

# Analyzing methods of Detection of Brain Tumors from an MRI scan using Machine Learning Techniques

By, Kousthubh Veturi

Advisor/Co Author: Dr.Mason McGill

# I. Abstract

The goal is to test and select the best performing solution to detect Brain Tumors and Types of Brain Tumors(Glioma, Pituitary, and Meningioma) from an MRI scan using Machine Learning Techniques. A well functioning model would intend to replace manual work in a physical medical environment, where trained medical professionals are not just expensive, but prone to human error, fatigue, and other unpredictable issues which could affect accuracy of non-computer systems in an environment where there is simply no room for error. Using the programming language Python, and libraries such as TensorFlow, Scikit-Learn, Keras, and Numpy, we used the VGG (Very Deep Convolutional Networks)<sup>[1]</sup> and various different classifiers such as KNearestNeighbors Classifiers, Logistic Regression, and Decision Trees classifiers in scikitlearn. This project will utilize the relative modular form of the VGG network to work with specific layer combinations—models with layers shaved off until different points— in order to evaluate the best functioning model. Classifiers with different parameters (Different numbers of neighbors for Knearest Neighbors, different numbers of depths for Decision trees) will be analyzed with three set layer cuts from the VGG, the *block5\_conv3* layer, the *fc2* layer, and the *flatten* layer.

P A R A M E T E R S	KNearestNeighbors	Decision Trees	Logistic Regression
	Numbers of Neighbors ranging from 1 to 8	Number of Depth Ranging from 1 to 8	No Parameters

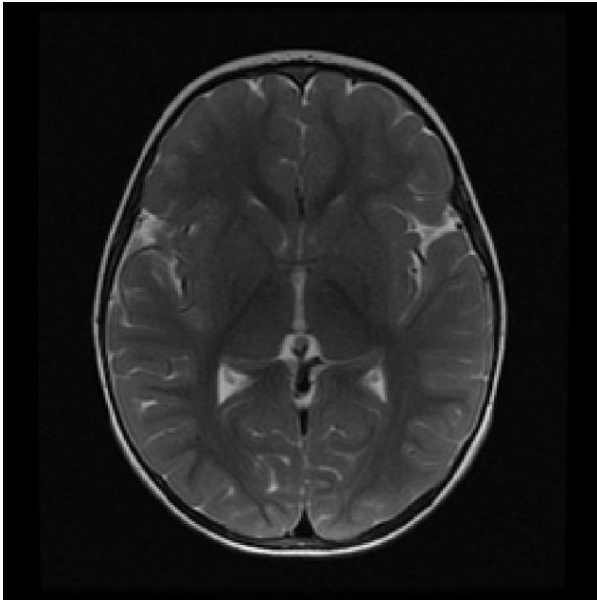
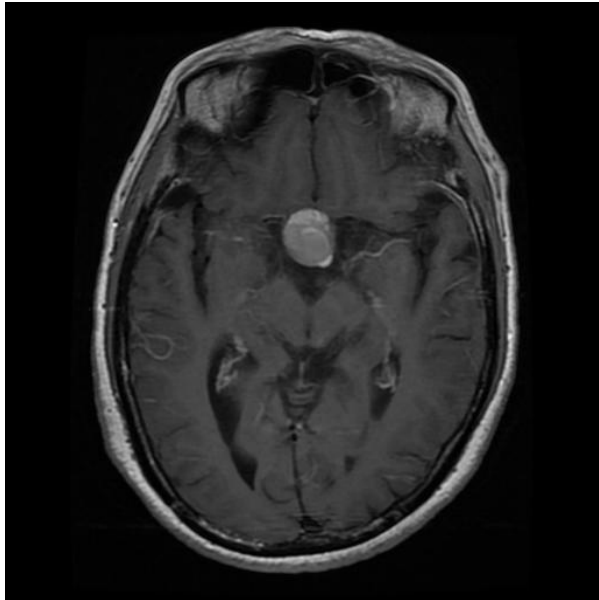
V G G  D E T A I L S	FC2	FLATTEN	block_5_conv3
	Full 16 Layer model - Last 2 Layers	Full 16 Layer model - Last Layer	Full 16 Layer model - Last 5 layers

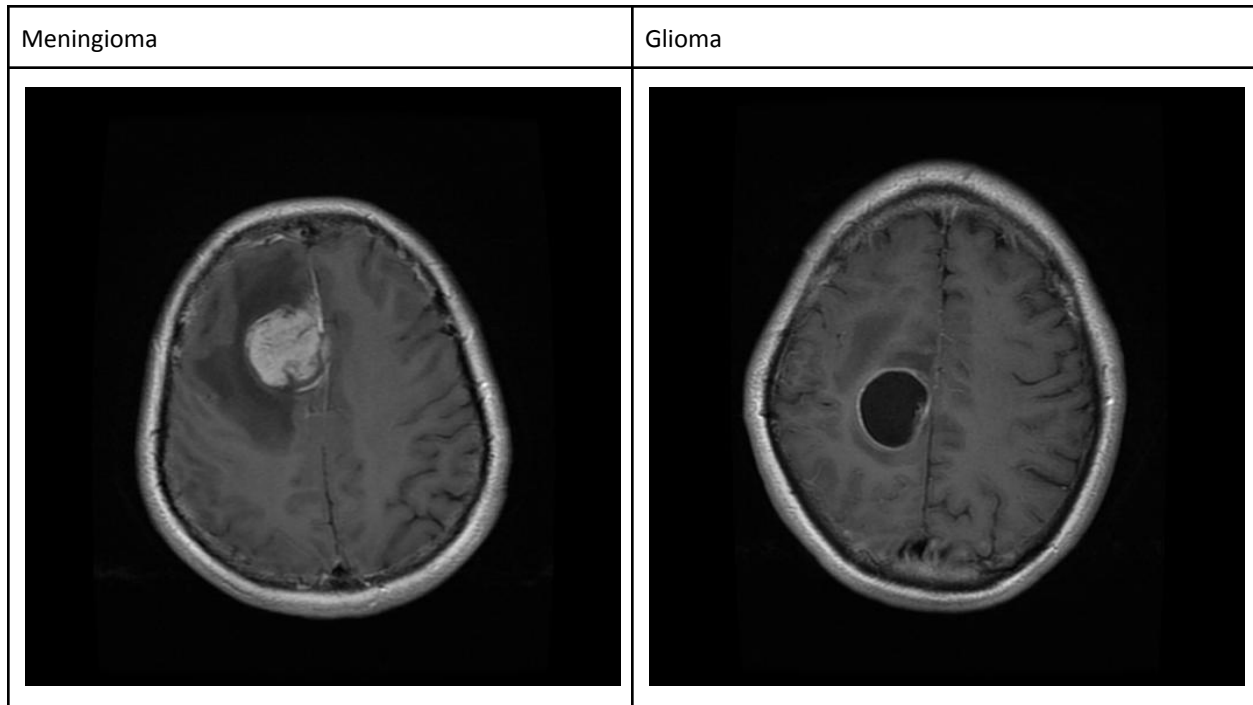
Through our experiments, we were able to evaluate the best model to have an accuracy of 98.39%.

