

# Multi-Class Brain Tumor Classification using Hybrid Models

**Sumeet Sachdev**

Khoury College of Computer Sciences  
Northeastern University  
Boston, MA, 02120  
sachdev.su@northeastern.edu

**Jivesh Poddar**

Khoury College of Computer Sciences  
Northeastern University  
Boston, MA, 02120  
poddar.j@northeastern.edu

## Abstract

Brain is the most vital organ in the human body. A brain tumor is a mass or growth of abnormal cells in the brain. Traditionally, the tumor is identified manually by medical professionals by looking at MRI images and CT scans. A huge amount of image data is generated through the scans. These images are examined by the radiologist. A manual examination can be error-prone due to the level of complexities involved in brain tumors. The fatality of the tumor demands the urgency for detection and classification of the tumors. Proper treatment, planning, and accurate diagnostics should be implemented to improve the life expectancy of the patients. Research in Image processing, Machine Learning, and Deep Learning have improved a lot over the years and have found their applications across all domains. In the past, many ML algorithms have been trained on this problem and have achieved decent accuracy but there is little to no pre-processing involved. The methods previously used did not consider the Rician noise generated by MRI. The methods discussed here describe the ways to overcome this issue and obtain better results. At the end, we compare the evaluation metrics of ML models such as Naïve Bayes, K-Nearest Neighbor, Logistic Regression, Random Forest, Support Vector Machine with the manually trained CNN and pre-trained CNNs such as InceptionResNetV2, DenseNet201, InceptionV3, EfficientNetB2, and VGG19. Hence, proposing a system performing efficient detection and classification by using various Machine learning and Deep Learning Algorithms would be helpful to doctors all around the world.

## 1. Introduction

Cancer is the second leading cause of death globally, according to the World Health Organization (WHO) [1]. Early detection of cancer can prevent death, but this is not always possible. Unlike cancer, a tumor could be benign, pre-carcinoma, or malign. Benign tumors differ from malign in that benign generally do not spread to other organs and tissues and can be surgically removed. Some of the primary brain tumors are gliomas, meningiomas, and

pituitary tumors. Gliomas are a general term for tumors that arise from brain tissues other than nerve cells and blood vessels. On the other hand, meningiomas arise from the membranes that cover the brain and surround the central nervous system, whereas pituitary tumors are lumps that sit inside the skull. The most important difference between these three types of tumors is that meningiomas are typically benign, and gliomas are most commonly malignant. Pituitary tumors, even if benign, can cause other medical damage, unlike meningiomas, which are slow-growing tumors.

In this project, we attempted at detecting and classifying the brain tumor and compare the results of this classification of brain tumors using various machine learning architectures such as Naïve Bayes, K-Nearest Neighbor, Logistic Regression, Random Forest, Support Vector Machine with the manually trained CNN and pre-trained CNNs such as InceptionResNetV2, DenseNet201, InceptionV3, EfficientNetB2, and VGG19. In the literature, there are other algorithms and different modifications of the pre-trained networks that are used for image analysis, classification, and segmentation. Different approaches have been tested on other medical databases, both on MRI images of brain tumors and on tumors from different parts of the human body.

The most common method of image pre-processing often involves image augmentation <sup>[2]</sup>. Image augmentation is the method of increasing the dataset by applying one or more transformations on the images (such as zooming, flipping over the axes, rotation. Most of the methods above use Otsu's method to separate background and use wavelet transform <sup>[3]</sup>. After processing in similar way, many researchers have tried to solve the problem using simple ML models – KNN <sup>[4]</sup>, Logistic Regression <sup>[5]</sup>, SVM <sup>[6]</sup>, and Random Forests <sup>[7]</sup>. The noise in MR images is due to the fluctuations of the magnetic field in the coil. Magnetic Resonance Imaging is corrupted by Rician noise. Rician noise makes quantitative measurements difficult. Therefore, obtaining high-quality denoising images is the most important task of pre-processing. PCA is also used for feature selection, so that the feature space gets reduced.

## 2. Preliminaries

The dataset is taken from Kaggle <sup>[13]</sup>. The dataset consists of 3264 files divided into Training and Testing folders. Each folder further contains 4 sub-folders divided by the type of tumors. There are 4 sub-folders: glioma\_tumor, meningioma\_tumor, no\_tumor, and pituitary\_tumor. The distribution is demonstrated below in the tree (Figure 1). The number of each image in the training and testing set is shown by bar plot in Figure 2 and Figure 3.

