Capstone 3 Report - Predicting the Recurrence of Thyroid Cancer for Post-Treatment Patients
Alex Lai

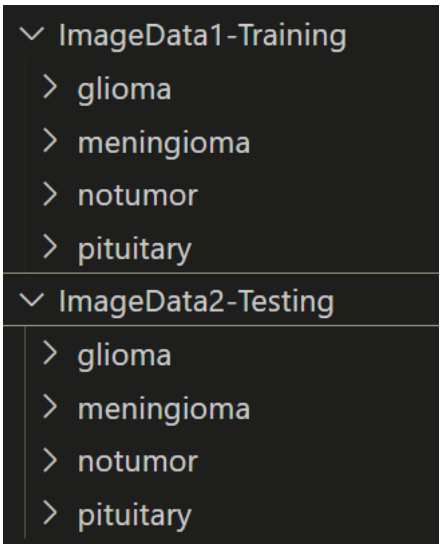
Roleplay: I am role-playing as a data scientist in a group presenting a new model to our hospital.

Problem:

- Our hospital has lots of unorganized brain MRIs and wants to be able to sort which ones have brain tumors or not. And some of these images have benign normal tumors that are not cancerous, but can only be differentiated from an expert. The amount of images is too much for doctors to take time out of their busy day and scan them all. Thus we need a solution that can automate this process while also having the skills of an expert.
- The solution is to make an image-processing brain tumor predictive model to automate on scale. It will be a CNN model done on pytrorch.
- We can make a prototype using open-source data on Kaggle.
- The stakeholders are the hospital and doctors.
- And we wish to have a validation loss under 0.15, no signs of overfitting, and an accuracy of 95%. This is just to make a quick prototype of a model working.

Data Description

- We are using this data from kaggle
    - https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset
- Which was provided by IEEE Data Port.
    - https://ieee-dataport.org/documents/brain-tumor-mri-dataset
- Exploring all the images and folder structure in File Explorer the image data are organized into this format (bottom right pic), where the main parent folder is split between training and testing, and there are the same 4 subfolders in both as either ‘glioma’, ‘meningioma’, ‘notumor’, ‘pituitary’. All the images together in all folders are 7023 images.
- Brain MRI Terminology:
    - Glioma (Cancer)
        - A type of cancer arising from glial cells in the brain or spinal cord.
    - Meningioma (Non-Cancer)
        - Usually a benign tumor from the meninges; rarely malignant.
    - Notumor (Non-Cancer)
        - This means no tumor was found.
    - Pituitary (Non-Cancer)
        - Refers to the gland; tumors (mostly benign) can form, rarely cancerous.

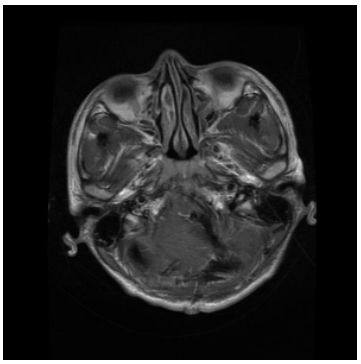


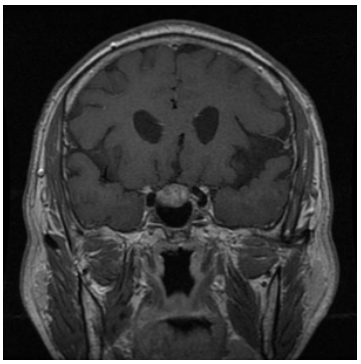
The Plan:

- The end goal is to make a model where you input image data of Brain MRI and will classify either Cancer or No Cancer.
- We will be experimenting with different data sets and different model methods.
    - Model 1 is to 4 classes (glioma, meningioma, notumor, pituitary), and have the model try to classify the 4 classes which will then be organized into 2 classifications (cancer or non-cancer).
        - The idea is that the model will place more effort into separating into 4 classifications (which already have balanced data) and thus be more accurate. And then we can add another code layer to organize the results of 4 classifications into 2 classifications of either cancer or no cancer.
    - Model 2 is to have the Glioma class as cancer and the rest of the classes merged as not cancer. Then train the AI to classify between the 2 classes. But this will create a data imbalance issue that we will have to address.

