Brain Tumor Classification using Deep Learning

Riyad Bin Rafiq Arefa Patwary Shoham Weiss
University of North Texas
Denton, Texas
riyadbinrafiq@my.unt.edu arefapatwary@my.unt.edu
shohamweiss@my.unt.edu

Abstract

Brain tumor is the deadliest condition for humans that can reduce one's life expectancy. In the medical field, clinicians and doctors detect and classify the tumors from Magnetic Resonance Imaging (MRI) by analyzing the data. But they can miss classify the tumor due to errors and biases. In this context, deep learning techniques can help to solve the problem as this field has become popular and shown satisfactory performance in image classification tasks. Existing solutions heavily relied on different image processing and deep learning methods and acquired acceptable accuracy in classification. In this project, we explored applying different deep learning techniques, particularly Convolutional Neural Network (CNN), UNet and Transfer learning on the dataset to classify three types of brain tumor (meningioma, glioma and pituitary tumor) from MRI data. The dataset consists of 3064 images from 233 patients. We evaluated the trained model using precision, recall and F1 metric so that we could compare the applied techniques and develop an understanding on how the models generalize on the unseen MRI data.

1. Introduction

Digital images are increasingly used for diagnosis in medical fields [1]. With better imaging technologies, more information can be analyzed for better diagnoses. Yet, human bias and error can still contribute to misdiagnoses. A machine learning model can learn to identify features in images and aid the diagnostician in finding the correct diagnoses [2].

Brain tumor detection and classification is usually done by human inspection of a highly trained radiologist. The data most commonly used for detecting brain tumors is Nuclear Magnetic Resonance Imaging (MRI) data. Radiologists look for specific characteristics in the MRI image to identify if there is a brain tumor and if so, which type of tumor. Several image processing techniques have been used to help these radiologists identify said features. We have proposed using deep learning techniques trained on enhanced MRI data to aid in the classification of brain tumor types.

Our project focused on investigating different deep learning techniques for extracting information from the MRI data. Specifically, we look at vectorizing the enhanced MRI images with UNet, classifying tumor types using CNNs, and attempting to use transfer learning on more complex pretrained models for tumor classification.

2. Related Work

Over the years, there has been much work on brain tumor classification using different techniques of deep learning. Some of the notable works are mentioned in this section. Recently in 2022, In a paper, a novel hierarchical deep learning-based brain tumor (HDL2BT), classification is proposed using CNN. The model used a total of 3264 images for the glioma, meningioma, and pituitary and no tumor class. Following tumor detection, the system divides the tumor into different classifications [3]. Another study in 2021 proposed utilizing a deep learning architecture called the EfficientNet-B3 model to automatically classify brain tumors in MRI data. This model can decipher MRI images with three different types of cancers: meningioma, glioma, and pituitary tumors [4]. In another the paper, the researchers provided a fully automatic brain tumor segmentation and classification model with a multiscale approach utilizing a Deep Convolutional Neural Network which can evaluate MRI scans with three types of tumors: meningioma, glioma, and pituitary tumor, in sagittal, coronal, and axial perspectives, and does not require preprocessing of input

images to remove skull or vertebral column components [5]. Another paper proposed three different CNN models for three separate categorization tasks. The first CNN model detects brain tumors with a 99.33% accuracy rate. The second CNN model can classify the brain tumor into five types: normal, glioma, meningioma, pituitary, and metastatic and the third CNN model can categorize brain tumors into three grades: Grade II, Grade III, and Grade IV. This is the first study to use CNN for multi-classification of brain tumor MRI images, with the grid search optimizer tuning practically all hyper-parameters [6].