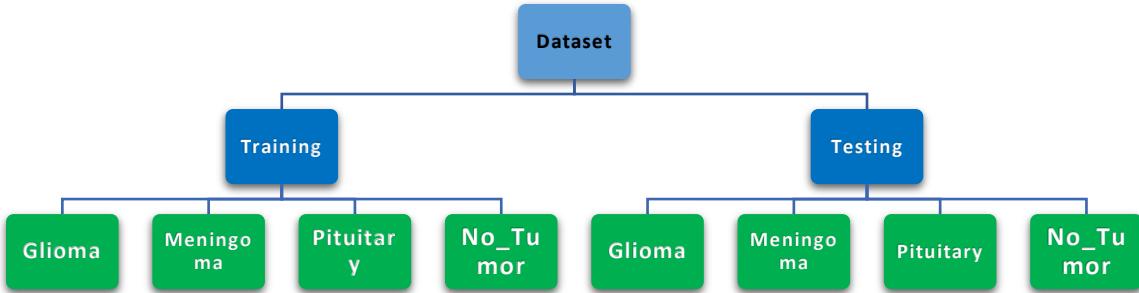


Figure 1: Structure of Dataset

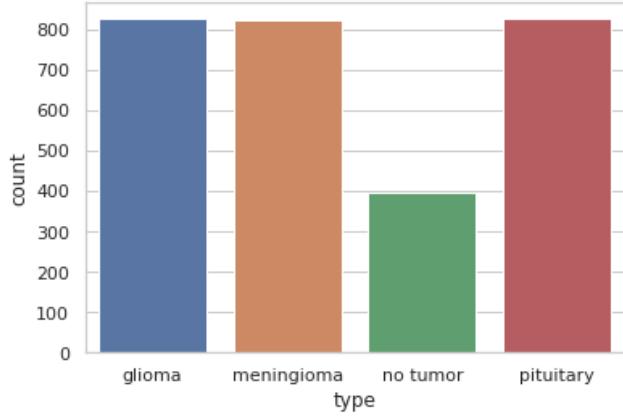


Figure 2: Number of Images in the Training Set

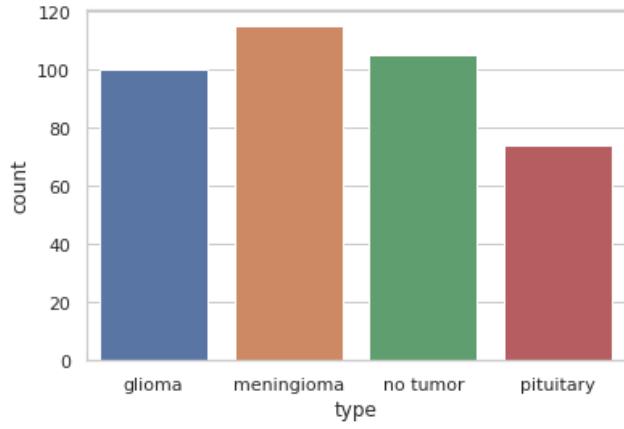


Figure 3: Number of Images in Testing Set

## 2.1 Initial Setup

The project was executed on Google Colab with GPU runtime. The project was coded in Python. The machine learning models were trained using scikit-learn. The deep learning models were trained using TensorFlow 2. The image pre-processing was done using OpenCV while the mathematical and analytical manipulation of data was done using NumPy and Pandas. Matplotlib and seaborn were used for making the graphs. The default size of each image is 512\*512 px. The original images are shown in Figure 4.

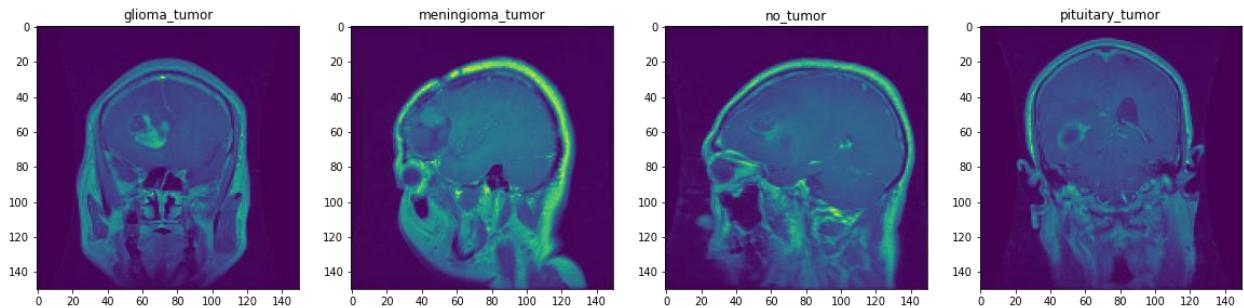


Figure 4: Sample of Original Images

## 2.2 Pre-Processing

In this project, we compare the performance of machine learning models with that of deep learning models. For machine learning models, we use extensive processing to get quality images and remove as much noise as possible without any loss of information. For deep learning models, however, there is little to no work required.