Sample of Image data

A sample of what the images look like. They are all grayscale MRI scans of a brain.

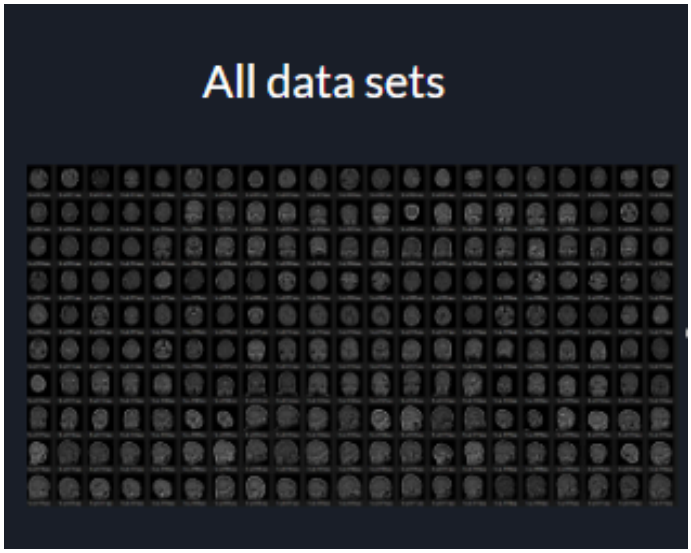


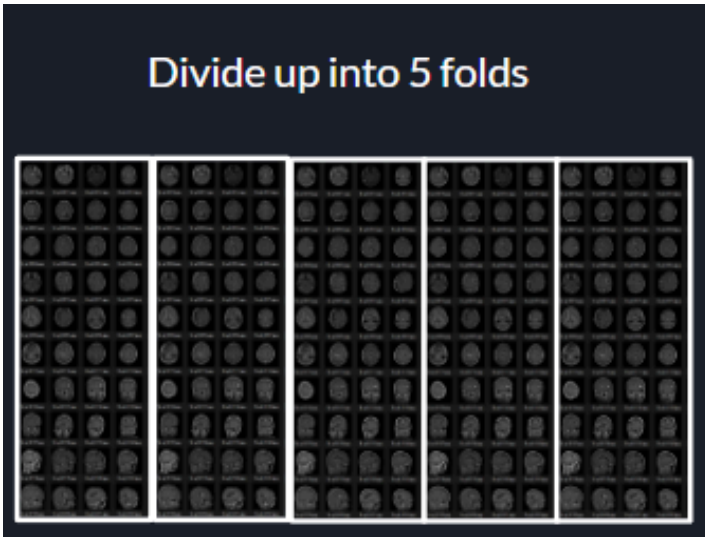


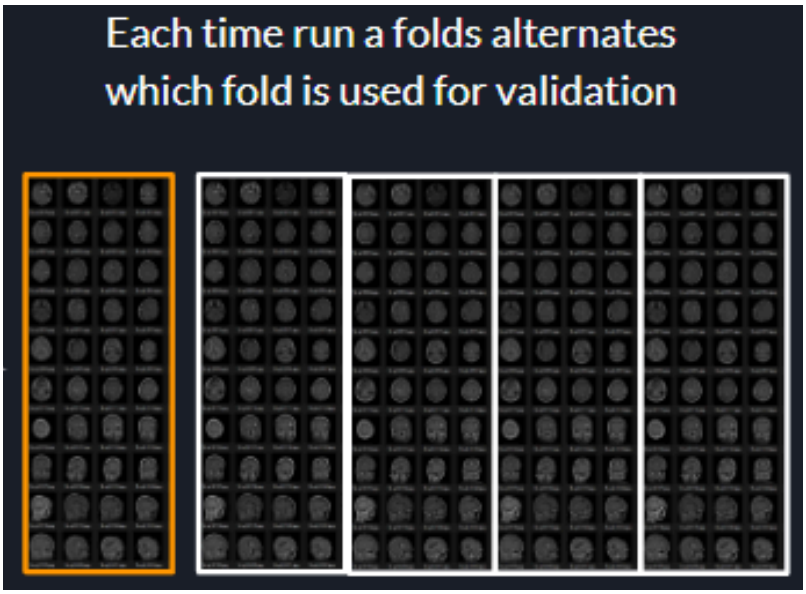
Data Splitting and Cross-validation

We will be testing multiple different models and data sets, thus will have to use cross-validation to decide which is the best. Since we will be using cross-validation, we will be splitting up the data with k-folds. However the current data is already split up into train and test pairs, thus we have to merge the data and then use code to split them up into folds.

We are using the k-fold method for cross-validation. 5 folds were chosen to align with the 80/20 training validation split. Basically we divide up the total data set into 5 folds and each time you're running a fold, one fold is selected to be used for validation and the other 4 folds for training.







Data Cleaning

We went to scan through all the files and found no corrupted images.

We scanned for duplicates and removed them. Out of the 7023 original images, 297 duplicates were found and removed with only 6726 images left.

