improvement, we observe the accuracy to be above 60% for all 4 types of MRI scans individually. The accuracy results are described in the table below.

While 3D CNN gave us decent results, we wanted to try out something different. In the past, researchers have combined the task of brain segmentation (i.e. identifying the region of tumor in brain MRI scans) with the task of detecting of MGMT promoter status, which has proved to give good accuracy results. We came across a model called U-net, which is a convolutional network architecture for fast and precise segmentation of images. This model leverages the model's performance on segmentation to improve the model accuracy further. In figure 2, each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower

```
ThreeDNetwork(
  (block1): Sequential(
    (0): Sequential(
      (0): Conv3d(1, 64, kernel_size=(3, 3, 3), stride=(2, 2, 2))
      (1): LeakyReLU(negative_slope=0.01)
      (2): MaxPool3d(kernel_size=(2, 2, 2), stride=(2, 2, 2), padding=0, dilation=1, ceil_mode=False)
      (3): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (1): Sequential(
      (0): Conv3d(64, 128, kernel_size=(3, 3, 3), stride=(2, 2, 2))
      (1): LeakyReLU(negative_slope=0.01)
      (2): MaxPool3d(kernel_size=(2, 2, 2), stride=(2, 2, 2), padding=0, dilation=1, ceil_mode=False)
      (3): BatchNorm3d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (fc): Sequential(
    (0): Linear(in_features=86400, out_features=1024, bias=True)
    (1): LeakyReLU(negative_slope=0.01)
    (2): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.2, inplace=False)
    (4): Linear(in_features=1024, out_features=1, bias=True)
  )
)
```

Figure 1: Pytorch 3D CNN Model summary

left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. Unlike the previous CNN model, this one takes into consideration all 4 MRI scans simultaneously to predict the MGMT status promoter. While this was an interesting model, it required us to use BraTS 2020 segmentation dataset which consisted of MRI data frames as NIfTI files. This code along with the 4 types of MRI scans also used segmentation which were identified by trained professionals to be the right portions of the scans for better results. A sample segmented image is displayed in Figure 3. Model accuracy and loss graphs are presented in the figures 4 and 5.

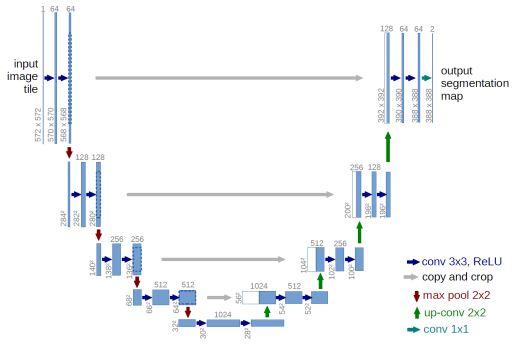

Figure 2: Implementation of U-Net Architecture

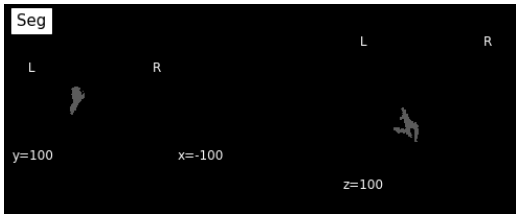

Figure 3: Segmented Brain Tumors

Unfortunately, we didn't have full access to to their corresponding mgmt values as outlined in our problem statement. Hence given the dataset restrictions, we decided to stick to BraTS 2021 dataset and decided to simply test different flavors of CNNs to see if we are able to get decent accuracy results for detection of MGMT promoter status.

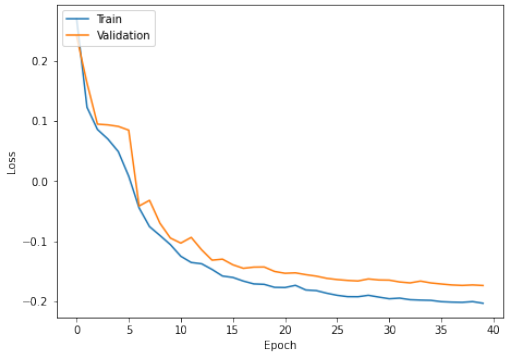

Figure 4: Training and validation loss graph.