### 2.2.1 Machine Learning Models

We evaluated our models on four sets of images. One set was used without any pre-processing, the other three sets used different kinds of pre-processing. This helped us decide the best way to approach the problem and select the best model for the problem. The models that were used for evaluation are:

1. Naïve Bayes (NB)
2. K-Nearest Neighbor (KNN)
3. Logistic Regression (LR)
4. Random Forest (RF)
5. Support Vector Machine (SVM)

The common pre-processing step involved was to convert the image to grayscale and resize the image from the original size to 150\*150. The original dataset contained four classes. Each class is assigned a number from 0-3. Since we are also performing the detection task, we assign the non-tumor class as 0 and any other class as 1. For the first set of images, there were no further steps involved. The following pre-processing steps were involved with the other three sets:

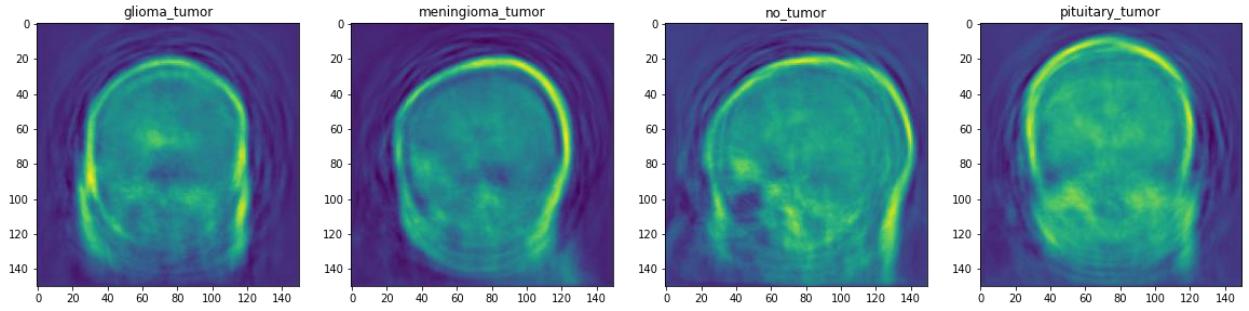
1. Principal Component Analysis (PCA)
2. Median Filtering (MF)
3. Non-local Means Denoising (NMD)

The results obtained using the models for classification and detection are presented in Section 3 and are discussed in Section 4.

#### 2.2.1.1 Principal Component Analysis (PCA)

PCA is a linear transformation algorithm. PCA finds the direction of maximal variance. In short, PCA algorithm tries to find the lower dimensional subspace onto which project the data to minimize the squared projection errors. Let  $X$  be the data matrix of size  $m \times n$ . Alternatively, PCA can be defined as the eigen decomposition of covariance matrix  $X^T X$ . Size of covariance matrix is  $m \times m$ .

In our case, size of the original image is 150 x 150 (i.e., 22,500 dimensions). We represent the image using much fewer dimensions (i.e. 130). The results obtained by PCA were better than the set of images with no pre-processing. But there is no significant improvement. More about the results are discussed in Section 4. The images with PCA are shown in Figure 5.

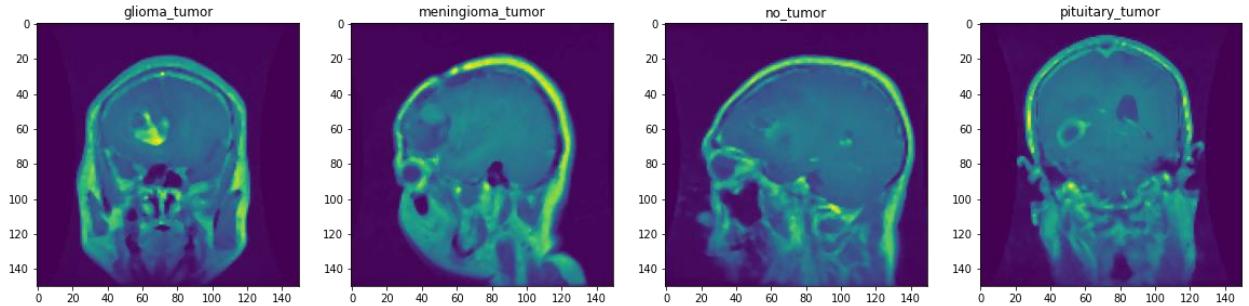


*Figure 5: Images with PCA*

MRI images are often corrupted by a lot of noise. The noise follows the Rice or Rician distribution. The noise in MR images is due to the fluctuations of the magnetic field in the coil. Magnetic Resonance Imaging is corrupted by Rician noise [8]. Rician noise makes quantitative measurements difficult. Therefore, obtaining high-quality denoising images is the most important task of pre-processing. The two methods discussed below demonstrate the most efficient ways to overcome the Rician noise.

#### 2.2.1.2 Median Filtering

Median Filtering [9] is a non-linear denoising technique widely used in image processing. This algorithm is widely popular as it helps preserves the edges and smoothens the images. Median filtering is often used to remove ‘salt and pepper noise’, though it blurs the image a little bit. A window of size  $p$  moves through the image pixel by pixel. It replaces each central pixel by the median value of its neighbors. For the noise present in MRI images, Median Filters outperforms Mean Filters and Wiener (or Adaptive) Filters [10]. The images with median filters applied are shown in Figure 6.



*Figure 6: Images with Median Filters*

Images with median filters are clearly blurred. Figure 7 shows the difference between the original image and the image with median filters. Higher the value of the median filter, higher the denoising will be and this will result in more blurred image.