Though all the images are grayscale they are still in the RBG format and thus we will need to convert all the images into grayscale 1 channel format, which we did in the pytorch transform class code.

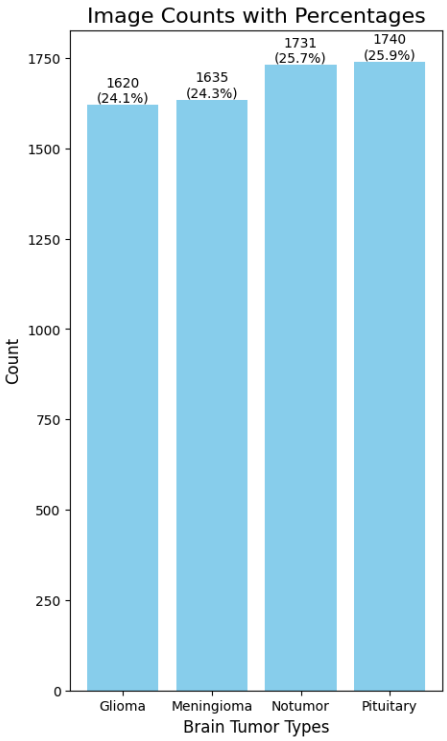
```
return transforms.Compose([
    transforms.Grayscale(num_output_channels=1),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.Resize(resize),
    transforms.ToTensor(),
])
```

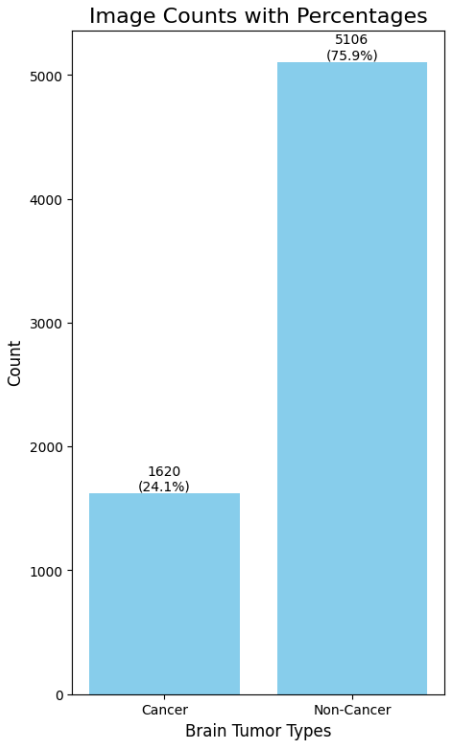
We scanned for metadata in all the images and the majority of them are in 512x512 dimensions, but alot of them have variable dimension sizes, here is their descriptive stat. We have 25 outliers but after inspecting all the outliers they are not a big issue as later we will rescale and resize them.

```
Width Statistics:
- Min width: 150
- Max width: 1920
- Mean width: 451.25
- Median width: 512.00
- Std deviation: 127.99
- Mode width: 512
```

```
Height Statistics:
- Min height: 168
- Max height: 1446
- Mean height: 453.12
- Median height: 512.00
- Std deviation: 122.38
- Mode height: 512

Total Outliers found: 25
```

