CS464 - INTRODUCTION TO MACHINE LEARNING FINAL REPORT

BRAIN TUMOR DETECTOR

Instructor: Ayşegül Dündar
TA: Yusuf Dalva

GROUP 1
- Ahmet Hakan Yılmaz - 21803399
- Akın Kutlu - 21803504
- Esra Genç - 21901962
- Osman Semih Tiryaki - 21801994
- Zehra Erdem - 21801977

TABLE OF CONTENTS

1. INTRODUCTION

Many different techniques have been used to detect brain tumors and humanity struggled to understand the cause and features of such a deadly brain disorder. However, a recent technology called Magnetic Resonance Imaging (MRI) scan produced very accurate results when detecting brain tumors. Today, an MRI scan is considered the best way to detect tumors around the brain and spinal cord area [1]. Instead of being looked at by doctors, MRI scan results can be analyzed by a machine learning model to produce accurate results. This project aims to train a machine learning model that can classify MRI scan results as normal or tumor detected with the type of the tumor among the 3 types of tumor: Glioma, Meningioma, and Pituitary. If it is healthy, it can also recognize it. Thus, brain tumors will be detectable much more precisely and instantly.

In this project, five machine learning methods were used to train five different models to be able to analyze differences among them. In the initial version, it was planned to do 3 models and 2 extra models added later. Logistic Regression, even though it is not widely used for image classification tasks, is one of the methods used in this project to analyze the accuracy rate of a model trained with a slightly non-optimal method. However, logistic regression classifies images within a decreased computational time so this part also demonstrates the tradeoff between the accuracy and pace of the model. The second model was trained using MLP. MLP is one of the more accurate techniques so the comparison of it with CNN demonstrated and interpreted in terms of accuracy, pace, and ease of implementation. Thirdly, the most commonly used technique in image classification tasks: CNN used to train another model. Since CNN models include convolutional layers which use neighboring points to calculate one point, CNN generally has accurate results in image processing. Thus, the most accurate results are expected from the CNN model. Fourthly, Random Forest is a traditional model like Logistic Regression. Lastly, Transfer Learning was selected to try a more complex model.

The dataset found from Kaggle [2]. The used dataset contains 7022 MRI images. There are 4 different labels which are Glioma, Meningioma, Pituitary and No Tumor. The images splitted around %80 (training) - %20 (testing).

2. RELATED WORKS

2.1 Preprocess

The main aim of the preprocessing was to improve model accuracy and train models in less time. Before applying any preprocessing function, two main machine-learning libraries were investigated. Their consistency with different libraries and their usage are examined with their official documents. Using Pytorch’s functions was more appropriate for our project and we decided to use it [3]. Resize, centerCrop, randomHorizontalFlip and toTensor functions were applied. The size of each image in the dataset decreased and each of them became easy to process. In the images, there were black useless areas. In order to get rid of them, centerCrop function was applied to images. In MRI images, the diseased area can be placed in different places. To consider this situation, RandomHorizontalFlip function was used to distribute the tumor into different parts of the images. toTensor function’s aim was to transform images into an array. The needed parameters (mean and standard deviation) for normalize functions were also used by Pytorch’s functions [4].

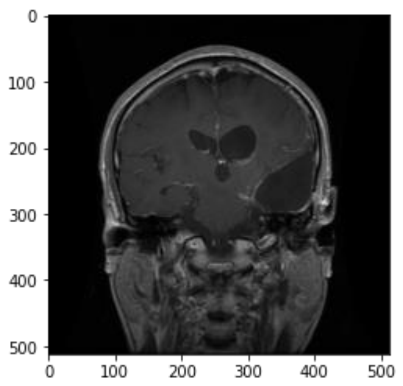


Figure 1: Original Image

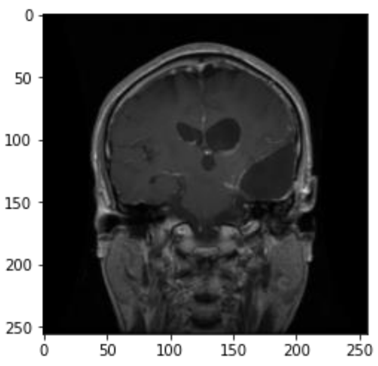


Figure 2: Resized Image

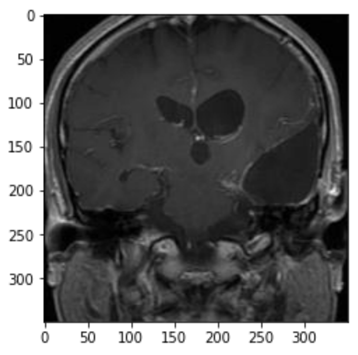


Figure 3: Center cropped image

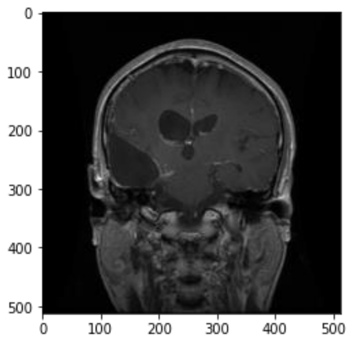


Figure 4: Horizontal flipped image

2.2 Logistic Regression

Logistic regression is one of supervised traditional machine learning techniques and it tries to maximize likelihood function. It uses logistic function to model probabilities of outcomes. For binary cases the output can be 0 or 1. For the multinomial logistic regression outcome can be between 1 to K when K is the number of classes. Different regularizations can be used within the optimization function [5].