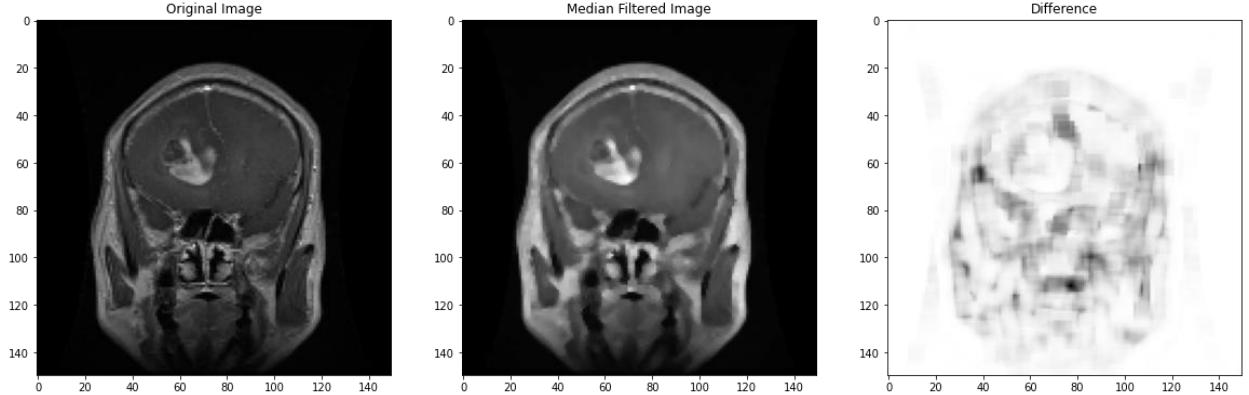


Figure 7: Difference of Original and Median Filtered Image

### 2.2.1.3 Non-local Means Denoising

NLM is a technique to remove noise in image processing. Unlike median filtering where median is used to replace the pixel in the window, mean is used here, as denoted by ‘means’ in the algorithm name. The ‘non-local’ part of the name suggests that the calculated is not local, that is, the replaced value is not the mean of the surrounding neighbors. Here, the pixel value is replaced by the weighted average of all pixels in the image. The weights are assigned based on the similarity of pixels  $i$  and  $j$ . Similar pixel neighborhoods have larger weights. As discussed in [11], NLM is very effective in denoising MRI images.

$$NL(i) = \frac{1}{C(i)} \sum_{j \in I} v(j)f(i,j) \quad \text{where,}$$

$$C(i) = \sum_{j \in I} v(j)f(i,j),$$

$v(j)$  is the value of image at point  $j$

$f(i,j)$  is the weighting function, and

$I$  is the area of an image.

The images with NLM denoising algorithm are shown in Figure 8. At the first glance, there is no difference between the original image and NLM image. But the background of the original image has been smoothed. The difference between the two is visualized in Figure 9.