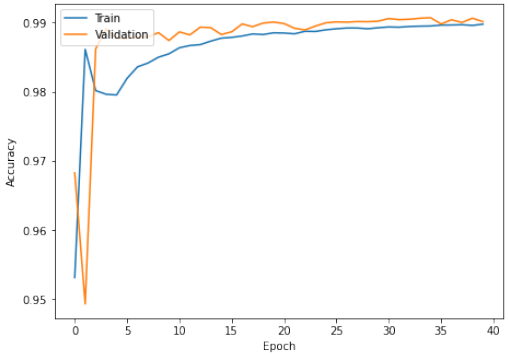

Figure 5: Training and validation accuracy graph.

Final Methodology and Implementation

Dataset

This dataset is part of the BraTS 2021 challenge which is organized by the Radiological Society of North America(RSNA), the American Society of Neuroradiology (ASNR), and the Medical Image Computing and Computer Assisted Interventions (MICCAI) society. The dataset has multi-parametric magnetic resonance imaging (mpMRI) consisting of Fluid-attenuated inversion recovery (FLAIR), T1 weighted image - pre-contrast(T1w), T1wCE - post contrast and T2 weighted images(T2w) which are in DICOM format. We have in all brain scans of 672 subjects of which 585 are in the training set and 87 are in the testing set with 4 mpMRI for each subjects. Each type of mpMRI scan consists of a set of image frames. These frames are available to us in DICOM format. For each of the subject ID's there is an associated value of MGMT promoter in the scans which we have to successfully predict on the test dataset.

Data Visualization

The two most common types of MRI image scans are T1W and T2W scans. The timing of radiofrequency pulse sequences used to make T1W scans results in images which highlight fat tissue within the body. On the other hand,

the timing of radiofrequency pulse sequences used to make T2W scans results in images which highlight fat as well as water within the body. A third commonly used MRI scan type is the Fluid Attenuated Inversion Recovery (Flair). The Flair sequence is similar to a T2W image except that the timing of radiofrequency pulse sequence is much longer. By doing so, abnormalities remain bright but normal CSF (i.e. Cerebrospinal Fluid) is attenuated and made dark. Figure below shows FLAIR, T1w, T1wCE and T2w Scans.