## II. Experiment Details

### The Dataset

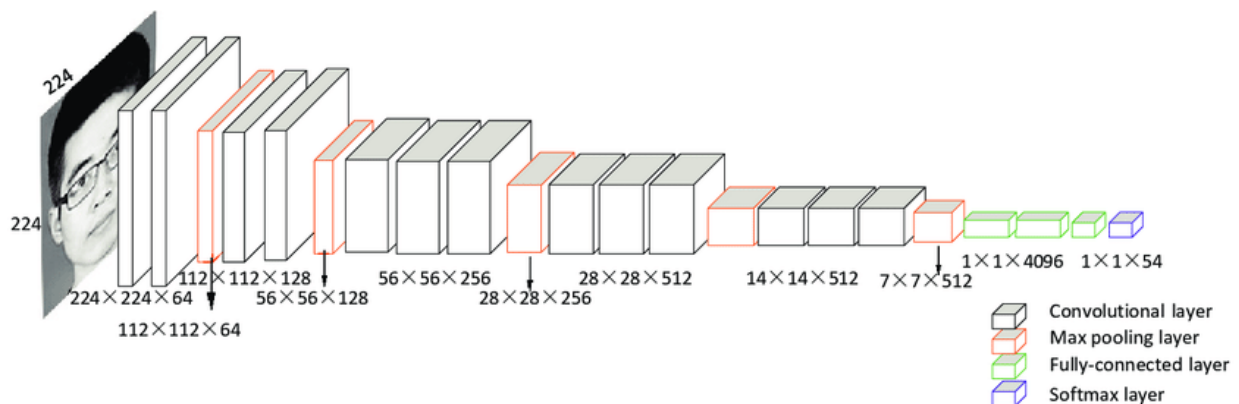
This project utilizes the “Brain Tumor MRI Dataset”<sup>[2]</sup> by Masoud Nickparvar, sourced from Kaggle, to provide the images for classification. This dataset contains a total of 7,022 images which are already split into training and testing sets of 5712 and 1311 images respectively. These images seem to be black-and-white scale at first glance, with there not being that much variation in color, resembling a greyscale scan, which is what an MRI is intended to look like. The MRI scans are further split into four classes, “notumor” - no presence of a tumor, “pitutary” - a tumor detected in the pitutary gland of the brain, “meningioma” - tumor detected in the surrounding membrane *meninges* of the brain, and “glioma” - tumor in the glial cells of the brain. These are a sample of two images from each class, taken from the training set (on next page)

No Tumor	Pitutary Tumors
	



## The VGG model

This project utilized the VGG model for its relative simplicity as a Sequential model, meaning that the model continues in a fixed order every time without any backtracks or loops. For this project, we opted to go with the VGG-16 model variant pretrained in Keras, which had 16 layers when unmodified:

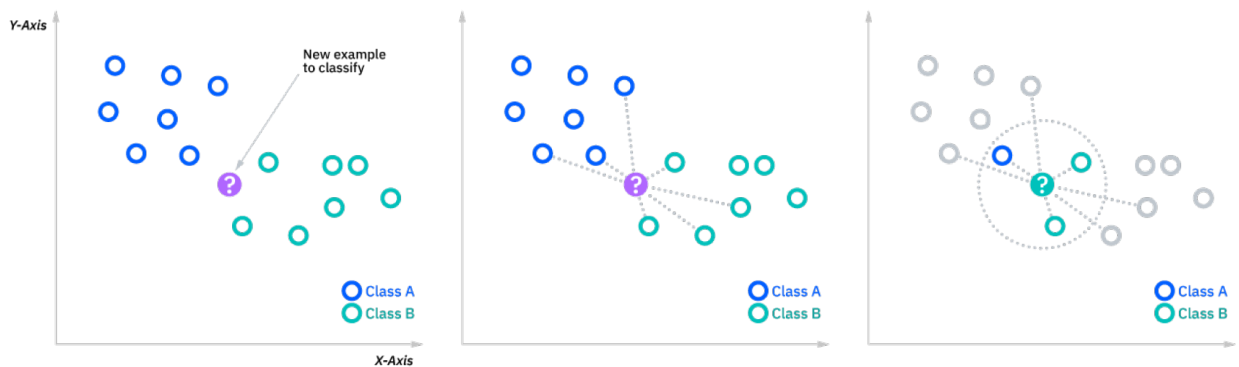


[3]

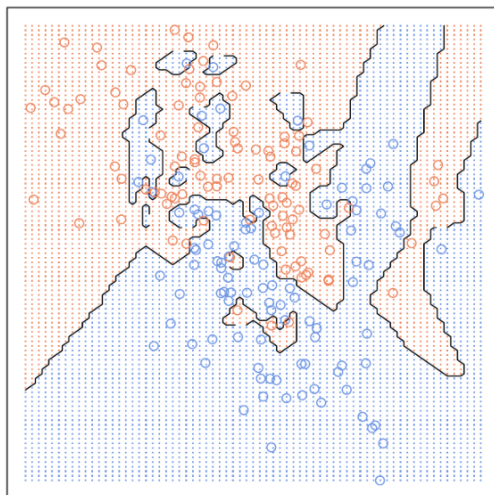
The usage of the VGG model was warranted in order to be able to generate the feature values that would then further be used to train and test classifiers of different types. The output feature array from the model was based on the layer that the features were extracted from.