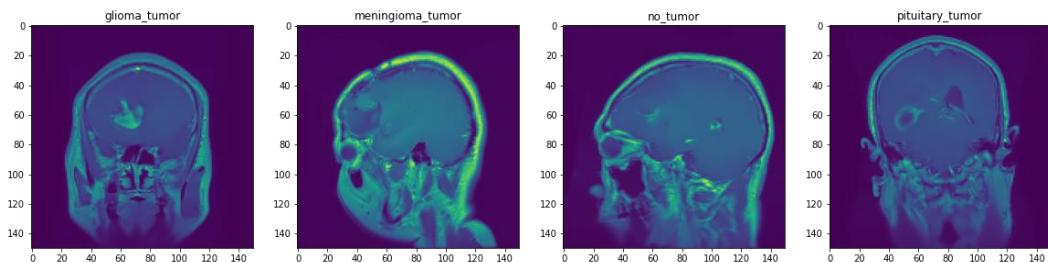


Figure 8: Images with NLM Algorithm

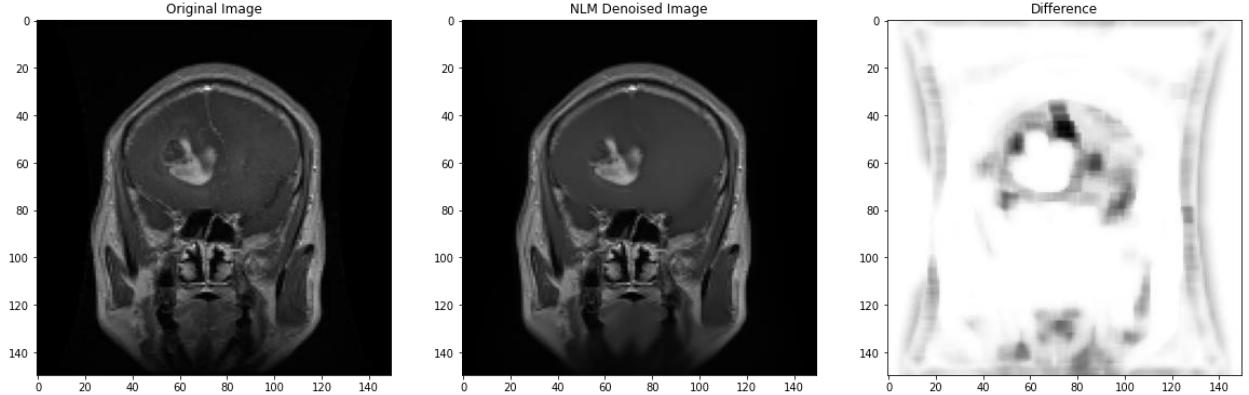
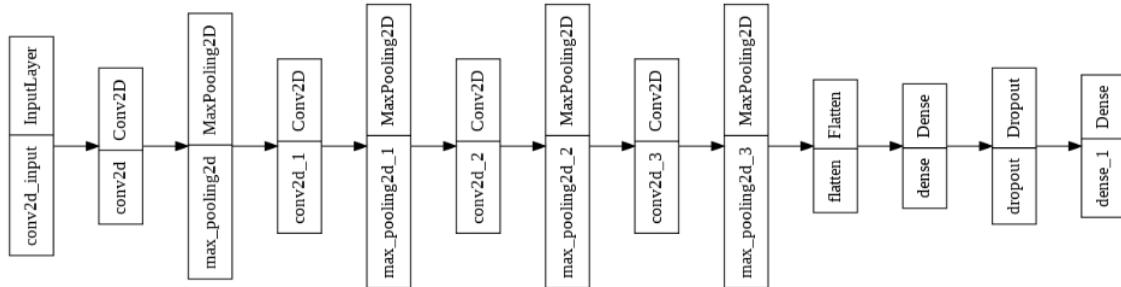


Figure 9: Difference Between the Original Image and NLM Image

### 2.2.2 Deep Learning Models

As the CNNs are good feature extractors, we did not pre-process the images beside resizing them to 150\*150 and converting them to grayscale. The part of CNN training data was used as a validation set. Data was augmented by applying transformations (such as flipping, zooming, cropping, shearing, rotating, etc.) to existing images. The labels were encoded using one-hot encoding. The images were first passed to a CNN model created by us and then compared with the pre-trained models. The architecture of our CNN model is shown in Figure 10.

As shown in the figure below, we used 3 convolution hidden layers followed by 2 fully connected layers. The activation functions used are: ReLU (Rectified Linear Unit) in the intermediate layers and SoftMax in the output layer. The kernel size, padding, and stride parameters were kept as default provided by the tensorflow. We let the model run for 50 epochs with early stopping and we noticed that the validation accuracy stopped increasing after 26<sup>th</sup> epoch.



### 3. Results

#### 3.1 No Pre-processing

Table 1 presents the results from models with no pre-processing done on images. The downside of this approach is the lower accuracy and the longer runtime.

*Table 1: No pre-processing*

| Model Type                      | Classification |        |          | Detection |        |          |
|---------------------------------|----------------|--------|----------|-----------|--------|----------|
|                                 | Precision      | Recall | F1-Score | Precision | Recall | F1-Score |
| <b>Naïve Bayes</b>              | 0.33           | 0.28   | 0.28     | 0.57      | 0.42   | 0.44     |
| <b>K-Nearest Neighbor (k=5)</b> | 0.57           | 0.51   | 0.51     | 0.79      | 0.72   | 0.74     |
| <b>Logistic Regression</b>      | 0.78           | 0.74   | 0.70     | 0.91      | 0.86   | 0.87     |
| <b>Random Forest</b>            | 0.80           | 0.68   | 0.63     | 0.90      | 0.85   | 0.85     |
| <b>Support Vector Machine</b>   | 0.65           | 0.64   | 0.60     | 0.90      | 0.87   | 0.87     |

#### 3.2 Median Filtering

Table 2 presents the results from models with median filtering applied on images. The advantage of this approach is that it produces comparable results with shorter runtime.

*Table 2: Median Filtering*

| Model Type                      | Classification |        |          | Detection |        |          |
|---------------------------------|----------------|--------|----------|-----------|--------|----------|
|                                 | Precision      | Recall | F1-Score | Precision | Recall | F1-Score |
| <b>Naïve Bayes</b>              | 0.28           | 0.24   | 0.22     | 0.51      | 0.34   | 0.34     |
| <b>K-Nearest Neighbor (k=5)</b> | 0.56           | 0.53   | 0.52     | 0.80      | 0.74   | 0.76     |
| <b>Logistic Regression</b>      | 0.80           | 0.74   | 0.72     | 0.91      | 0.86   | 0.87     |

|                               |      |      |      |      |      |      |
|-------------------------------|------|------|------|------|------|------|
| <b>Random Forest</b>          | 0.80 | 0.69 | 0.64 | 0.91 | 0.86 | 0.87 |
| <b>Support Vector Machine</b> | 0.65 | 0.64 | 0.60 | 0.89 | 0.88 | 0.88 |

### 3.3 Non-Local Means

Table 3 presents the results from models with non-local means algorithm applied on images. The advantage of this approach is that it produces much better results and drastically reduces noise.