In another study, they presented a three-class deep learning model for categorizing Glioma, Meningioma, and Pituitary tumors but did not include no-tumor. They employed a pre-trained InceptionV3 model and applied transfer learn-ing that extracts features from brain MRI images and classifies them using the softmax classifier method. The proposed approach outperforms all existing methods in a patient-level five-fold cross-validation process on the CE-MRI dataset from figshare [7]. In another paper the researchers have augmented tumor region via image dilation which is used as the ROI and then that region is split into increasingly fine ring-form subregions and then evaluated. The efficacy of the proposed method on a large dataset with three feature extraction methods, namely, intensity histogram, gray level co-occurrence matrix (GLCM), and bag-of-words (BoW) model [8]. Lastly, there has been research that developed a Content-based image retrieval (CBIR) system for retrieving brain tumors in T1-weighted contrast-enhanced MRI image and proposed a novel feature extraction framework to improve the retrieval performance [9].

3. Datasets and Metrics for Experiments

The dataset used in this project is from the Brian Tumor Image Dataset on Kaggle [10] gathered by Southern Medical University, Guangzhou, China. The dataset contains 3064 T1-weighted contrast-enhanced MRI images from 233 patients. The samples are of meningioma [708 slices], glioma [1426 slices], and pituitary tumor [930 slices] and are labeled as such.

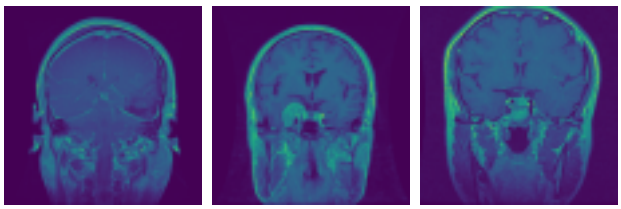

Figure 1. Left: Meningioma - usually appear as an enhancing mass on the outside lining of the brain tissue, which may or may not brighten with contrast. Malignant meningiomas can also invade the brain tissue [11]. Middle: Glioma - called intra-axial brain tumors because they grow within the substance of the brain and often mix with normal brain tissue [12]. Right: Pituitary tumor - and abnormal growth in the pituitary gland. The pituitary is a small gland in the brain. It is located behind the back of the nose [13].

We used the prelabeled data as our source of truth for training the CNN and transfer learning models. We used precision, recall and F1 score to evaluate our models.

4. Methodology

In this project, we applied three deep learning techniques such as UNet, CNN and transfer learning for the classification task. We evaluated the UNet relative to another saliency map approach. We also compared the performance accuracy among the classification approaches. The applied methods are described as follows:

4.1 Saliency Maps

To give brain tumor diagnosticians a better tool for diagnosing a type of brain tumor, it is not enough to predict the type of tumor present based on a contrast-enhanced MRI image, it's also important to also show the diagnostician what the model was looking at when it made the classification that it did. This is where saliency maps come in. A saliency map will show which neurons fired from the original image contributed most to the resulting classification. There are multiple methods to create saliency maps, we attempted a novel custom method and a known gradient tape method.

4.1.1 UNet

Our custom attempt at creating a saliency map involved using a UNet model. The following is the proposed technique: First, we train a UNet model to output the original input image. This first step's purpose is to create a decoder from a learned latent space. Our UNet structure is shown in figure 2. After training a full UNet, we train the encoder sub-structure of the UNet as shown in figure 3 separately as a classifier. To make the encoder a classifier, we extend the encoder with a flatten layer and a fully connected layer to train this sub-structure as a CNN classifier of the 3 brain tumor types. This step is so that the new latent space created by the CNN a.k.a encoder is tuned on the features that help classify the image. Third, we connect the newly trained encoder of the classification task with the previously trained decoder, as shown in figure 4. This final model will recreate the images from the latent space the classifier learned. Thus, our CNN model will classify the image and our decoder will tell us what the model was looking at when making this

classification.