## The KNearestNeighbors Classifier

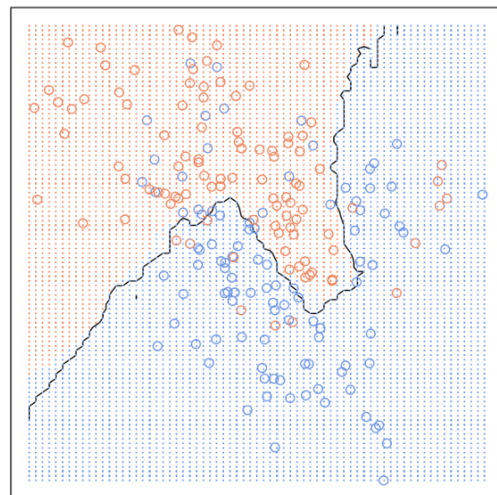
KNearestNeighbors, also known as kNN alias KNN, is a supervised learning classifier which utilises proximity and distance to make a prediction about a specific data point, hence the name “nearest neighbors”. In the context of this task, there are four different classes, hence, there would be four different clusters on a plane. In this project, we set the number of neighbors as a value from 1 to 8 inclusive. The more number of neighbors there are, the more flexible the classifier is to outliers. Attached is a diagram of how a KNearestNeighbors classifier works<sup>[4]</sup>, and below the first image is a diagram displaying the effect of more neighbors<sup>[5]</sup>



**1-nearest neighbours**



**20-nearest neighbours**



Initially, values for distances from the input point to the other data points are calculated with Euclidean Distance, which returns a distance vector, calculated as follows:

$$Distance_{euclidean}(x, y) = \sqrt{\sum_{i=1}^n (y_i - x_i)^2}$$

In the equation,  $y_0$  and  $x_0$  are the input euclidean vectors that start from the origin, or the initial point. The variable  $n$  refers to the  $n$ -space, or the number of dimensions being analyzed and worked with. This euclidean distance is used to implement the idea of “plurality vote”, that is, if the majority of close points have a specific label, then that point is given said label. If a point has a result of 35% class A, 25% class b, 20% class C, 15% class D, and 5% class E, the point is assigned class A because the *plurality* of close points are of class A.

Ultimately, advantages to this classifier include the relative simplicity of the construction of the classifier and the simplicity of the mathematics that make this classifier work.

## Logistic Regression Classifier

Logistic Regression guesses and estimates the probability of a class based on the dataset given. Logistic Regression involves the application of a “logit transformation” on the probabilities. Since the project’s task involves multiple classes, Multinomial Logistic Regression is utilized. Multinomial Logistic Regression uses softmax instead of sigmoid for the loss function. The softmax and sigmoid functions are as follows:

$$softmax(x_0) = \frac{e^{x_0}}{\sum_{i=1}^n e^{x_i}} \quad sigmoid(x) = \frac{1}{1+e^{-x}}$$

$$\widehat{Multinomial Probability} = softmax(x_0)$$

For this project, the base Logistic Regression classifier in Scikit-learn is utilized for classification without any modification of parameters. The default parameters include defaulting to  $l2$  penalties and *multinomial* as there are four classes being worked with.