We made 2 datasets, one is the native 4 classes and the other has Meningioma, Notumor, and Pituitary classes all merged into a single non-cancer class; thus 2 classes now. As you can see the merged non-cancer class is imbalanced now. To address this data imbalance we made a custom code during modeling to make the minority class weigh more during training.





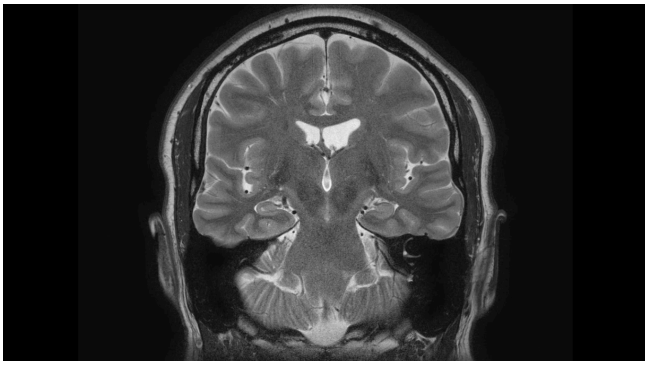

```
def calculate_class_weights(dataset):
```

Resize:

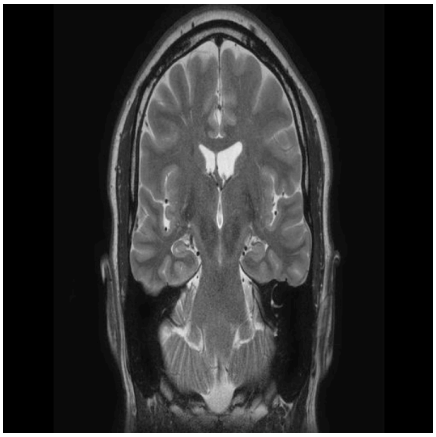
Since we will be using a CNN, that means it is preferred that all the images will have to be standard size. From our last info it appears the majority of the images are 512x512 so will resize it to those dimensions.

Our 1st attempt to resize the images was just to use the normal resize function, but that caused some images to stretch and will effect the CNN's filters on specific shapes.

Original:

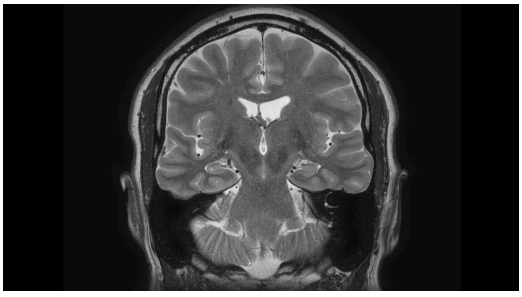


Resized (stretched/squashed):

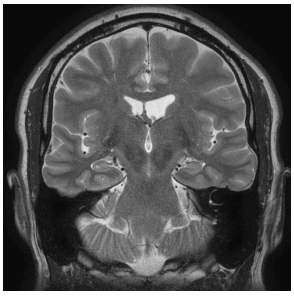


Our 2nd attempt was to crop the images first and then resize. We decided to crop the images as we are assuming that the wrong-resolution pictures will have the most important info (the tumors) in the middle of the picture, and all the things that I will be cropping out are not that important. We will crop them into a square based on the shortest dimension like height or width and then resize it when it's in this square format. The final results look good.