2.3 Random Forest

Random forest uses different decision trees within it. Every decision tree takes a sample set from the data and learns from it. If it is wanted all data can be given to every tree. After training, decision trees make a prediction for an input and vote for the output class. The class who gets more votes is predicted as the final output by random forest. Some implementations use votes that are weighted by probability estimations of trees [6].

2.4 MLP

MLP is short for multilayer perceptron and it is a supervised machine learning approach. It is one of the most basic types of artificial neural networks. It is widely used to remove noise from input feature sets [7]. At the basic level, there is a single vertical layer at the input, another similar layer at the output, and one (or more) hidden layer in between to construct an MLP model. The layers of an MLP model convert a sequence of inputs into a single output between 0 and 1 using a set of perceptrons, or equations with inputs, outputs, and weights. [8]. A visual representation of the MLP model is shown below.

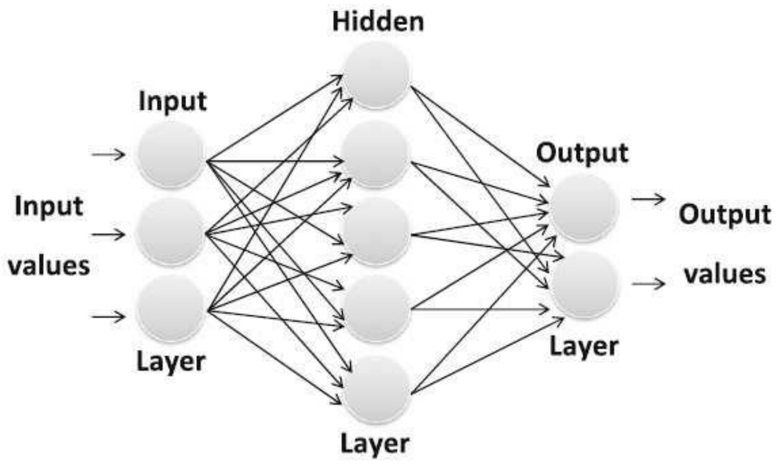


Figure 5: MLP structure [9]

2.5 CNN

CNN is used for pattern recognition in the images and so for image classification. Different from MLP, CNN has convolutional layers. These layers have a kernel with specified sizes. Instead of applying different parameters for every feature, in the convolutional layer, every part of the picture that has the same size as the kernel is multiplied by the same kernel. This means they have shared parameters and if a kernel causes the recognition of an image, since the same kernel will be used in the other parts, other instances of this object within the image can be recognized as well. Therefore, CNN is one of the successful models for pattern recognition and classification [10].

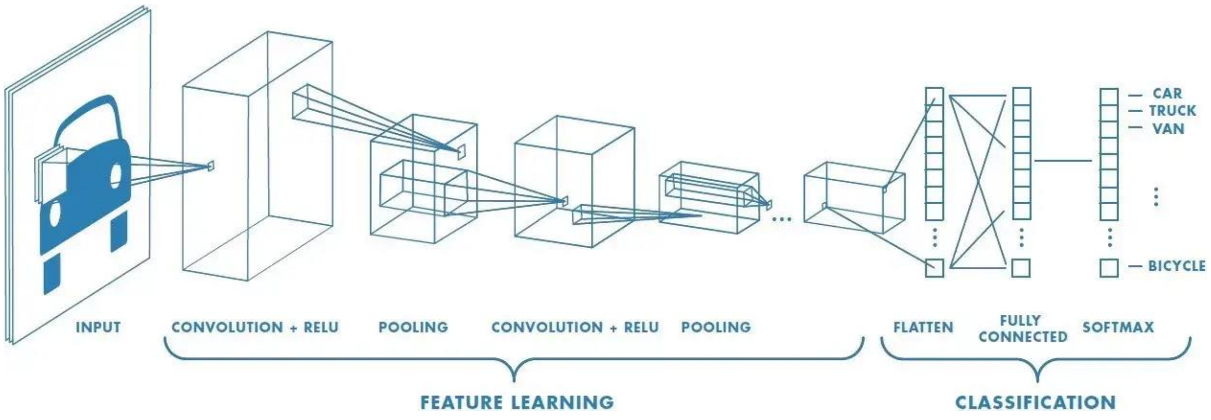


Figure 6: CNN structure [11]

2.6 Transfer Learning

Transfer Learning is a machine learning method when a model trained a task repurposed and retrained for another task. Its main idea is that the learned features from the first task can be useful for the second task even though they are different. Transfer learning reduces training time and provides better model performance because the pre-trained model already learnt useful features for the second task too. For transfer learning Resnet-18 and Resnet-50 models used as pretrained models. ResNet-18 and ResNet-50 are two types of convolutional neural networks that were developed by researchers at Microsoft Research. ResNet-18 and ResNet-50 are trained on large datasets of over 1 million images from the ImageNet dataset labeled with 1000 different class types. ResNet-18 has 18 layers (including the input and output layers), and are less complex compared to Resnet-50. Both of these models are widely used in image classification and it is the reason we selected to use them.

3. METHOD

3.1 Logistic Regression

The Logistic Regression of the Sklearn library is used for training. This model can be used for both binary and multinomial classification. Maximum number of iterations is increased to 1000 because of the large dataset. This model can use different solvers and penalties. Different solvers are tried to get better results such as “lbfgs”, “saga”. The best result is observed when the solver is “lbfgs” and L2 is used as the penalty.