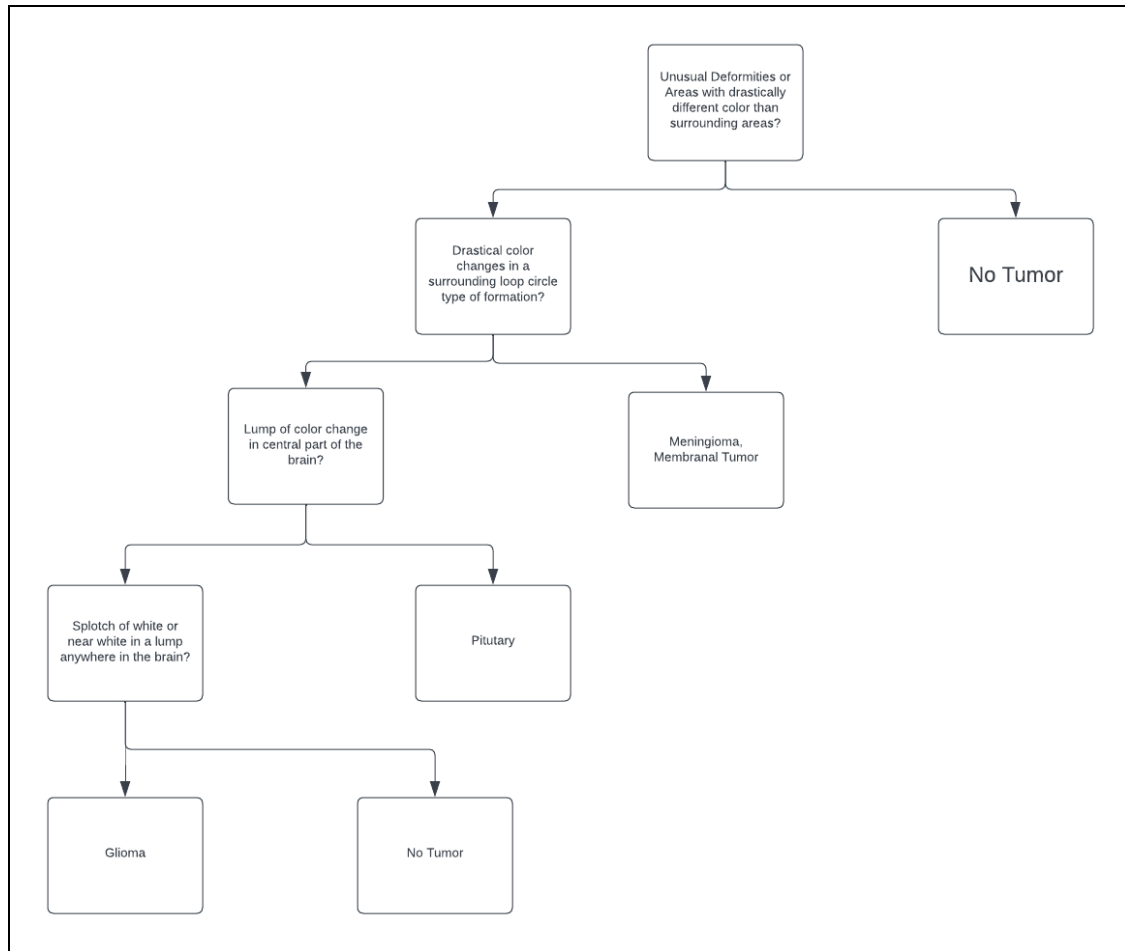
## Decision Trees Classifier

Decision Tree Classifiers are unsupervised classifiers that work based on a tree like structure. There is a Primary node, from which internal nodes break off , which further break into leaf nodes. This classification task would involve four classes, so the leaf nodes would always be either no tumor detected, or the three types of brain tumors. In order to find splits into various nodes, Decision Tree classifiers employ the usage of greedy searches. The decision for what to split into nodes on is called an attribute and is calculated using the Gini Impurity. This is the probability of incorrectly classifying data points, and can be calculated as

$$Gini\ Impurity = 1 - \sum_{i=1}^n (p_i^2)$$

, where  $p_i$  denotes the probability of each class, and  $n$  denotes the number of classes.

Furthermore, an important aspect of decision trees is the concept of depth. Each split into further nodes is one level of depth. For example, splitting a question into two answers is one level of depth. Depth is the distance of the longest path from a leaf to a primary node in a decision tree. The more depth there is, the longer the distance from the primary node to the farthest leaf node possible, highlighting that there is increasing complexity and on paper, more accuracy for our classification task. Below is a simple potential representation of a decision tree classifier(on next page).



As visible, the more the depth, the more number of combinations of nodes that can result in selection and prediction of classes. The less the depth, the more limited the options are, and thus, the lower accuracy.