*Table 3: Non-Local Means*

| <b>Model Type</b>               | <b>Classification</b> |               |                 | <b>Detection</b> |               |                 |
|---------------------------------|-----------------------|---------------|-----------------|------------------|---------------|-----------------|
|                                 | <b>Precision</b>      | <b>Recall</b> | <b>F1-Score</b> | <b>Precision</b> | <b>Recall</b> | <b>F1-Score</b> |
| <b>Naïve Bayes</b>              | 0.30                  | 0.25          | 0.25            | 0.50             | 0.35          | 0.37            |
| <b>K-Nearest Neighbor (k=5)</b> | 0.57                  | 0.54          | 0.52            | 0.79             | 0.77          | 0.78            |
| <b>Logistic Regression</b>      | 0.80                  | 0.74          | 0.72            | 0.90             | 0.86          | 0.87            |
| <b>Random Forest</b>            | 0.80                  | 0.86          | 0.83            | 0.90             | 0.84          | 0.85            |
| <b>Support Vector Machine</b>   | 0.92                  | 0.86          | 0.84            | 0.98             | 0.97          | 0.97            |

### 3.4 Principal Component Analysis

Table 4 represents the results from models with PCA applied on images. The advantage of this approach is that it produces comparable results with shorter runtime, but the results are not any better than non-local means.

*Table 4: Principal Component Analysis*

| <b>Model Type</b>  | <b>Classification</b> |               |                 | <b>Detection</b> |               |                 |
|--------------------|-----------------------|---------------|-----------------|------------------|---------------|-----------------|
|                    | <b>Precision</b>      | <b>Recall</b> | <b>F1-Score</b> | <b>Precision</b> | <b>Recall</b> | <b>F1-Score</b> |
| <b>Naïve Bayes</b> | 0.33                  | 0.29          | 0.29            | 0.55             | 0.41          | 0.44            |

|                                 |      |      |      |      |      |      |
|---------------------------------|------|------|------|------|------|------|
| <b>K-Nearest Neighbor (n=5)</b> | 0.58 | 0.53 | 0.53 | 0.78 | 0.74 | 0.75 |
| <b>Logistic Regression</b>      | 0.52 | 0.51 | 0.49 | 0.71 | 0.68 | 0.69 |
| <b>Random Forest</b>            | 0.80 | 0.77 | 0.72 | 0.90 | 0.85 | 0.85 |
| <b>Support Vector Machine</b>   | 0.72 | 0.70 | 0.66 | 0.96 | 0.85 | 0.87 |

### 3.5 Convolutional Neural Networks

Table 5 presents the results from various CNN models. The first network is the network created by us. The other networks listed below are pre-trained networks.

*Table 5 Convolutional Neural Networks and Transfer Learning*

| Model Type                          | Classification |        |          | Detection |        |          |
|-------------------------------------|----------------|--------|----------|-----------|--------|----------|
|                                     | Precision      | Recall | F1-Score | Precision | Recall | F1-Score |
| <b>Convolutional Neural Network</b> | 0.74           | 0.71   | 0.72     | 0.83      | 0.79   | 0.81     |
| <b>InceptionResNetV2</b>            | 0.91           | 0.87   | 0.90     | 0.95      | 0.91   | 0.94     |
| <b>DenseNet201</b>                  | 0.89           | 0.83   | 0.87     | 0.96      | 0.92   | 0.92     |
| <b>InceptionV3</b>                  | 0.86           | 0.82   | 0.85     | 0.92      | 0.89   | 0.92     |
| <b>VGG19</b>                        | 0.74           | 0.70   | 0.72     | 0.81      | 0.80   | 0.81     |
| <b>EfficientNetB2</b>               | 0.92           | 0.91   | 0.91     | 0.98      | 0.96   | 0.97     |

The training and validation loss for training of EfficientNet are shown in images below. Figure 11 shows the accuracy for detection and Figure 12 shows the accuracy and loss for classification.