3.2. Random Forest

The Random Forest Classifier of Sklearn library is used for training. The default “gini” criterion is kept. Every decision tree takes the whole data for training. Different numbers of trees are tried during the training such as 50, 80, 100, 150. The best result is observed when the number of trees is 100.

3.3 MLP

PyTorch library was used for the implementation of the MLP model. Pytorch library was chosen to simplify the implementation process with its built in functions.

After choosing the framework, for the next step the parameters are decided as:

- Batch size: 100
- Number of epochs: 10
- Number of layers: 3 (including one input layer, one hidden layer, and one output layer)

Following that, the model is defined as:

- Pytorch’s nn.Module class is used as a backbone for the model. This step is common for neural network models defined in Pytorch. The upcoming layers are added on top of this module.
- The model consists of 3 layers: one input layer, one hidden layer, and one output layer:
    - Input layer gets the input in input dimension 256*256*3 = 196608 and reduces it to 250 dimensions,
    - The hidden layer gets the output of the input layer and reduces the dimension to 100,
    - Output layer gets the output of the hidden layer and gives the output in 4 dimensions.
- The tensor is flattened with the x.view(-1) method because the Linear layer only accepts a vector.

- torch.nn.functional.F.relu function is used as the activation function and it is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.

After defining the model, the model is trained on 5712 * 0.8 images with a train and evaluation loop. For the loss function, the nn.CrossEntropyLoss() is used and for the optimizer, optim.Adam() is used. Adam's optimizer is a technique that optimizes learning rate along the way when the model thinks that it is not learning anymore. In the training loop, the model was tested on the test set in every 100 iterations and the results were obtained.

3.4 CNN

As for the implementation of Convolutional Neural Network, PyTorch library has been used.

There are different frameworks that could be used for implementation of a CNN and for this project's image classification task Pytorch library was chosen since it provides easy implementation and efficient algorithms.

After choosing the framework, for the next step the parameters are decided as:

- Batch size: 100
- Number of iterations: 1000
- Number of epochs: 17

$$\begin{aligned}Number\ of\ epochs &= Number\ of\ iterations \\ &* Batch\ size\ /\ length\ of\ training\ dataset\end{aligned}$$

$$Number\ of\ epochs\ = 1000\ * 100\ /\ 5712\ =\ 17$$

- Learning rate: 0.001

Following that, the model is defined as:

- Pytorch's nn.Module class is used as a backbone for the model. This step is common for neural network models defined in Pytorch. The upcoming layers are added on top of this module.
- 2 convolutional layers with kernel size (5,5) are used as the convolutional part of the model.
- First convolutional layer increased channel size from 3(RGB) to 6, and the second 6 to 16. This means the channel size becomes 16 after convolutional blocks.
- Each convolutional block consists of 3 layers:
    - First the input is given to convolutional layer which increases channel size
    - Rectified linear unit (RELU) layers are used as the activation function for the results of convolutional layers.

    - And results of RELU were fed to the MaxPool2d(2,2) function which is the implementation of max pooling in pytorch. With this layer, dimensionality of features were reduced and performance improved.
- After convolutional blocks, the results are flattened to work on linear layers.
- 2 fully connected linear layers are used to have an array with a size of 4 in the end.
- These 4 elements are the prediction for the current image, each representing the probabilities of types of tumors.

After defining the model, the model is trained on 5712 * 0.8 images with a train and evaluation loop. For the loss function, the cross entropy loss is used and for the optimizer, Adam's optimizer is used. Adam's optimizer is a technique that optimizes learning rate along the way when the model thinks that it is not learning anymore. In the training loop, the model was tested on the test set in every 100 iterations and the results were obtained.

3.5 Transfer Learning

We used Adam optimizer as our optimizer and CrossEntropyLoss function as our criterion function. For transfer learning firstly we freezed all the parameters of the pretrained models and replaced the final layer with a new linear layer to train the model. Final linear layer we put has input equals to the output of the remaining model and output equals to 4 as we have 4 different classes for classification. We applied the same procedure to both Resnet-18 and Resnet-50. We also tried to update the full model (Resnet-18) without freezing parameters and replacing the final layer.

4. EXPERIMENTS

4.1 Logistic Regression

The confusion matrix of the best model can be seen in Figure 7. The accuracy is 0.9000762777. Since logistic regression is a traditional model and the problem is an image classification, this result can be seen as a good one for logistic regression. However, for the problem the better results should be obtained. Another important metric is the false negative rate (the rate of false healthy predictions in all healthy predictions) since when a person is patient and they are not diagnosed it can lead to several consequences. For logistic regression, the false negative rate is 4.1%.