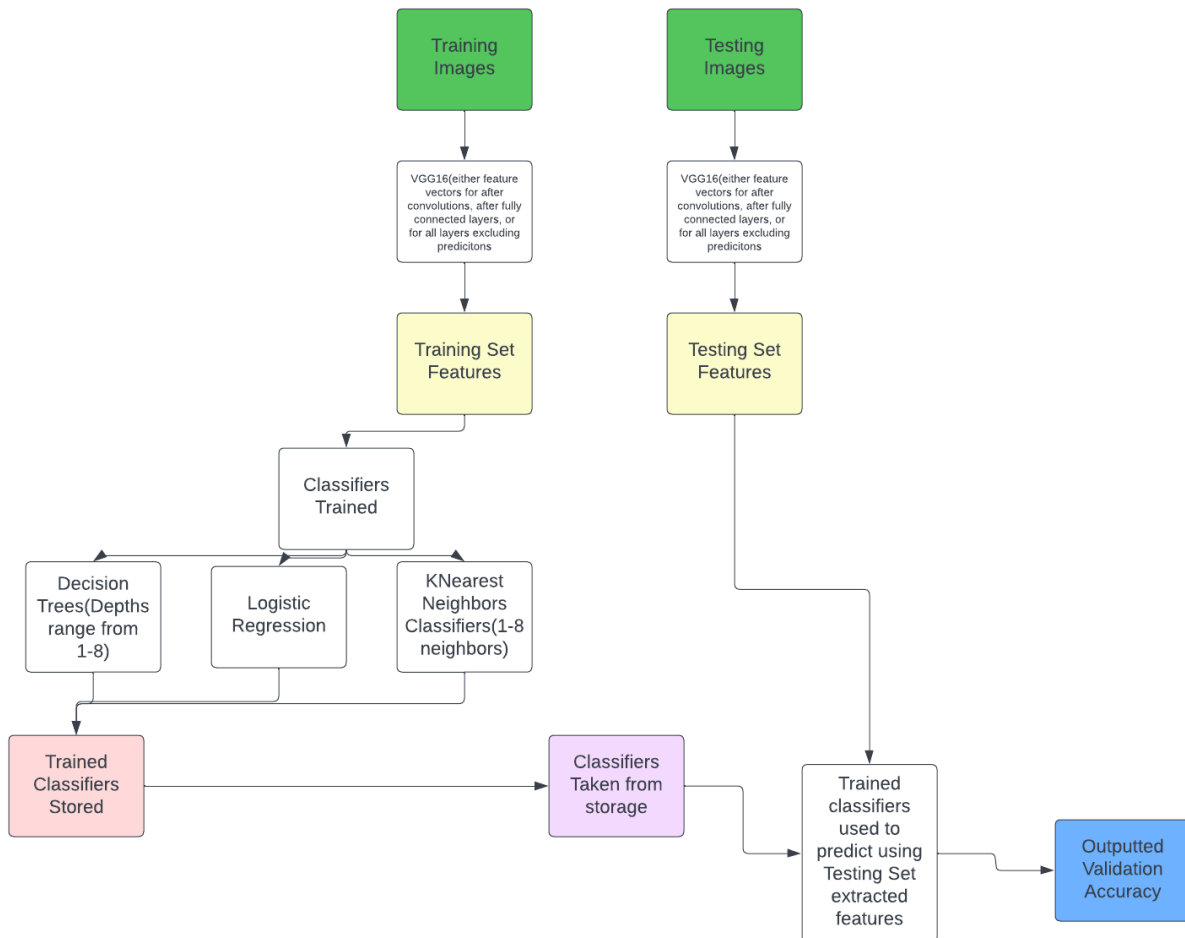
## Extraction of Features from VGG-16

For the project, we opted to extract feature vectors from the *block\_5\_conv3* layer, *fc2* layer, and the *flatten* layer. None of these include the prediction layer as the predictions for the 1000 classes of ImageNet aren't of use for this classification task. The reasons for the selection of said layers for which to shave off the successive layers is as following. Shaving off after the *block\_5\_conv3* layer gives features for after data has gone through only the convolution layers, *fc2* gives the vectors for data that has gone through everything except for the flattening and prediction layers, and *flatten* gives us the feature values that haven't been converted into 1000



class predictions. In summary, these three layers are critical parts of the VGG-16 model, working as the last layers of each “block” of the VGG-16 model.

Diagram of Full Classification task:



### III. Results of Experiments

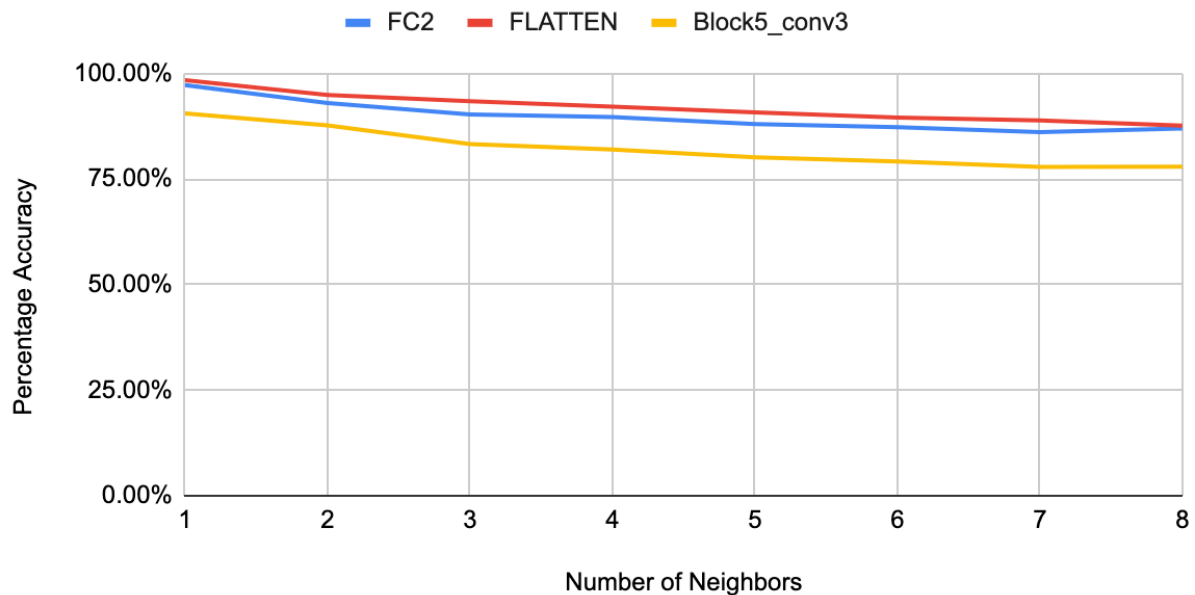
#### K Nearest Neighbors Results

The K-Nearest Neighbors classifiers often performed best with the numbers of neighbors being set to lower values, with accuracies decreasing as numbers of neighbors went up. This is understandable as earlier discussed, the increase in the number of neighbors results in the increased flexibility of the inclusion of outliers, hence the lower accuracy.

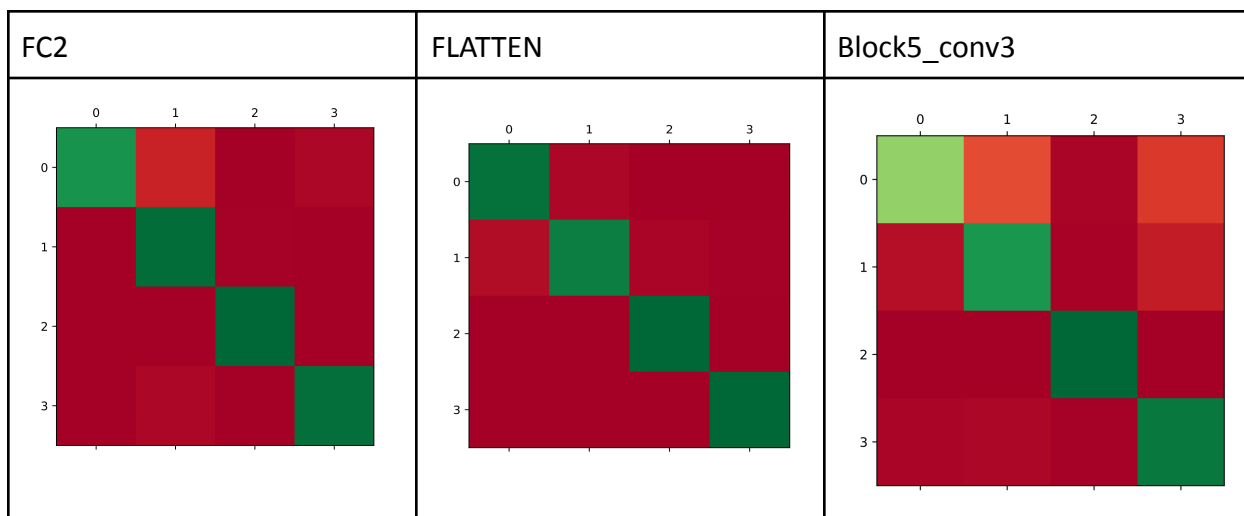
ACCURACIES (In Percentages to two places after the decimal point)