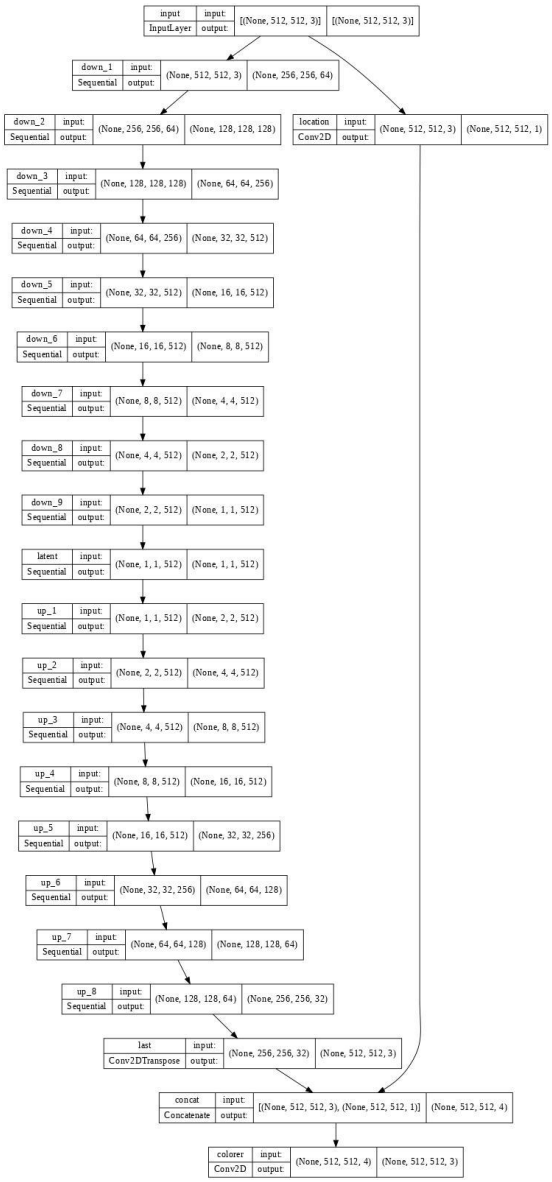

Figure 2: The UNet model used to train the decoder. Each down layer consists of a 2D convolution with 3x3 kernels and a batch normalization. Each up layer consists of 2D deconvolution with 3x3 kernels and a batch normalization.

To ensure the UNet model recreated the image from the latent space, we split the model into two side, a left side consisting of all the convolutional down sampling and up sampling layers and the right side consisting of one untrainable convolutional layer with a kernel of size 1x1 and 1 feature space. The left layer is the full autoencoder which we are training. A known issue with using autoencoders when trying to regenerate images is that localization information tends to be lost in the encoding phase. Therefore a skip connection (the right layers), is used to pass the full image back through to the output.

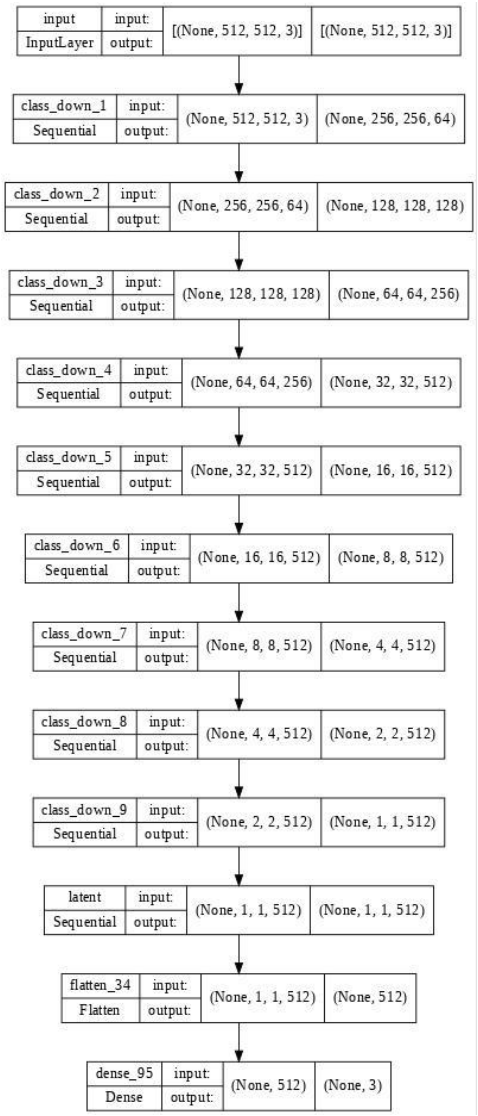

Figure 3: The CNN/ encoder model. The name of the down layers are now class_down to be used later as the layers of the Final UNet's encoder. An extra dense layer is added for the classification task.