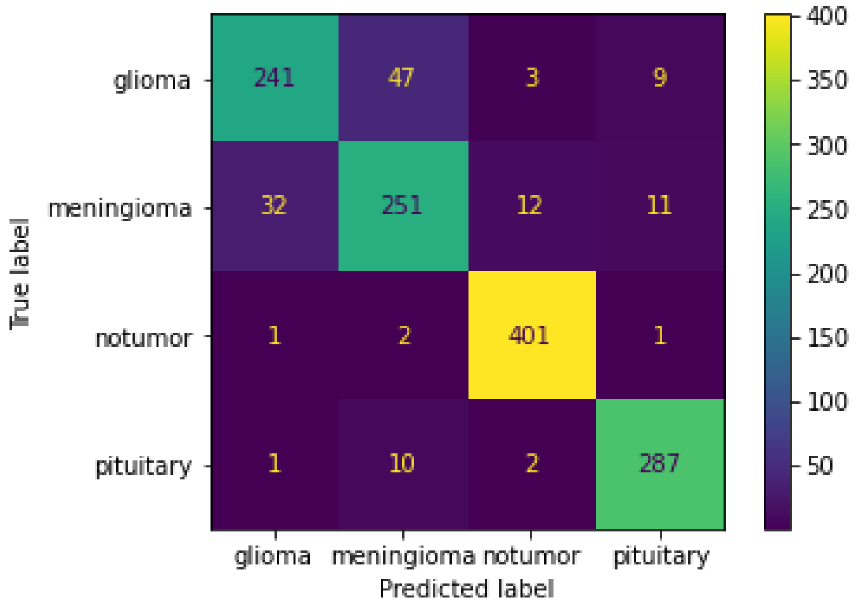


Figure 7: Confusion matrix for logistic regression

4.2 Random Forest

The confusion matrix of the best model can be seen in Figure 8. The accuracy is 0.9405034325. This accuracy can be considered as a good one since it is over 90% and the used model is not a neural network. The false negative rate for this model is 2.1% that is half of the logistic regression result. Therefore, as accuracy and false negative rate this model is much better than logistic regression.

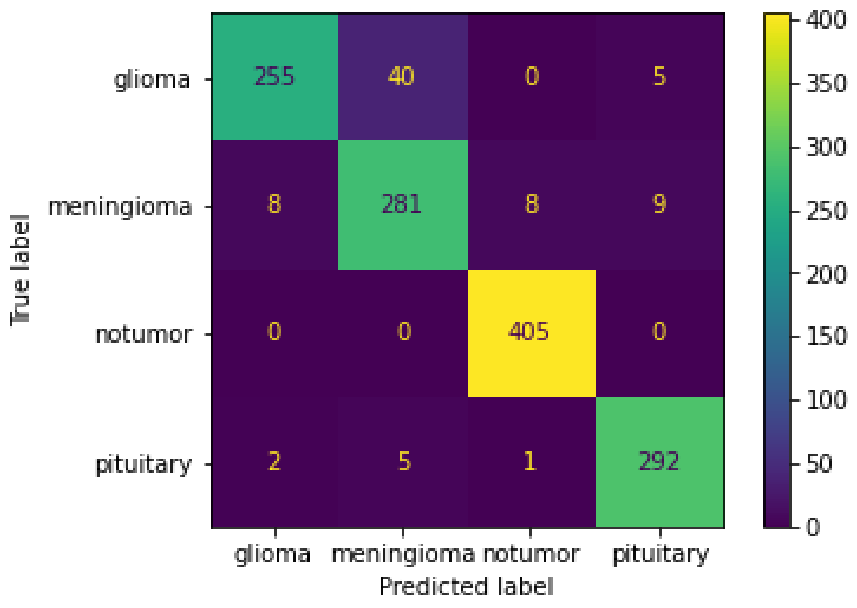
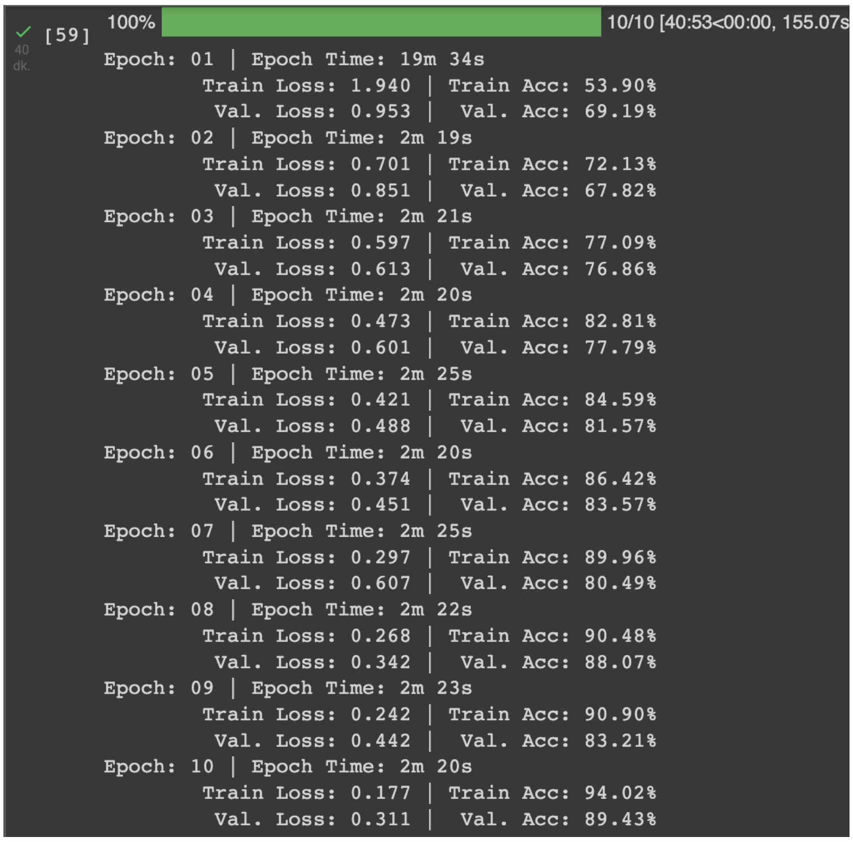


Figure 8: Confusion matrix for random forest

4.3 MLP

Experiments performed displayed increasing accuracy for the MLP model. Final accuracy for 10 epochs was 89.43%. After observing this increasing accuracy, the same experiment was performed with 20 epochs. However, results showed decreasing accuracy rate after 10 epochs. The accuracy and loss rates after each epoch for the first experiment can be found below:



```
✓ [59]  100% ████████████████████████████ 10/10 [40:53<00:00, 155.07s
40
dk.     Epoch: 01 | Epoch Time: 19m 34s
                Train Loss: 1.940 | Train Acc: 53.90%
                 Val. Loss: 0.953 |  Val. Acc: 69.19%
        Epoch: 02 | Epoch Time: 2m 19s
                Train Loss: 0.701 | Train Acc: 72.13%
                 Val. Loss: 0.851 |  Val. Acc: 67.82%
        Epoch: 03 | Epoch Time: 2m 21s
                Train Loss: 0.597 | Train Acc: 77.09%
                 Val. Loss: 0.613 |  Val. Acc: 76.86%
        Epoch: 04 | Epoch Time: 2m 20s
                Train Loss: 0.473 | Train Acc: 82.81%
                 Val. Loss: 0.601 |  Val. Acc: 77.79%
        Epoch: 05 | Epoch Time: 2m 25s
                Train Loss: 0.421 | Train Acc: 84.59%
                 Val. Loss: 0.488 |  Val. Acc: 81.57%
        Epoch: 06 | Epoch Time: 2m 20s
                Train Loss: 0.374 | Train Acc: 86.42%
                 Val. Loss: 0.451 |  Val. Acc: 83.57%
        Epoch: 07 | Epoch Time: 2m 25s
                Train Loss: 0.297 | Train Acc: 89.96%
                 Val. Loss: 0.607 |  Val. Acc: 80.49%
        Epoch: 08 | Epoch Time: 2m 22s
                Train Loss: 0.268 | Train Acc: 90.48%
                 Val. Loss: 0.342 |  Val. Acc: 88.07%
        Epoch: 09 | Epoch Time: 2m 23s
                Train Loss: 0.242 | Train Acc: 90.90%
                 Val. Loss: 0.442 |  Val. Acc: 83.21%
        Epoch: 10 | Epoch Time: 2m 20s
                Train Loss: 0.177 | Train Acc: 94.02%
                 Val. Loss: 0.311 |  Val. Acc: 89.43%
```

Figure 9: MLP iterations

MLP is one of the most preferred neural networks for image classification tasks so the accuracy obtained from the experiments was expected. To further examine the experiment results, a confusion matrix for the MLP model with the 89.43% accuracy is shown below.

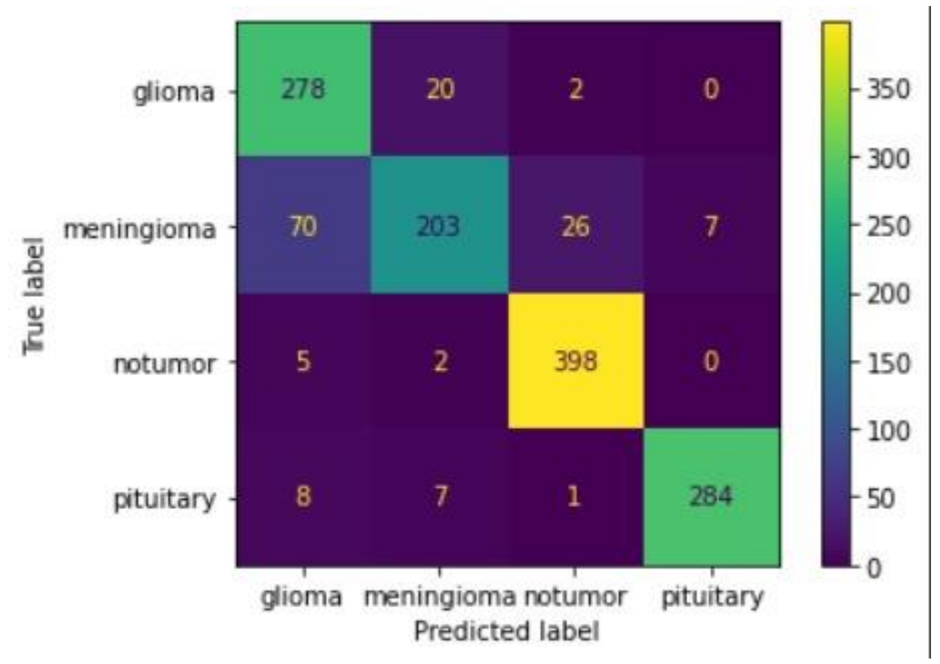
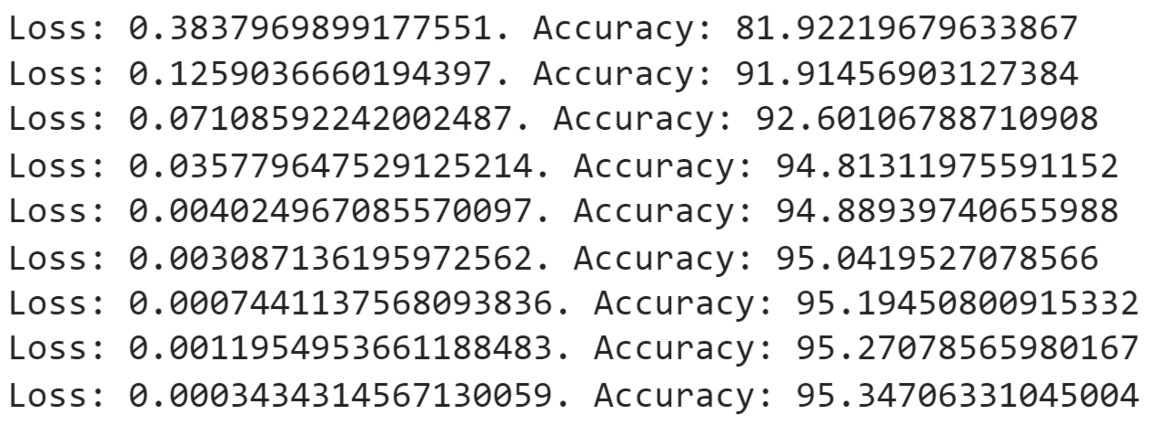


Figure 10: Confusion matrix for MLP

As it can be interpreted from the confusion matrix, the false negative rate (the rate of false healthy predictions in all healthy predictions) is only 6.7% which is very important since falsely classifying a patient as healthy could have devastating results. Therefore, even though the MLP model can be considered successful for classifying the images accurately, it is not the best choice of neural network model in our specific problem, brain tumor detection.

4.4 CNN

Experiments showed very promising results with over 95.3% accuracy for CNN. The accuracy and loss rates after each 100 iteration can be found below:



```
Loss: 0.3837969899177551. Accuracy: 81.92219679633867
Loss: 0.1259036660194397. Accuracy: 91.91456903127384
Loss: 0.07108592242002487. Accuracy: 92.60106788710908
Loss: 0.035779647529125214. Accuracy: 94.81311975591152
Loss: 0.004024967085570097. Accuracy: 94.88939740655988
Loss: 0.003087136195972562. Accuracy: 95.0419527078566
Loss: 0.0007441137568093836. Accuracy: 95.19450800915332
Loss: 0.001195495366118483. Accuracy: 95.27078565980167
Loss: 0.0003434314567130059. Accuracy: 95.34706331045004
```

Figure 11: CNN iterations

CNN is known for its performance for image classification tasks so this much of an accuracy was expected. For detailed visualization of results can be found below.

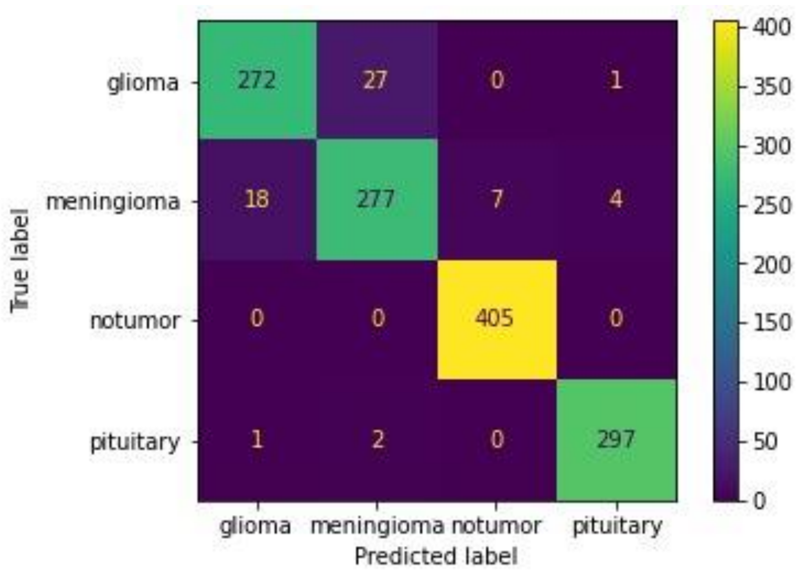


Figure 12: Confusion matrix for CNN

As it can be interpreted from the confusion matrix, the false negative rate (the rate of false healthy predictions in all healthy predictions) is only 1.6% which is very important since falsely classifying a patient as healthy could have devastating results. Therefore CNN model, considering the size of the dataset is very small, can classify the images accurately and have a clear advantage over other models in brain tumor detection.

4.5 Transfer Learning

Initially, we encountered unexpected test accuracies in our results. Then we decided to observe our training accuracy too. We observed our training accuracy increase however our test accuracy does not improve at all. Even considering we already have 4 classes for our classification task, accuracy at the test set was close to the random classification. We got similar results when we used Resnet-50. The reason behind that is pretrained models Resnet-18 and Resnet-50 are too complex for our classification task and as a result of this overfitting occurs. In the below it can be seen the first 8 epochs when we used Resnet-18.

```
Epoch for training: 1. Loss: 1.1570062637329102. Accuracy: 38.28781512605042
Epoch Test: 1. Loss: 1.3767707347869873. Accuracy: 28.222730739893212
Epoch for training: 2. Loss: 1.0680627822875977. Accuracy: 60.25910364145658
Epoch Test: 2. Loss: 1.3633978366851807. Accuracy: 28.527841342486653
Epoch for training: 3. Loss: 0.8015925288200378. Accuracy: 71.18347338935574
Epoch Test: 3. Loss: 1.482558250427246. Accuracy: 28.90922959572845
Epoch for training: 4. Loss: 0.8067604303359985. Accuracy: 76.34803921568627
Epoch Test: 4. Loss: 1.5141246318817139. Accuracy: 29.82456140350877
Epoch for training: 5. Loss: 0.7644698619842529. Accuracy: 78.8515406162465
Epoch Test: 5. Loss: 1.598374843597412. Accuracy: 27.45995423340961
Epoch for training: 6. Loss: 0.6013705730438232. Accuracy: 80.70728291316527
Epoch Test: 6. Loss: 1.479073405265808. Accuracy: 28.83295194508009
Epoch for training: 7. Loss: 0.611485481262207. Accuracy: 81.88025210084034
Epoch Test: 7. Loss: 1.6016472578048706. Accuracy: 29.90083905415713
Epoch for training: 8. Loss: 0.6662734746932983. Accuracy: 82.44047619047619
Epoch Test: 8. Loss: 1.6412585973739624. Accuracy: 27.91762013729977
```

Figure 13: Transfer Learning iterations

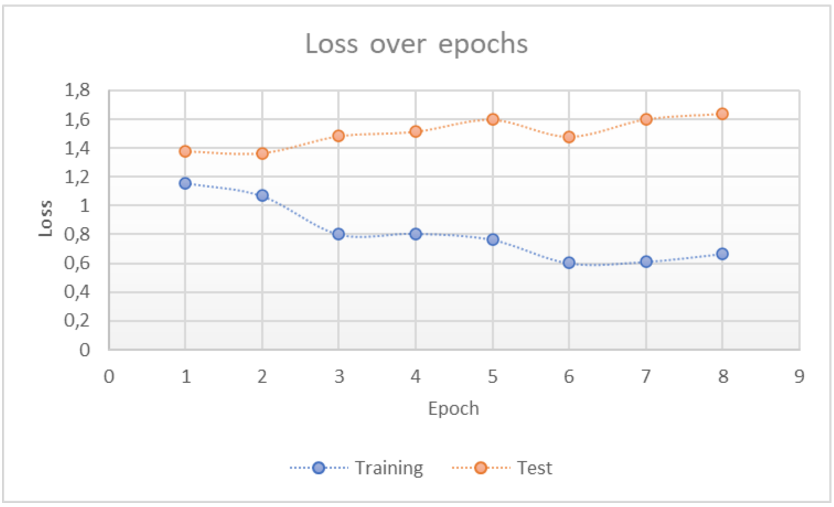


Figure 14: Loss over epochs for transfer learning

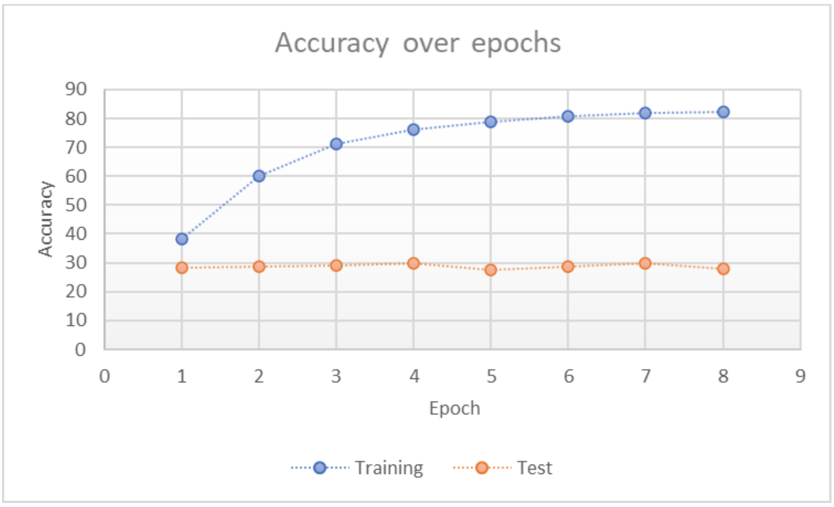
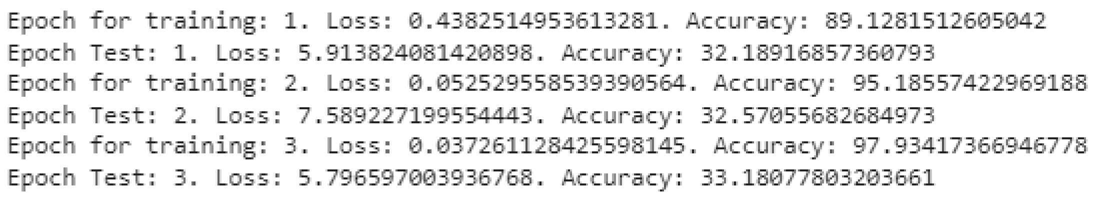


Figure 15: Accuracy over epochs for transfer learning

We also made an experiment on updating all parameters of Resnet-18. Training the model without freezing any layer lasted a lot more (approximately 1 hour for epoch). So we could not get high epoch numbers. However, it can be clearly seen from the result that in that situation there is also overfitting happening just like the situation when we replaced the final layer.



```
Epoch for training: 1. Loss: 0.4382514953613281. Accuracy: 89.1281512605042
Epoch Test: 1. Loss: 5.913824081420898. Accuracy: 32.18916857360793
Epoch for training: 2. Loss: 0.052529558539390564. Accuracy: 95.18557422969188
Epoch Test: 2. Loss: 7.589227199554443. Accuracy: 32.57055682684973
Epoch for training: 3. Loss: 0.037261128425598145. Accuracy: 97.93417366946778
Epoch Test: 3. Loss: 5.796597003936768. Accuracy: 33.18077803203661
```

Figure 16: Transfer learning iterations

5. CONCLUSION

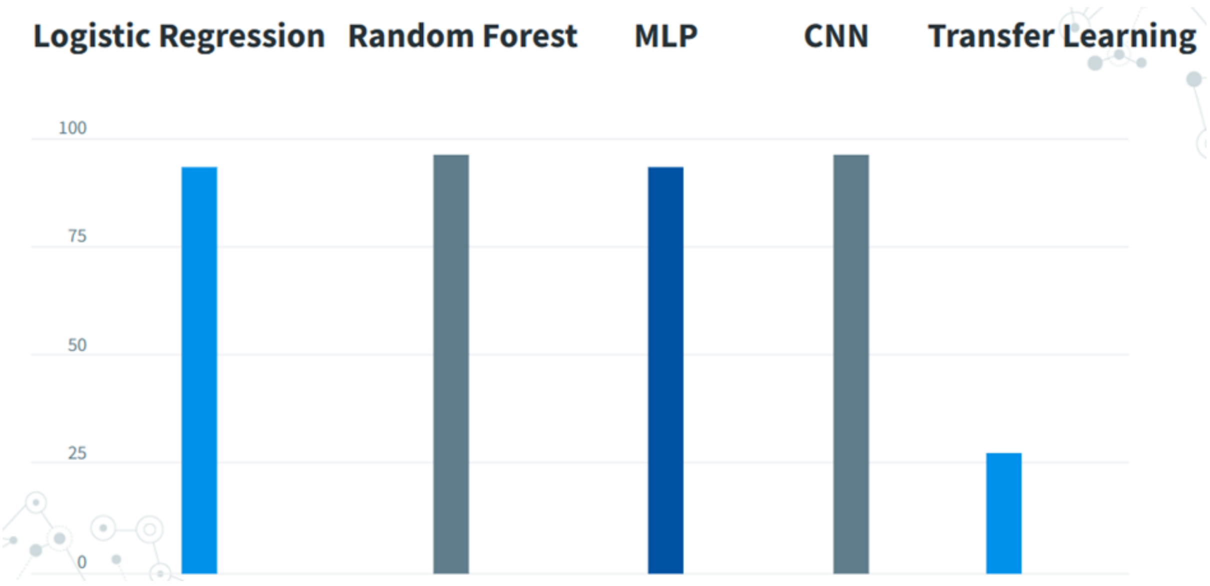


Figure 17: Comparison of model accuracies

The applied preprocessing steps increase the accuracy and reliability of models. When the accuracies of models are compared: CNN (0.95) > Random Forest (0.94) > Logistic Regression (0.90) > MLP (0.89) > Transfer Learning (0.28). CNN gives the best results for brain tumor detection. Random Forest, Logistic Regression, and MLP also give good results. However, Transfer Learning has the worst result by far. As expected, neural network models gave better results. However, the traditional models also worked pretty well. The reason is that the used dataset is not too complex. Our transfer learning model encountered overfit because in transfer learning, the complexity of the pretrained model is significant and our model complexity was high for our classification task.

Another observation from these results are that the glioma and meningioma classes are the ones that are more confused. However, it is hard for a non professional to understand the similarity of differences between MRI scans of these two classes and an explanation for it could not be found by doing a search. A professional might be able to say why these two are most confused.

6. REFERENCES

[1] "Tests for brain and spinal cord tumors in adults," *American Cancer Society*. [Online]. Available: https://www.cancer.org/cancer/brain-spinal-cord-tumors-adults/detection-diagnosis-staging/how-diagnosed.html. [Accessed: 03-Dec-2022].

[2] M. Nickparvar, "Brain tumor MRI dataset," *Kaggle*, 24-Sep-2021. [Online]. Available: https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset?select=Training. [Accessed: 28-Dec-2022].

[3] S. V. R, "Pytorch transformations: 10 pytorch transformations for Data scientists," *Analytics Vidhya*, 22-Apr-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/04/10-pytorch-transformations-you-need-to-know/. [Accessed: 03-Dec-2022].

[4] "Transforming and augmenting images," Transforming and augmenting images - Torchvision 0.14 documentation. [Online]. Available: https://pytorch.org/vision/stable/transforms.html. [Accessed: 03-Dec-2022].

[5] "1.1. Linear Models," *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression. [Accessed: 26-Dec-2022].

[6] "1.11. ensemble methods," *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html#forest. [Accessed: 26-Dec-2022].

[7] G. Latif, M. Mohsin Butt, A. H. Khan, M. Omair Butt, and J. F. Al-Asad, “Automatic Multimodal Brain Image Classification using MLP and 3D glioma tumor reconstruction,” *2017 9th IEEE-GCC Conference and Exhibition (GCCCE)*, 2017.

[8] T. Prole, “What is an MLP, and why should you care?,” *Medium*, 26-Apr-2022. [Online]. Available: https://towardsdatascience.com/what-is-an-mlp-and-why-should-you-care-bfa06741a183. [Accessed: 02-Dec-2022].

[9] M. Alkasassbeh, “MLP structure”. [Online]. Available: https://www.researchgate.net/figure/MultiLayer-Perceptron-MLP-sturcture-334-MultiLayer-Perceptron-Classifier-MultiLayer_fig2_309592737. [Accessed: 19-Dec-2022].

[10] K. O'Shea and R. Nash, “An introduction to Convolutional Neural Networks,” *arXiv.org*, 02-Dec-2015. [Online]. Available: https://arxiv.org/abs/1511.08458. [Accessed: 03-Dec-2022].

[11] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way” [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 03-Dec-2022].