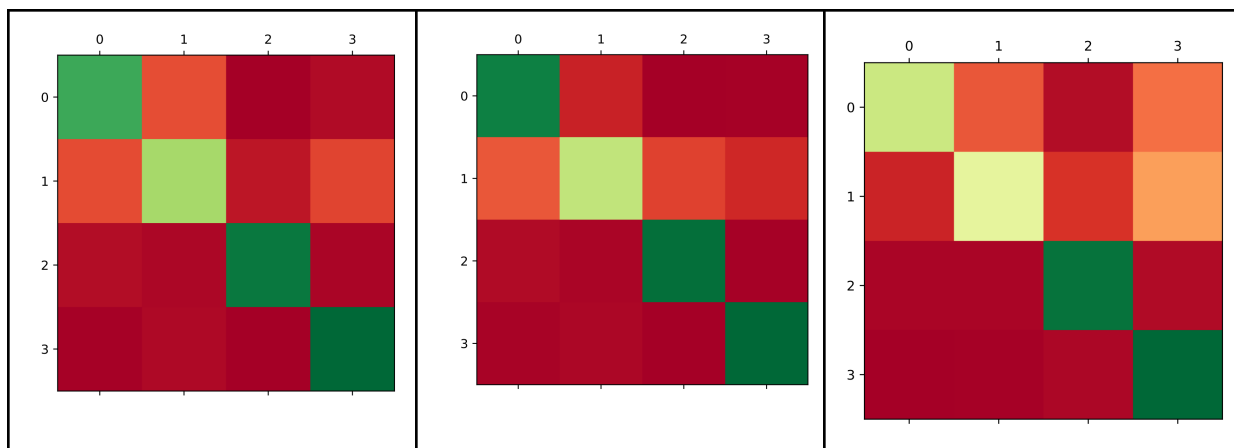
# Neighbors	FC2	FLATTEN	Block5_conv3
1	97.25%	<b>98.39%</b>	90.54%
2	92.98%	94.89%	87.71%
3	90.31%	93.44%	83.29%
4	89.70%	92.14%	81.99%
5	88.02%	90.77%	80.16%
6	87.26%	89.54%	79.17%
7	86.11%	88.86%	77.87%
8	86.95%	87.64%	77.95%

## Accuracies of the KNearestNeighbors Classifiers for different Layer Features



We can see that the highest accuracies still do remain at the classifiers with only one neighbor, with the Highest Accuracy belonging to the KNearestNeighbor classifier that had one neighbor, trained with the features extracted from the flattened layer. Across all tested features, we see that the highest accuracies are always in the one neighbor form of the classifier, which may be consistent with the more rigidity on the flexibility of inclusion of outliers. To further analyze the results, we use confusion matrices. Below are confusion matrices for the three best, and the three worst classifiers:



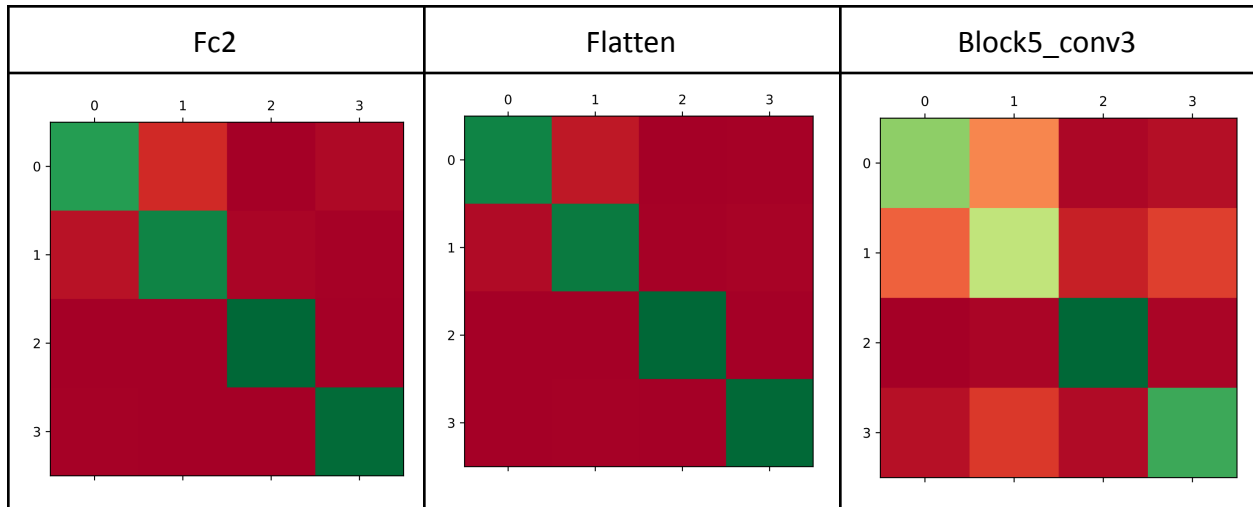


## Logistic Regression Results

Since only the base Multinomial Logistic Regression classifiers were tested, there was only one tested classifier per layer from which the features were extracted and tested with

Layer	Accuracy
fc2	95.80%
flatten	97.48%
block5_conv3	78.94%

It is clear that the classifier tested with the flatten layer's features performed the best, with the block5\_conv3's features testing the worst out of the tested features extracted. This may be because the earlier extractions don't have the features as developed as the flatten layer does (flatten layer is closest to the end of the model, block5\_conv3 is the farthest). These are the Confusion Matrices (on next page):