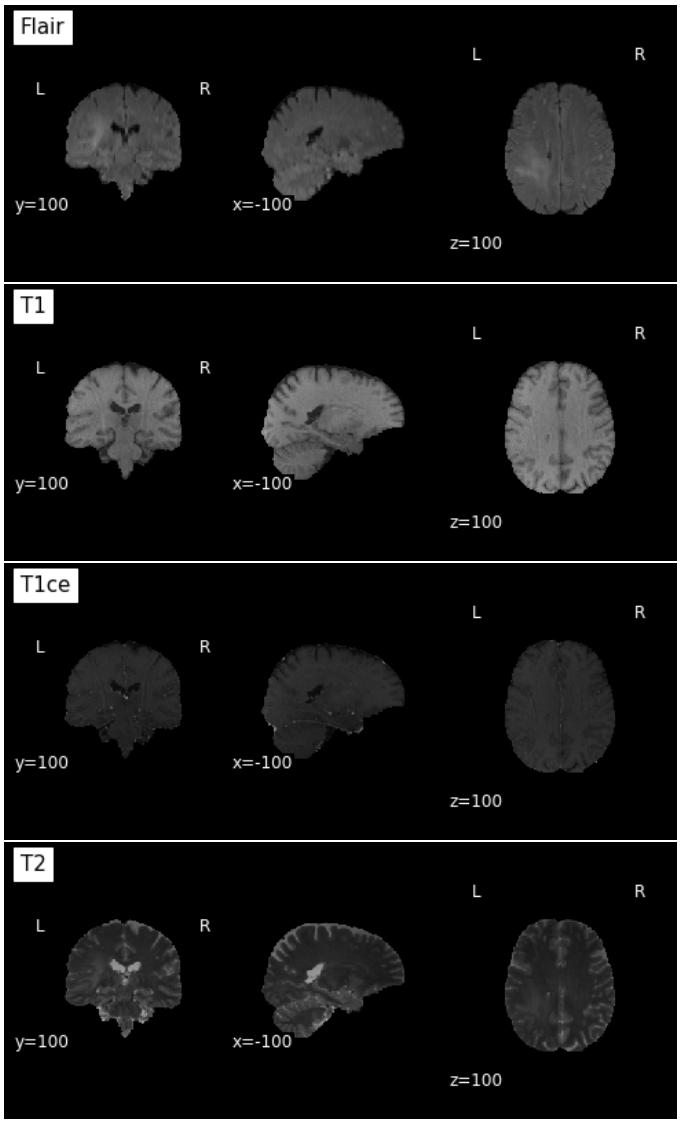

Figure 6: FLAIR, T1w, T1wCE, T2w Scans respectively

Training, Validation and Test Data Split

Following are the total number of image frames used in training, validation and test datasets for different image scans.

Total image frames per MRI scan type		
MRI Type	Dataset Type	Total Image Frames
FLAIR	Training	7306
	Validation	333
	Test	1119
T1W	Training	7347
	Validation	389
	Test	1187
T1WCE	Training	9312
	Validation	387
	Test	1441
T2W	Training	9830
	Validation	325
	Test	1463

Data Pre-processing

The MRI image frames available to us have different pixel spacing given they are generated from different MRI machines. In this step, we rescaled the images by a factor of 1/255. In addition, we filtered out noisy image frames by removing frames above a certain Hounsfield Unit threshold. The final dimension of the image frames was 300 × 300.

Data Augmentation

We decided to augment the data to see if the accuracy improves. First, we only rotated the image frames by 0.2 degree and flipped some of the images horizontally to see if the accuracy improves. Next, we decided to include additional augmentation techniques such as zooming in by a factor of 0.2, randomly translating images vertically or horizontally, and setting the brightness range within $[0.8, 1.2]$ to see if accuracy improves.

Models and Architecture

We explored various flavors of CNN to predict the MGMT promoter status. We started by exploring the possibility of using VGG16. Further, we explored slightly more complicated architectures like ResNet50 and EfficientNetB3 to see how they compare to VGG16.

Input and Output of Models: For all different models/architectures, the input is a set of all image frames (resized to a dimension of 300 x 300) for a particular MRI scan type whereas the output is the probability indicating the existence of MGMT promoter status per MRI scan. Also, we trained our architectures using the Adam's optimizer and binary cross entropy loss function.

VGG16: VGG is a CNN model proposed by K. Simonyan and A. Zisserman. The model primarily deals with a large image dataset and investigates the depth of layers with a small convolutional filter size (3 × 3). Pre-trained VGG models are available with different layer lengths ranging from 11 to 19.
In this project, we plan to primarily use VGG16 model to predict the value of MGMT promoter status. The description of all the layers is described in Figure 7. Note that dropout is set to 0.4. Layer $dense\_4$ uses a $relu$ activation whereas

$pred$ layer uses sigmoid activation to retrieve the probability.

```
Model: "VGG16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, None, None, 3)]   0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, None, None, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, None, None, 64)    36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, None, None, 64)    0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, None, None, 128)   73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, None, None, 128)   147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, None, None, 128)   0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, None, None, 256)   295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, None, None, 256)   0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, None, None, 512)   1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, None, None, 512)   0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, None, None, 512)   0
_________________________________________________________________
avg_pool (GlobalAveragePooli (None, 512)               0
_________________________________________________________________
batch_normalization_8 (Batch (None, 512)               2048
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                16416
_________________________________________________________________
batch_normalization_9 (Batch (None, 32)                128
_________________________________________________________________
dropout_9 (Dropout)          (None, 32)                0
_________________________________________________________________
pred (Dense)                 (None, 1)                 33
=================================================================
Total params: 14,733,313
Trainable params: 17,537
Non-trainable params: 14,715,776
_________________________________________________________________
```