Original:



Crop then Resize:



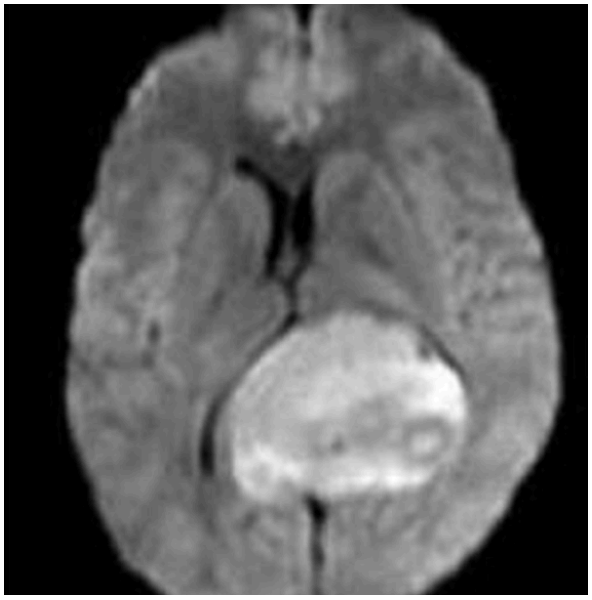
All of the images are now resized as 512x512.

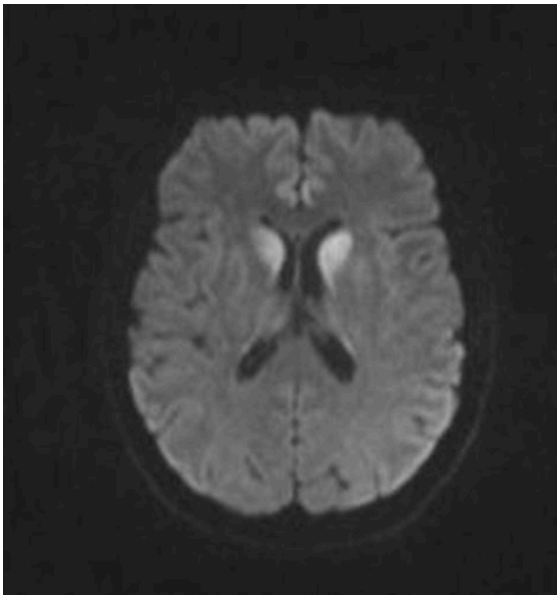
```
Dimension Counts:
- (512, 512): 6719 files

Width Statistics:
- Min width: 512
- Max width: 512
- Mean width: 512.00
- Median width: 512.00
- Std deviation: 0.00
- Mode width: 512
```

```
Height Statistics:
- Min height: 512
- Max height: 512
- Mean height: 512.00
- Median height: 512.00
- Std deviation: 0.00
- Mode height: 512

Total Outliers found: 0
```

Check for Blurry Images

After resizing, some small images scaled up may be blurry. I went to scan for blurry images using the Laplacian variance method. I set the threshold to 100 and all the images looked good. I kept setting the threshold lower and lower til 10. At 10 the 4 images are returned as blurry. I checked the images and yes they do seem too blurry for the AI, so I deleted those 4.