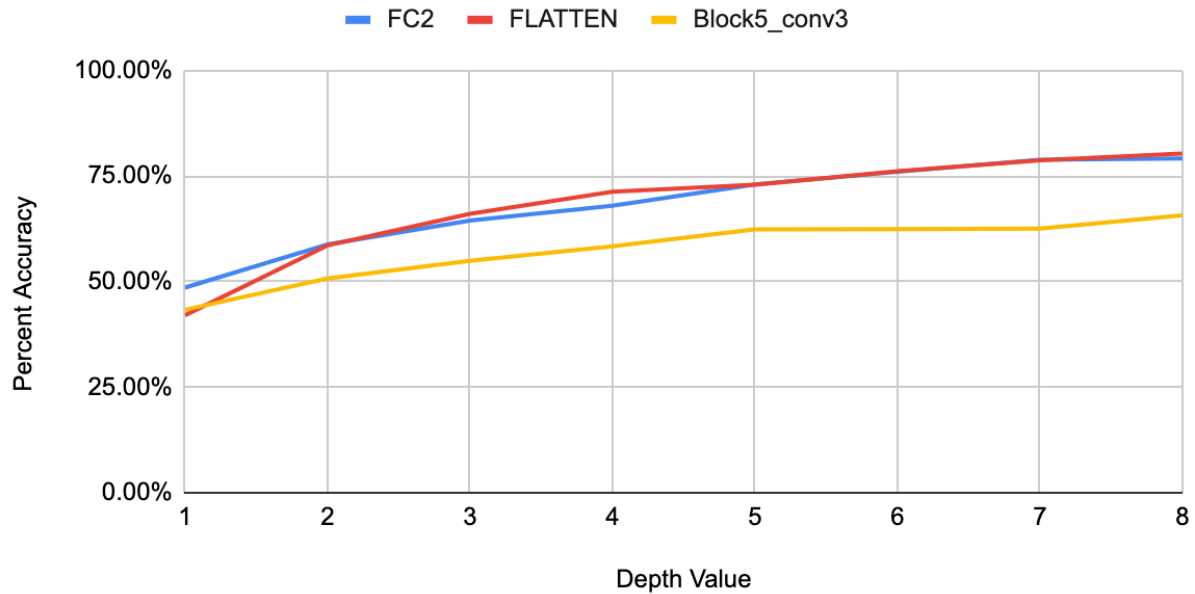
## Decision Tree Results

The Decision Tree classifiers were tested with depths ranging from 1 to 8. The results are as follows:

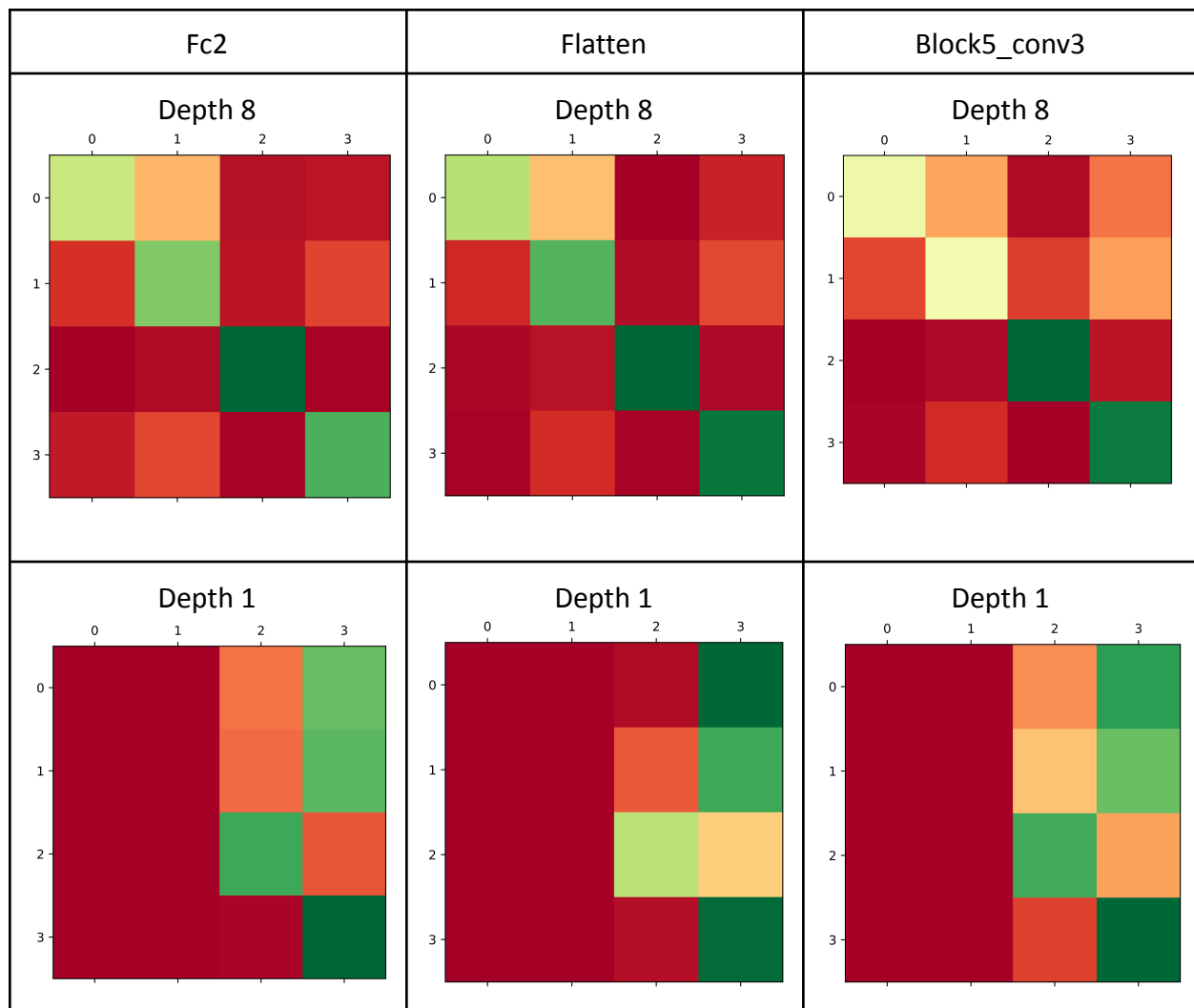
ACCURACIES (In Percentages to two places after the decimal point)