Figure 7: VGG16 Model summary

ResNet50: ResNet stands for Residual Network. It was first introduced by K. He, X. Zhang, S. Ren, and J. Sun. These networks make use of residual blocks and "skip connections" to improve the accuracy of the models.
While ResNet has different flavors, for this project, we used ResNet50 model to predict the value of MGMT promoter status. The description of all the layers is described in Figure 8. Note that, in the figure, we skipped a couple of layers in the middle as we couldn't fit the entire model in one image. The dropout is set to 0.4. Layer $dense\_13$ uses a $relu$ activation whereas $pred$ layer uses sigmoid activation to retrieve the probability.

```
Model: "ResNet50"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_15 (InputLayer)           [(None, None, None,  0
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, None, None, 3 0           input_15[0][0]
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, None, None, 6 9472        conv1_pad[0][0]
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, None, None, 6 256         conv1_conv[0][0]
__________________________________________________________________________________________________
conv1_relu (Activation)         (None, None, None, 6 0           conv1_bn[0][0]
__________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, None, None, 6 0           conv1_relu[0][0]
__________________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, None, None, 6 0           pool1_pad[0][0]
__________________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, None, None, 6 4160        pool1_pool[0][0]
......
......
......
__________________________________________________________________________________________________
conv5_block3_3_conv (Conv2D)    (None, None, None, 2 1050624     conv5_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, None, None, 2 8192        conv5_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_add (Add)          (None, None, None, 2 0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_out (Activation)   (None, None, None, 2 0           conv5_block3_add[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)         0           conv5_block3_out[0][0]
__________________________________________________________________________________________________
batch_normalization_26 (BatchNo (None, 2048)         8192        avg_pool[0][0]
__________________________________________________________________________________________________
dropout_26 (Dropout)            (None, 2048)         0           batch_normalization_26[0][0]
__________________________________________________________________________________________________
dense_13 (Dense)                (None, 32)           65568       dropout_26[0][0]
__________________________________________________________________________________________________
batch_normalization_27 (BatchNo (None, 32)           128         dense_13[0][0]
__________________________________________________________________________________________________
dropout_27 (Dropout)            (None, 32)           0           batch_normalization_27[0][0]
__________________________________________________________________________________________________
pred (Dense)                    (None, 1)            33          dropout_27[0][0]
==================================================================================================
Total params: 23,661,633
Trainable params: 69,761
Non-trainable params: 23,591,872
__________________________________________________________________________________________________
None
```

Figure 8: ResNet50 Model summary

EfficientNetB3: EfficientNet was introduced by Tan at el. It is a CNN architecture that uniformly scales all dimensions of depth/width/resolution using a compound coefficient.
For this project, we focused on using EfficientNetB3 model to predict the value of MGMT promoter status. The description of all the layers is described in Figure 9. Note that, in the figure, we skipped a couple of layers in the middle as we couldn't fit the entire model in one image. The dropout is set to 0.4. Layer $dense$ uses a $relu$ activation whereas $pred$ layer uses sigmoid activation to retrieve the probability.

Model Training and Hyper-parameter Selection

We decided to use weights which were pre-trained on ImageNet as training all the different models above from scratch can be time-consuming. Adam's optimizer was used for training purposes with a learning rate of $1e-3$. We realized that training the dataset over too many epochs could lead to overfitting which is why we decided to use early stopping. For early stopping, we monitored validation loss for $patience = 5$ and $mode = min$ i.e. we stopped training if the validation loss wasn't decreasing in the last 5 epochs. Further, in order to optimize, we also set $ReduceLROnPlateau$ parameter which reduced the learning rate by a factor of 0.5 if no improvement was seen in validation loss over last 2 epochs. Lastly, we used the binary cross-entropy loss function to measure the loss over epochs.