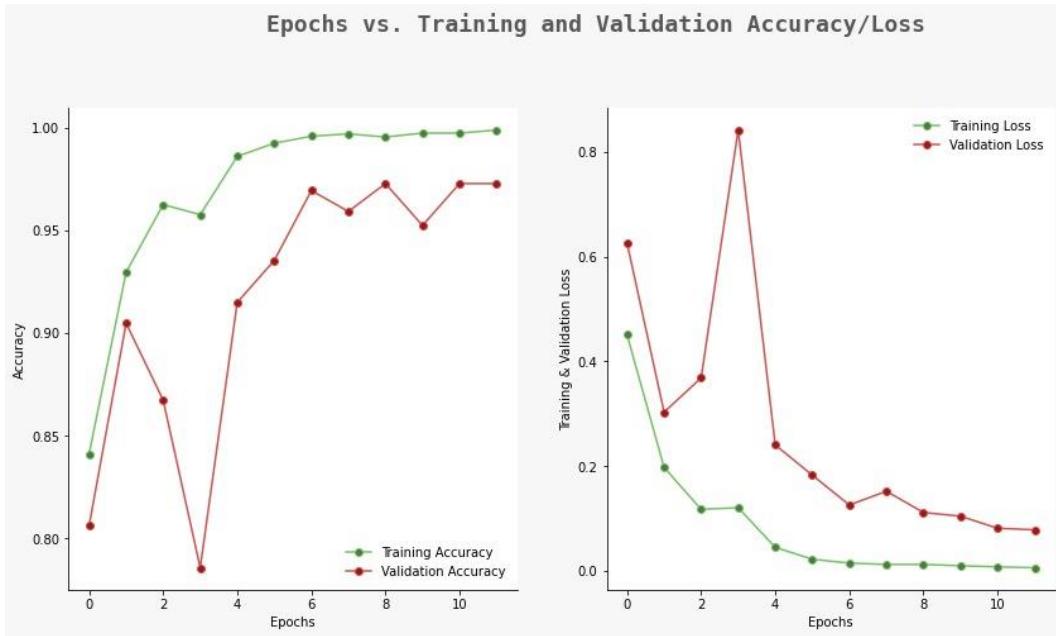


Figure 11: Training and Validation Metrics for Detection for EfficientNet

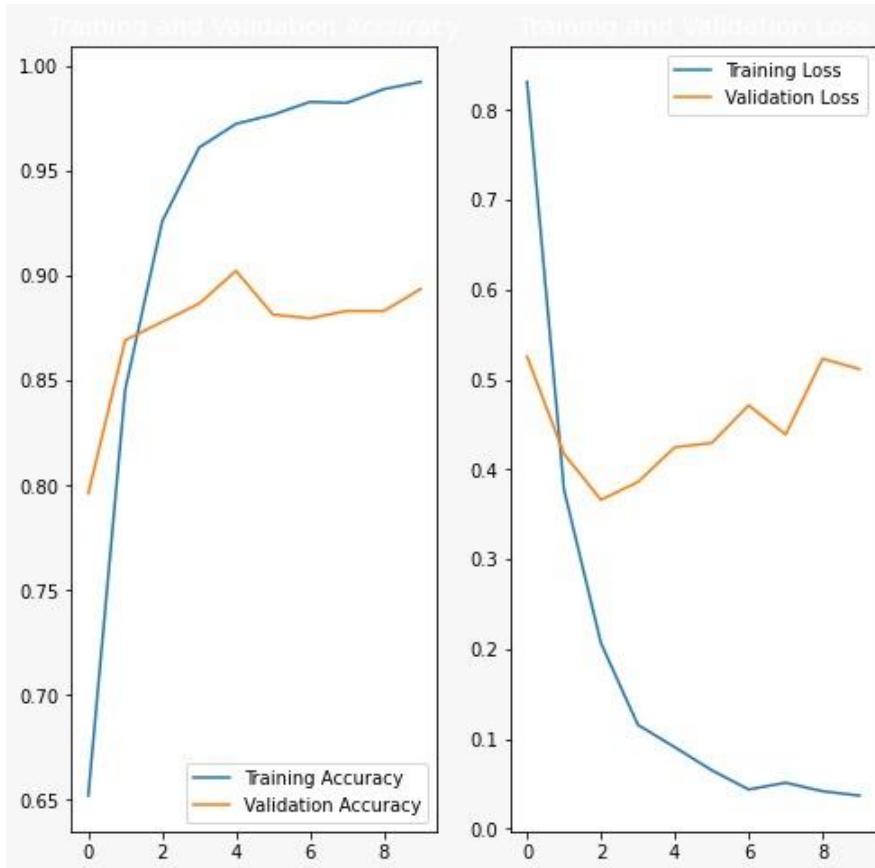


Figure 12: Training and Validation Metrics for Classification for EfficientNet

## 4 Discussion

The overall highest score obtained for classification is 91% and that of detection is 97%. Both results are achieved by Efficient Net. We are using F1 score as metric to compare the models over accuracy as it is:

- More robust towards imbalanced data.
- More relevant when False Positives and False Negatives are crucial.

For detection, our dataset is imbalanced. Also, false positives and false negatives have higher stakes in medical domain. So, it makes sense to include F1-score as our primary metric. Now, we compare the models for every technique and explain the results.

For Table 1 through 4, we observe one common result: score of detection is higher than classification which is expected. In all the tables, we observe Naïve Bayes and K-Nearest Neighbors produce the worst results. Logistic Regression sometimes produces comparable and better, even, results than Random Forest and Support Vector Machines.

In Table 1, Logistic Regression performs the best task of classification and SVM produces the best results for the task of detection. SVM is inherently a robust binary classifier, so that is why it performs a good job at detection while it lags for classification. The surprising result that stands out is the highest score of Logistic Regression over other classifiers. The possible explanation for this result is provided by authors in [12], wherein, they show that in presence of noise variables, Logistic Regression stands out in terms of accuracy.

In Table 2, using median filtering, we obtain similar, and sometimes better and faster results than Table 1. There is an increase in the score but not significant. From Figure 3, we observe that with median filters, image is still noisy. The presence of noise might explain the better performance of Logistic Regression.

In Table 3, we use non-local means denoising algorithm and observe a drastic change in the results. Also from the Figure 6, we can observe that lot of noise is removed from the image and only relevant pixels are retained. In ideal scenario, one would expect either SVM or Random Forest to be superior algorithms here which is proven by the results in the table. The algorithm works wonder as we see the detection score rise to whooping 97%.