# Depth	FC2	FLATTEN	Block5_conv3
1	48.58%	42.02%	43.32%
2	58.81%	58.58%	50.72%
3	64.45%	66.06%	54.91%
4	67.96%	71.31%	58.35%
5	72.99%	72.92%	62.39%
6	76.04%	76.12%	62.40%
7	78.87%	78.71%	62.54%
8	79.17%	80.32%	65.67%

## Accuracies of the Decision Tree Classifiers for different Layer Features



Out of all the different classifiers experimented with, the Decision Tree Classifiers performed the worst. This may be because the depths of the classifiers was not enough to provide for optimal classification. Furthermore, the rigidity of the Decision Tree structure may have contributed to the lower accuracies. Below are confusion matrices for the three best ,and three worst classifiers(next page)



## Analysis of Confusion Matrices

In general the less neighbor classifiers of KNN, Logistic Regression with flatten layer features, and more depth classifiers of Decision Trees performed the best. In the K Nearest Neighbor Classifiers, the best results had a straight diagonal green trend, which is what we want, with the remaining squares being dark red, which is also what we want. However, in the worst performing KNN classifiers, it is noticeable that the areas where the accuracy decreases was the detection of cancer when there was no actual cancer detected, and the prediction of no cancer when there was actually cancer. In logistic regression, the classifier performed well on all feature layer extractions except for the one using block5\_conv3 features. This one had a greenish line through the center, a mediocre sign as the earlier parts of the diagonal are a

lighter green, indicating lower frequencies, but the frequencies outside of the diagonal tended to be in the detection of cancer when there was no cancer, and the detections of classes 0 and 1 were mixed up, with 1 detected at 0, and vice versa. However, the worst performance is visible in Decision Tree Classifiers, where the worst classifiers simply do not classify any of class 0 or 1 properly, with the accuracies being extremely low, sub 50% in these classes. Even the best classifiers suffer from this issue, with there being a little more improvement, with the diagonal cross being slightly yellowish and green, but the issue still remains in the misclassification of tumors versus non tumors. The reason for this is most likely the lack of depth in the classifier, with larger depths more likely doing better than less depth. However, this would make the classifier larger and slower, which is not ideal.

## IV. Conclusion

From the experiment we were able to deduce that the K-Nearest-Neighbors Classifier with one neighbor, along with features extracted from the flatten layer of the VGG-16 model resulted in the highest accuracies, and lowest critical errors (no tumor when there was tumor) out of all models, with an accuracy of **98.39%**. This is much higher than we expected, and would be the best option in a medical environment. The reasons that this specific classifier combination works best would have been the better developed features from the VGG-16 flattening layer, –one step away from predictions, so at the near end–, and the lower flexibility with the inclusion of outlier in the single neighbor option. The Decision Trees classifiers performed the worst overall and this is solely because of the usage of multiple classes of prediction and the need for more depth in order to maximize accuracy, which is more time consuming and therefore would not be a preferable option. Logistic Regression fared well but proved to be completely dependent on the layer from which features were extracted, with the later layers doing better for extraction than the earlier ones. Overall, the flattening layers worked the best for this project, and the K-Nearest Neighbors classifiers performed the best, with less neighbors being better. Overall, Logistic Regression with flattening features, K-Nearest neighbors with one neighbor and utilisation of flattening features and fc2 features were the best functioning in this project, with respective accuracies of 97.48%, 98.39%, and 97.25%.

I would like to thank Dr.McGill for being a phenomenal advisor on this project and being there to check on the work being done and helping make this project run smoothly and well executed.



## V. References

[1] Simoyan, Karen, and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." <https://arxiv.org/abs/1409.1556>.

[2] Nickparvar, Masoud "Brain Tumor MRI Dataset",  
<https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset>

[3] Image Borrowed

Link:

[https://www.researchgate.net/figure/An-illustration-of-the-architecture-of-the-VGG-16-model-The-training-dataset-is-used-to\\_fig1\\_336061375](https://www.researchgate.net/figure/An-illustration-of-the-architecture-of-the-VGG-16-model-The-training-dataset-is-used-to_fig1_336061375)

Origin of Diagram from link:

Pei, Zhao & Xu, Hang & Zhang, Yanning & Guo, Min & Yang, Yee-Hong. (2019). Face Recognition via Deep Learning Using Data Augmentation Based on Orthogonal Experiments. Electronics. 8. 1088. 10.3390/electronics8101088.

[4] Diagram Borrowed from following link:

IBM "K-Nearest Neighbors Algorithm"

<https://www.ibm.com/topics/knn>

[5] Image Borrowed

i-king-of-ml, "KNN(K-Nearest Neighbour) algorithm, maths behind it and how to find the best value for K",

<https://medium.com/@rdhawan201455/knn-k-nearest-neighbour-algorithm-maths-behind-it-and-how-to-find-the-best-value-for-k-6ff5b0955e3d>

Dr.McGill was the primary advisor for this project. All work in this paper is from experiments of code that we ran for this project, our findings and our own work, from knowledge of Machine Learning concepts that we know, and the sporadic usage of the sources listed above. All Generated figures that were not marked with a reference marker to the sources above were ones we generated.

The usage of "we" in this paper is to denote people who participated in this research project, all of who are authored and credited on the first page.

All questions can be emailed to [kousthubh.veturi@gmail.com](mailto:kousthubh.veturi@gmail.com)