Model Evaluation

We evaluated our model by calculating the overall loss, accuracy as well as area under the receiver operator characteristic curve (AUC). The model with the best validation loss was chosen to predict the MGMT values for test dataset.

Figure 9: EfficientNetB3 Model summary

Results

The following tables summarizes the test dataset results for different models:

MRI Level Results

These results are calculated at MRI Level. This means that we simply combined image frames corresponding to a particular MRI Scan Type (eg. FLAIR, T1W etc) without caring about which particular patient the scan belongs to. Here, we are simply interested in understanding which type of scans give us the best accuracy.

VGG16 Results:

VGG16 - Test Dataset Results without Data Augmentation			
MRI Type	Loss	Accuracy	AUC
FLAIR	0.528	0.810	0.847
T1W	0.512	0.821	0.744
T1WCE	0.470	0.903	0.416
T2W	0.529	0.714	0.559

Table 1: VGG16 - Results without data augmentation

VGG16 - Test Dataset Results with Data Augmentation			
MRI Type	Loss	Accuracy	AUC
FLAIR	0.614	0.710	0.847
T1W	0.621	0.697	0.694
T1WCE	0.649	0.644	0.922
T2W	0.533	0.743	0.786

Table 2: VGG16 - Results with data augmentation

ResNet50 Results:

ResNet50 - Test Dataset Results without Data Augmentation			
MRI Type	Loss	Accuracy	AUC
FLAIR	0.586	0.670	0.549
T1W	0.844	0.220	0.382
T1WCE	0.806	0.519	0.353
T2W	0.457	0.781	0.508

Table 3: ResNet50 - Results without data augmentation

ResNet50 - Test Dataset Results with Data Augmentation			
MRI Type	Loss	Accuracy	AUC
FLAIR	0.442	0.886	0.772
T1W	0.694	0.553	0.575
T1WCE	0.657	0.604	0.653
T2W	0.408	0.942	0.764

Table 4: ResNet50 - Results with data augmentation

EfficientNetB3 Results:

EfNetB3 - Test Dataset Results without Data Augmentation			
MRI Type	Loss	Accuracy	AUC
FLAIR	0.618	0.685	0.862
T1W	0.659	0.636	0.151
T1WCE	0.684	0.523	0.951
T2W	0.619	0.760	0.861

Table 5: EfficientNetB3 - Results without data augmentation

EfNetB3 - Test Dataset Results with Data Augmentation			
MRI Type	Loss	Accuracy	AUC
FLAIR	0.554	0.917	0.832
T1W	0.500	0.922	0.396
T1WCE	0.526	0.892	0.952
T2W	0.563	0.970	0.299

Table 6: EfficientNetB3 - Results with data augmentation

Patient Level Results

These results are calculated at Patient Level. A particular patient may have a combination of different MRI scans. Thus, it is important to compute the accuracy at patient level as well by using a mix of different MRI scans. This was achieved by aggregating all the MGMT promoter status predictions for different image frames (or MRI scans) for a particular patient and ultimately computing the accuracy. Patient level accuracies were computed both with and without Data Augmentation (DA).

Test Dataset Results at Patient Level		
Model Type	without DA	with DA
VGG16	0.655	0.448
ResNet50	0.414	0.586
EfficientNetB3	0.482	0.804

Table 7: Test Dataset Results at Patient Level for different models

Observations and Discussion of Results

Best Overall Model: Out of the three models, it is observed that EfficientNetB3 with data augmentation produces the best accuracy at both MRI Level as well as Patient Level. At MRI Level, the accuracy is observed to be > 0.9 for most scans. At Patient Level, the accuracy is observed to be 0.80.

Best MRI Scan Type: It is noticed that T2W scans give the best accuracy regardless of the model we use. For example, if we were to look at the results of different models with data augmentation, the accuracy of T2W scans is 0.743, 0.942 and 0.970 with VGG16, ResNet50 and EfficientNetB3 respectively. This tells us that if we had access to only T2W scans for all patients, we would still be able to predict the MGMT promoter status with a good accuracy.