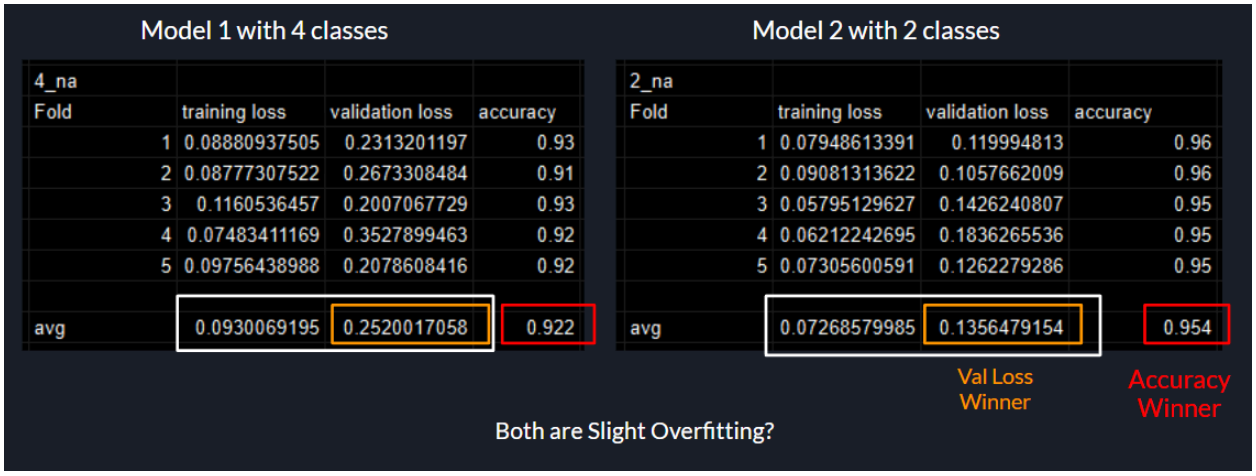
Data Augmentation

To make the most of our data, in the Python data transform class we used a feature that allows the data to be randomly flipped or rotated. This should make the model more robust in determining.

```
return transforms.Compose([
    transforms.Grayscale(num_output_channels=1),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.Resize(resize),
    transforms.ToTensor(),
])
```

Modeling

Here are the results:



4_na

Fold	training loss	validation loss	accuracy
1	0.08880937505	0.2313201197	0.93
2	0.08777307522	0.2673308484	0.91
3	0.1160536457	0.2007067729	0.93
4	0.07483411169	0.3527899463	0.92
5	0.09756438988	0.2078608416	0.92
avg	0.0930069195	0.2520017058	0.922

2_na

Fold	training loss	validation loss	accuracy
1	0.07948613391	0.119994813	0.96
2	0.09081313622	0.1057662009	0.96
3	0.05795129627	0.1426240807	0.95
4	0.06212242695	0.1836265536	0.95
5	0.07305600591	0.1262279286	0.95
avg	0.07268579985	0.1356479154	0.954

In orange, it appears model 2 average validation loss (and validation loss across each fold) is the lowest.

In red, it appears model 2 average accuracy (and accuracy across each fold) is the lowest.

In white, it appears both models have their training loss a bit lower than their validation loss which hints at possible overfitting.

Conclusion

Our main goal was to make an image-processing brain tumor predictive model to automate on scale. For that model to have a validation loss under 0.15, no signs of overfitting, and an accuracy of 95%.

We concluded to use Model 2 which was trained on the merged classes of cancer and non-cancer. It seems that merging the 4 classes into 2 classes is easier for the model. However we acknowledge that Model 2 may be overfitting a bit and do wish to improve the validation loss.

Possible Risks

Though the AI is good in the testing phase we have not tried it out on the field yet. There is always a risk the AI itself can mislabel a patient's brain MRI with false positive cancers or false negative cancers. In our backlog of images, the risk is not that bad, but if we eventually used this in real-time for patients after they took their MRI, mislabeling can open the hospital up to lawsuits.

Future Scope:

- We would like to research more on why merging into 2 classes is better than 4.
- We would like to experiment with augmenting the data (rot and flip). Upon experimenting, with this method we can increase the amount of data sets from 6K to 87K.

    - Of course, we tried that method and it took 2 hours for one fold thus will need to figure out a more optimized solution to expedite the process.
    - One method is to optimize parallel computing to process more data faster.
- Experiment with different types of CNN model structures than only the simple one.