In Table 4, we use Principal Component Analysis. PCA does not necessarily improves results of the model [14]. PCA works well when there is a variance in any direction. For brain tumor images from MRI, there is not a lot of variation. Hence, there are not a lot of components we can drop. Therefore, the results are not that better than non-local means but are sometimes better than and sometimes comparable to median filtering.

Table 5 lists the results provided by CNN. As expected, CNN produces better results than the traditional Machine Learning algorithms. This is expected as CNN preserves the spatial

relation between the pixels and for ML models, the spatial correlation is lost when the image is flattened. We used transfer learning models to get the best results. This reduces the time overhead of feature extraction. As expected, older models like VGG16 produce worse results than more recent models like Efficient Net.

## 5 Conclusion

From the above stated results, we infer that, we secured the highest precision and accuracy from EfficientNetB2. We know that EfficientNet architecture varies from 0 to 7. Higher the architecture, better is the model accuracy. We had limited resources that's why we couldn't use the B7 model. EfficientNet outperforms in both classification and detection of brain tumor with respect to its other counterparts. We observed decent F-1 score from InceptionResNetV3 as compared to other Transfer Learning models. When we did PCA as a data-preprocessing method, highest accuracy was achieved by random forest model while doing classification and in detection SVM gave the best results. Whereas, taking non-local means denoising as reference, highest precision and F-1 score was attained from the SVM. The results we got after these two pre-processing methods were expected. However, after applying median filtering as a data pre-processing technique, Logistic regression gave better precision and F-1 score for classification which is quite unusual. While SVM continued to give best results for the detection of brain tumor. This observation was similar when no pre-processing was done before applying all the ML models. Hence, we can generalize that deep learning architectures outperformed machine learning models.

We would like to know what would happen if the images passed to CNNs were pre-processed the same way as that for Machine Learning models. We would like to experiment this and find out. In the future, we plan to integrate machine learning model with deep learning model. CNNs are the best feature extractors. We plan to extract features from CNNs and leave the prediction task to powerful machine learning models like SVM or Ensemble Models. To do this, we would need to remove the final fully connected layers, which are used for prediction. Layers before that are used for feature extraction. We are curious to know the results and see how it turns out to be.

## 6 References

1. <https://www.cdc.gov/cancer/dcpc/research/update-on-cancer-deaths/index.htm#:~:text=Cancer%20was%20the%20second%20leading,females%20and%20315%2C876%20among%20males>.
2. Sultan, H.H., Salem, N.M., & Al-Atabany, W. (2019). Multi-Classification of Brain Tumor Images Using Deep Neural Network. *IEEE Access*, 7, 69215-69225.
3. Garg, G., & Garg, R. (2021). Brain Tumor Detection and Classification based on Hybrid Ensemble Classifier. *ArXiv*, abs/2101.00216.
4. Aiwale, P., & Ansari, S.M. (2020). Detection of Brain Tumor using KNN and LLOYED Clustering.

5. Żurek, Grzegorz & Błach, Michał & Giedziun, Piotr & Czakon, Jakub & Fulawka, Lukasz & Hałoń, Łukasz & Krajewski, Piotr & Dyrka, Witold. (2015). Brain tumor classification using logistic regression and linear support vector classifier.
6. S. T. S. Kumar, K. Rashmi, S. Ramadoss, L. K. Sandhya and T. J. Sangeetha, "Brain tumor detection using SVM classifier," 2017 Third International Conference on Sensing, Signal Processing and Security (ICSSS), 2017, pp. 318-323, doi: 10.1109/SSPS.2017.8071613
7. O. Meet, and R. Kapdi, "Brain Tumor Disease Identification Using Random Forest Classifiers," 2015 IJCSC Volume 7 Number 1, pp.202-204
8. Pereza, M.G., Concib, A., Belen Morenac, A., Andaluza, V.H., & Hernándezd, J.A. (2014). Estimating the Rician noise level in brain MR image. 2014 IEEE ANDESCON, 1-1.
9. Huang, Thomas S.; Yang, George J.; Tang, Gregory Y. (February 1979). "[A fast two-dimensional median filtering algorithm](#)" (PDF). *IEEE Transactions on Acoustics, Speech, and Signal Processing*. **27** (1): 13–18. doi:[10.1109/TASSP.1979.1163188](https://doi.org/10.1109/TASSP.1979.1163188)
10. Isa, I.S., Sulaiman, S.N., Mustapha, M., & Darus, S. (2015). Evaluating Denoising Performances of Fundamental Filters for T2-Weighted MRI Images. *KES*.
11. Manjón, J.V., Carbonell, J., Lull, J.J., García-Martí, G., Martí-Bonmatí, L., & Robles, M. (2008). MRI denoising using Non-Local Means. *Medical image analysis*, 12 4, 514-23 .
12. Kirasich, K., Smith, T., & Sadler, B. (2018). Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets.
13. <https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri>
14. <https://stackoverflow.com/questions/30779009/why-does-classifier-accuracy-drop-after-pca-even-though-99-of-the-total-varian>