Data Augmentation vs. No Data Augmentation: It is observed that both ResNet50 and EfficientNetB3 improve the results significantly when the data is augmented. For example, if we look at patient level results, accuracy for ResNet50 improves from 0.414 to 0.586 with data augmentation whereas accuracy for EfficientNetB3 improves from 0.482 to 0.804 with data augmentation. Similarly, a visible jump can be seen in accuracies for both ResNet50 and EfficientNetB3 at MRI Level as well.
It is interesting to observe that the accuracy for VGG16 decreases with data augmentation. The reason for this could be that the model is small in capacity. At the time of training, we observed that the loss for both training and validation sets was quite high (> 0.7) for all image scans. This could potentially be a sign of underfitting. This could have been avoided by either training the model for a longer time or moving to a model with larger capacity or depth. Unfortunately, we couldn't train VGG16 with data augmentation for a longer time as it was already taking way too long to train. We did move to models with larger capacity though (eg. ResNet50 and EfficientNetB3) and results for those clearly show that data augmentation helps in improving the overall accuracy.

Project Outcome and Conclusion

Overall, the goal of this project is to come up with a model that would give us a decent accuracy for predicting the MGMT promoter status in MRI scans. In addition, we wanted to deepen our knowledge of various deep learning models by applying them to the area of medical imaging analysis. We achieved this goal by trying out different models and architectures. We started our preliminary investigation with 3D CNN and UNets. Later we moved to trying more concrete flavors of CNN such as VGG16, ResNet50 and EfficientNetB3. We ultimately observed that EfficientNetB3 gives us the best accuracy for detecting MGMT promoter status on the BRaTs2021 dataset at both MRI and Patient Level. Results for this model can be found in Table 5, 6 and 7.

Code

Code can be found here: `https://github.com/prernaluthra/DL_Project`

References

American Journal of Neuroradiology
Yogananda CGB, Shah BR, Nalawade SS, Murugesan GK, Yu FF, Pinho MC, Wagner BC, Mickey B, Patel TR, Fei B, Madhuranthakam AJ, Maldjian JA. MRI-Based Deep-Learning Method for Determining Glioma MGMT Promoter Methylation Status. AJNR Am J Neuroradiol. 2021 May;42(5):845-852. doi: 10.3174/ajnr.A7029. Epub 2021 Mar 4. PMID: 33664111; PMCID: PMC8115363.

International Workshop on Radiomics and Radiogenomics in Neuro-oncology
Booth T.C. et al. (2020) Machine Learning and Glioblastoma: Treatment Response Monitoring Biomarkers in 2021. In: Kia S.M. et al. (eds) Machine Learning in Clinical Neuroimaging and Radiogenomics in Neuro-oncology. MLCN 2020, RNO-AI 2020. Lecture Notes in Computer Science, vol 12449. Springer, Cham.

Institute of Electrical and Electronics Engineers
S. E. I. El kaitouni and H. Tairi, "Segmentation of medical images for the extraction of brain tumors: A comparative study between the Hidden Markov and Deep Learning approaches," 2020 International Conference on Intelligent Systems and Computer Vision (ISCV), 2020, pp. 1-5, doi: 10.1109/ISCV49265.2020.9204319.

International Workshop on Radiomics and Radiogenomics in Neuro-oncology
S. Grampurohit, V. Shalavadi, V. R. Dhotargavi, M. Kudari and S. Jolad, "Brain Tumor Detection Using Deep Learning Models," 2020 IEEE India Council International Subsections Conference (INDISCON), 2020, pp. 129-134, doi: 10.1109/INDISCON50162.2020.00037.

Blog
"Introduction to Working with MRI Data in Python"

Dataset
"U.Baid, et al., "The RSNA-ASNR-MICCAI BraTS 2021 Benchmark on Brain Tumor Segmentation and Radiogenomic Classification", arXiv:2107.02314, 2021."

Proceedings of the 36th International Conference on Machine Learning
Mingxing Tan, Quoc Le; "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks"

Computer Vision and Pattern Recognition
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition"