RSNA-MICCAI Brain Tumor Radiogenomic Classification with 3D CNN

Quoc Anh (Alan) Bui
University of Massachusetts Amherst
qhbui@umass.edu

Gregory Fleming
University of Massachusetts Amherst
gfleming@umass.edu

***Abstract*—We present an approach to reducing the number of required invasive surgeries in patients with glioblastoma through the use of computer vision to detect MGMT promoter methylation in tumors. The approach utilizes convolutional neural networks trained on FLAIR mpMRI brain scans with indicated MGMT promoter methylation values.**

***Index Terms*—Glioblastoma, MGMT promoter methylation, FLAIR, 3D CNN.**

I. Introduction

Cancer, especially of the brain, is a devastating diagnosis for a patient to receive. With this diagnosis comes all but a guarantee of months of hardship ahead. Glioblastoma is a form of brain cancer and it has one of the worst prognoses, with median survival time being less than 1 year for those affected by it. Unfortunately, glioblastoma is the most common form of brain cancer for adults. Currently, a treatment option for glioblastoma is alkylating chemotherapy, but it does not work for all patients. To tell if chemotherapy will be effective for a patient, surgeons must take a tissue sample of the tumor, and send the sample to be tested for MGMT promoter methylation. MGMT promoter methylation has been proven to be an indicator for a patient responding well to alkylating chemotherapy [3]. This strategy works well enough, but it involves an invasive surgery and time-consuming testing. What if we can find a way to detect MGMT promoter methylation more quickly and without an invasive surgery? If we can do this, it will save patients and surgeons time, and allow the patients to start chemotherapy or some other treatment option more quickly. This will have the further benefits of potentially increasing the likelihood of a recovery, since the early stages of cancer treatment are so vital. In this paper we describe our approach to using MRI brain scans to predict the presence of MGMT promoter methylation.

II. Methodology

A. Dataset

We will be using a real-world dataset collected and authorized by the RSNA (Radiological Society of North America) and the MICCAI Society (Medical Image Computing and Computer Assisted Intervention Society) released in 2021. The dataset contains 585 cases (patients) and their mpMRI brain scans divided in 4 categories (subfolders). The 4 types of mpMRI scans include: FLAIR, T1w, T1Gd and T2. Correspondingly, the label of each case is `MGMT_value` - 0: unmethylated and 1: methylated. The distribution of the `MGMT_value` is ∼53% for 0 and ∼47% for 1. The dataset is **raw** and **uncleaned** for any model to use. Thus, any necessary preprocessing steps must be performed before feeding into the applicable model.

In this project, we chose to use **FLAIR**. We chose to use only one type of scan because the size of the whole dataset containing all four types of scans was roughly 132GB: storing and training the model with all of the data would be extremely challenging in terms of storage and processing power. We chose the FLAIR scans specifically for a number of reasons. We found that of the four types of scans, FLAIR had the highest repetition time (TR) and time to echo (TE) [4]. TR is the time between pulses applied to the same slice of brain, and TE is the time between the emission of the pulse and the return time. Due to these high TR and TE values, abnormalities in FLAIR scans are more accentuated than in other scans. Since our focus is on tumors, which are abnormalities, we postulated that using the scan that highlighted them the most would be the best for analyzing them.

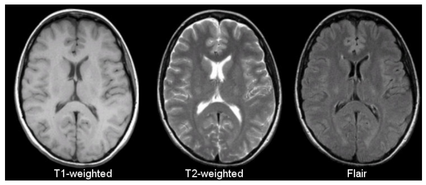


Fig. 1. MRI scans of T1-weighted, T2-weighted and Flair

B. Convolutional Neural Networks

In this project, we decide to use *Convolutional Neural Network* (CNN) as we believe it is the most sufficient computer vision model for this task. Since the model was not introduced in the lectures, it is worth to go over the basic overviews of the network. (2D) CNN consists the following essential components:

- **Convolution Layer** (`Conv2D`): Input of size (H, W, C) 'image' and output of size (H', W', C') 'image', where H: height, W: width and C: channels - usually `RGB` values. What `Conv2D` does underneath depends on its *filters*.

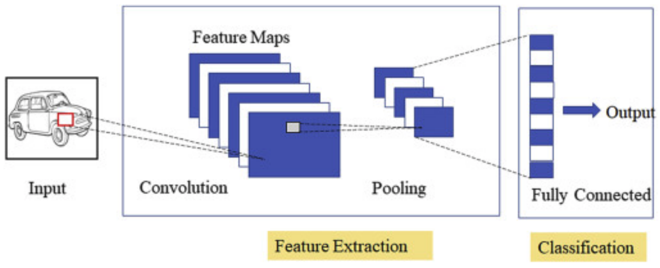


Fig. 2. Simple 2D CNN illustration with an input image of a car

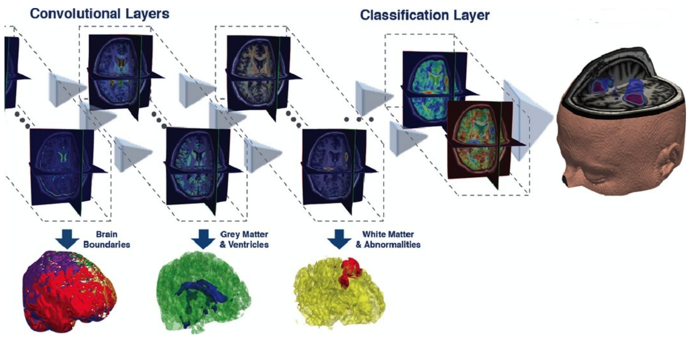


Fig. 3. CNN intuition from a similar project (*DeepMedic* released in 2015)

- **Filter**: The heart of CNN. Similar to any other ML models, filter is a 'window' of learnable weights (parameters) that slides over regions of the input image along the height and width dimensions to compute the dot product between the filter and the corresponding image portions. That explains why the dimensions of the filter is (FH, FW, C). After the first iteration of the filter, it will produce a **Feature Map** which reveals a certain hidden feature (any unique characteristics of the car like wheels, rectangle shape, etc. in Fig. 2.) of the image. Adding more filters will produce more feature maps and thus help the model learn more features of the image.
- **Pooling layer**: Downsampling the feature maps to reduce the number of parameters and output the spatial average of the feature maps in order to have compatible shape before feeding into *fully connected layer*.
- **Fully Connected Layer:** Input of size (N, D) and output of size (N, C): learns the non-linear combinations of those extracted features from those `Conv2D` layers.

However, notice that we are working with time-lapse mpMRI scans of the brain instead of a single image (e.g a car image in Fig. 2.), it is worth to note that we use `Conv3D` layer and consequently `Conv3D` filter that convolves or slides over an additional depth D dimension (the total number of mpMRI scans that construct or represent the brain) of the 'image'. This guarantees that the model can capture the general structure of the brain. Therefore, we introduce another dimension to our `Conv3D` input (H, W, D, C) and the filter dimensions as the result become (FH, FW, FD, C). As the result, the output of the `Conv3D` layer is (H', W', D', C'). The rest follows the same procedure.

In summary, at each Convolution Layer, the model will learn some hidden features from different parts of the brain that contribute to the presence of MGMT (Fig. 3.)

C. Data Processing and Loader

*1) **Data Loading**:* Recall that our input needs a depth D dimension, i.e. number of images (the other dimensions are the default dimensions of the scan so we do not need to pay too much attention for those). However, a problem arises: the number of mpMRI scans vary among patients. We determine that the average number of scans is ~ 120 and the min number is 15. Thus, we decide to use 2^N number of images, in this case the ideal N values are 5 or 6. Finally, we go with 32 images in this project (N is a hyperparameter).

*2) **Data Padding**:* But what about the case where the number of images less than 32? We decide to do zero padding to the front and the back: basically add 0 pixel value images to make up total of 32. An interesting question is that why we did not generate new images by taking the average of pixel values. This would definitely produce a meaningful image in terms of pixel values but visually non-sense in terms of actual MRI scan; we do not want to ruin the structure of the brain by adding new images to the series of default scans.

*3) **Data Augmentation**:* We apply some rotations such as `cv2.ROTATE_180`, `cv2.ROTATE_90_CLOCKWISE` and `cv2.ROTATE_90_COUNTERCLOCKWISE` if necessary so that all the images have uniform rotation.

*4) **Data Normalization**:* We use min-max normalization, which is one of the most recommended ways of normalizing pixels of CT scans, so that the pixel values stay in [0,1].

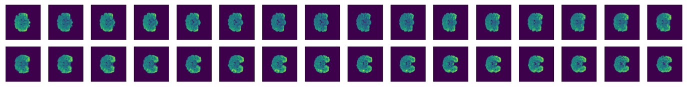


Fig. 4. Patient originally has enough number of mpMRI scans

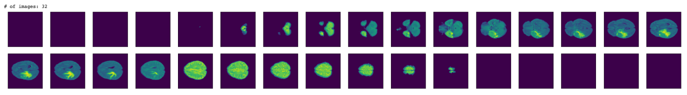


Fig. 5. 0 pixel values padding when not enough mpMRI scans

*5) **Data Splitting**:* Once the data is cleaned and visualized, it is split 80% for training, 10% for validation and 10% for testing in stratified fashion using the labels (i.e MGMT value).

*6) **Data Loader**:* A class that randomly generates data batches of size m for training. This helps the model to reduce number of computations and converge faster.

D. Model Architecture

Now we have an idea of what model we are going to use and what data we are going to feed into the model.

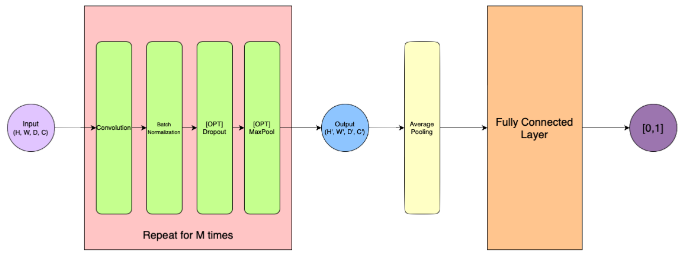
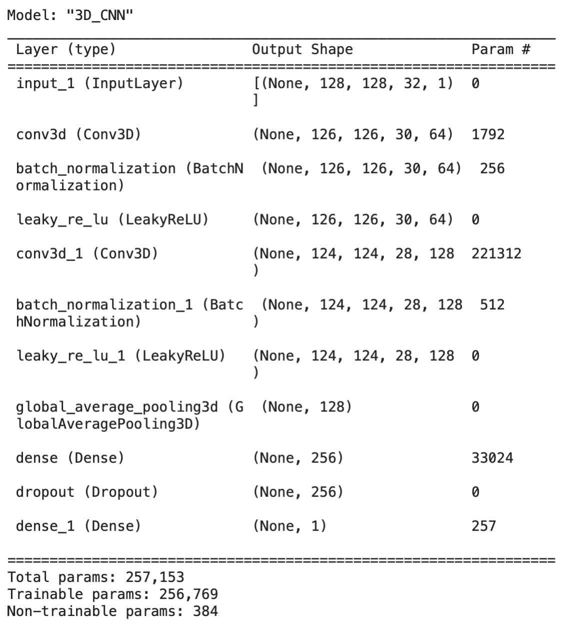


Fig. 6. Input $\rightarrow$ [Hidden Layer]$\times m$ $\rightarrow$ [Average Pooling] $\rightarrow$ [FC] $\rightarrow$ Output

Let's design the architecture for the model. The architecture basically follows the barebones of CNN that we defined earlier in *B*. Additionally, we add the following layers in the hidden layer:

- **Batch Normalization**: Normalize the output of previous `conv_layer` into 0 mean and unit variance. This helps the model to be more stable and converge faster.
- **Dropout**: Acts as a regularizer so that the model does not overfit.
- **Max Pooling**: Downsampling the feature maps to reduce the parameters to the 'sharpest' ones. This layer is optional.


```
Model: "3D_CNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 128, 128, 32, 1)  0
                             ]

 conv3d (Conv3D)             (None, 126, 126, 30, 64)  1792

 batch_normalization (BatchN  (None, 126, 126, 30, 64)  256
 ormalization)

 leaky_re_lu (LeakyReLU)     (None, 126, 126, 30, 64)  0

 conv3d_1 (Conv3D)           (None, 124, 124, 28, 128  221312
                             )

 batch_normalization_1 (Batc  (None, 124, 124, 28, 128  512
 hNormalization)             )

 leaky_re_lu_1 (LeakyReLU)   (None, 124, 124, 28, 128  0
                             )

 global_average_pooling3d (G  (None, 128)              0
 lobalAveragePooling3D)

 dense (Dense)               (None, 256)               33024

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 257,153
Trainable params: 256,769
Non-trainable params: 384
```

Fig. 7. Summary of the model with $m = 2$ hidden layers

And finally we use *Binary Entropy* or *Negative Log* loss function with metrics of AUC.

E. Training Loop

After initializing the model, we construct `keras.optimizers.schedules.ExponentialDecay` with initial learning rate of $1e-4$ that exponentially decays the learning rate to avoid overstepping the minimum. We specifically choose batch size of $m = 8$ and Adam optimizer. Additionally, we make sure to save the best model by monitoring `val_auc` and set early stopping if `val_auc` does not improve for certain number of epochs. We train for 50 epochs and record training and validation loss, accuracy and auc. Ultimately, the best model will the highest `val_auc` be used for predicting the test set.

III. Results

After training, the best model has an accuracy of 0.621 and an AUC of 0.604 when predicting on the validation data. Using this model, we made predictions on the test data. These predictions have an accuracy of 0.535 and an AUC of 0.525. Figure 8 illustrates the ROC curve from which the AUC metric is derived. Since the AUC of this ROC curve is only 0.525, this suggests that this model is barely able to distinguish between samples in either classification.

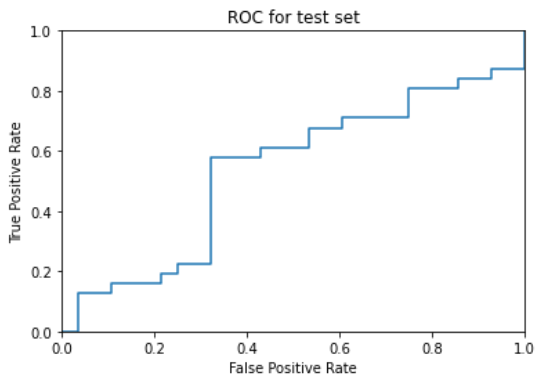


Fig. 8. Receiver operating characteristic curve for the test dataset

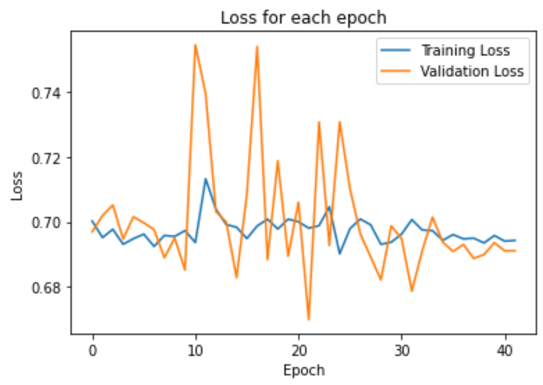


Fig. 9. Training and validation loss during training

In the graph in figure 9, loss for both the training and validation sets fluctuate around 0.70, indicating the model has found a local minimum, and the model is not getting closer to a better fit.

IV. Discussion

In summary, we got 0.54 for *accuracy* and 0.53 for *AUC score*. Eventually, this is not a state-of-the-art result! We are totally aware of this and believe it was due to several

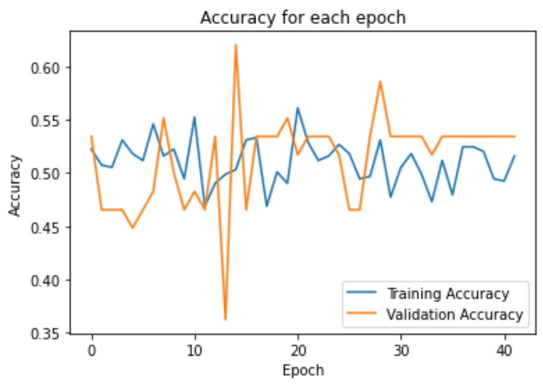


Fig. 10. Training and validation accuracy during training

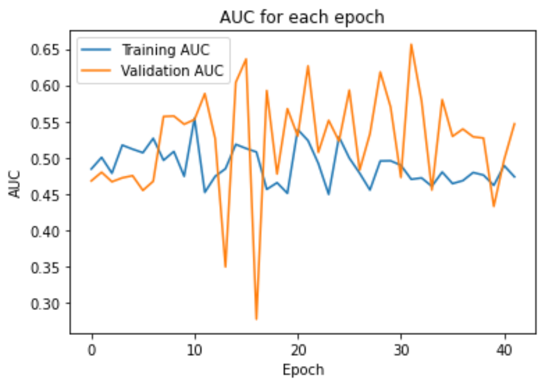


Fig. 11. Training and validation AUC during training

limitations and procedures that we were not able to complete in this project:

1) **Cross validation and Hyperparameters tuning**: Due to high training time where it takes several days to train the network with batch size of $m = 8$ in 50 epochs. However, we were able to experiment with various learning rates and optimizers such as Adam, RMSProp, etc in the first 5 epochs before deciding what are the final hyperparameters.
2) **Batch size:** Again, due to the computational limitations in our device and accessed service, we only use a small batch size of 8. Larger batch size could totally help the model converge to sharper and better minimizers.
3) **Model architecture:** Increase the number of `Conv3D` layers to increase the model's capacity and experiment different structure / combinations of layers could help improve the results.
4) **Heavier Data processing (augementation)**: This can be done in the next future iteration of the project due its independence on the computational power on any service or device.

V. Future Work

While processing the mpMRI scans, we notice that the scans were taken on multiple viewpoints of the brain. In fact, MRI can be visualized in 3 different planes: *Axial*, *Coronal* and *Sagittal*, shown in figure 12.

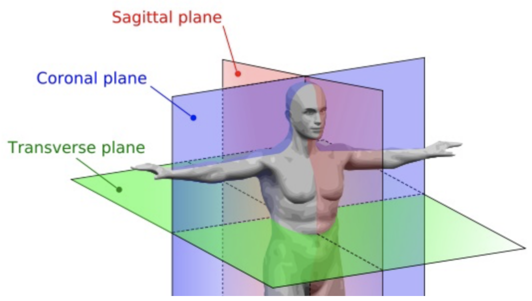


Fig. 12. MRI plane

There are some examples shown in figures 13, 14, and 15.

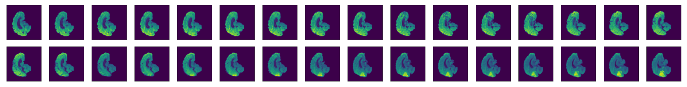


Fig. 13. MRI scans along Sagittal Plane

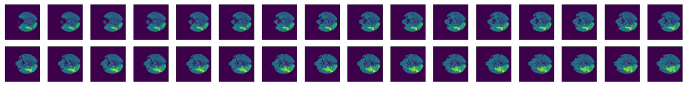


Fig. 14. MRI scans along Traverse Plane

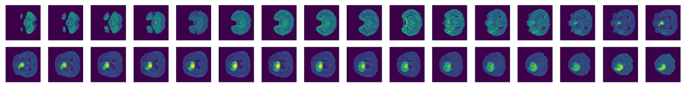


Fig. 15. MRI scans along Coronal plane

Our hypothesis is that while this might add some generalizations to the model by providing multiple views of the brain for the model to train on. Without heavy data augmentation, this could potentially affect the learning performance of the model. Therefore, we could apply more proper rotations for the MRI scans; however this is our hypothesis and not guarantee to get the model work better.

References

[1] Baid and et la. The RSNA-ASNR-MICCAI BraTS 2021 Benchmark on Brain Tumor Segmentation and Radiogenomic Classification. arXiv preprint arXiv:2107.02314, 2021.

[2] Satya P. Singh, Lipo Wang, Sukrit Gupta, Haveesh Goli, Parasuraman Padmanabhan, and Bala zs Gulya s. 3D Deep Learning on Medical Images: A Review. arXiv preprint arXiv:2004.00218, 2020.

[3] Weller, M., Stupp, R., Reifenberger, G., Brandes, A. A., van den Bent, M. J., Wick, W., Hegi, M. E. (2010). MGMT promoter methylation in malignant gliomas: ready for personalized medicine?. Nature reviews. Neurology, 6(1), 39–51. https://doi.org/10.1038/nrneurol.2009.197

[4] D. C. Preston, "Magnetic Resonance Imaging (MRI) of the Brain and Spine: Basics," MRInbsp; BASICS. [Online]. Available: https://case.edu/med/neurology/NR/MRI%20Basics.htm. [Accessed: 04-May-2022].