Since the right layers pass the whole image to the output layers, the model can rely on only these layers to recreate the image. To prevent the model from relying on the right layers to recreate the image, we make the right layers convert from the 3 feature space consisting of red, green, and blue of the original image to a 1 feature space image using the 1x1 kernel convolutions. We further make all the layers on the right untrainable. In this configuration,

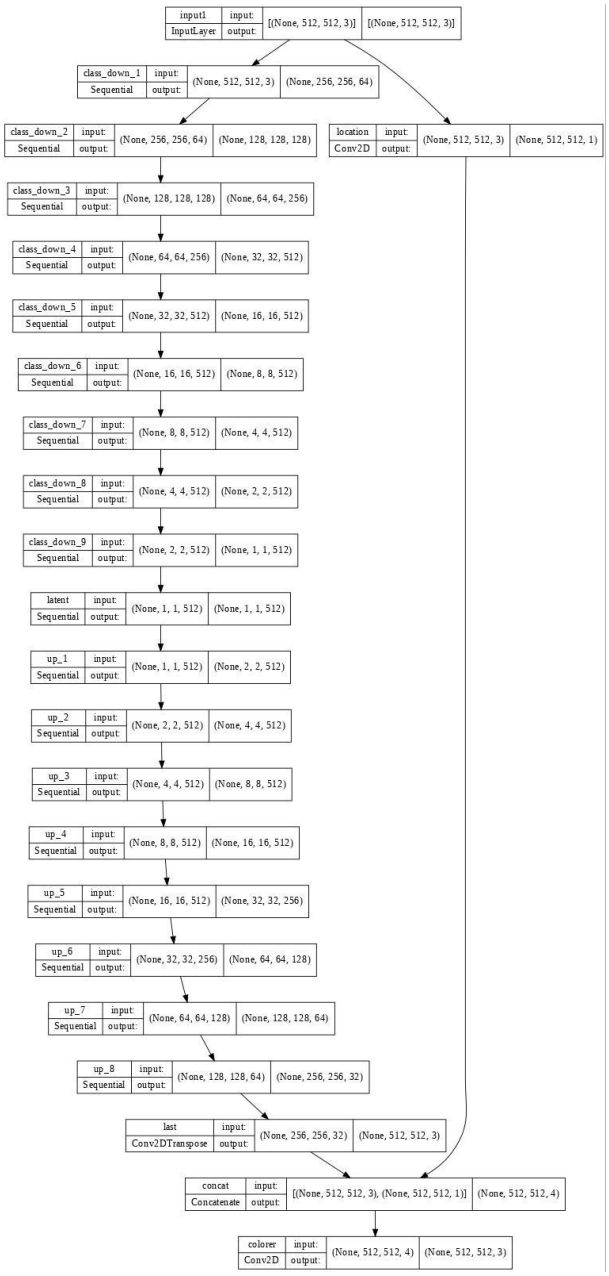

Figure 4: The Final UNet. The encoder layers are named class_down as they are taken from the classifier and the up layers are taken from the previously trained UNet. This UNet is not trained, just used to combine the pretrained weights and provide an image of what the encoder looks at.

all of the location details in the image are passed through the right layers without passing information about the color of each region. The coloring of the details are learned by the left layers. To prove this is the case, we look at the right (figure 6) and left (figure 5) layers separately as the model trains to confirm that the left layers are learning the correct coloring and the right layers are learning nothing.

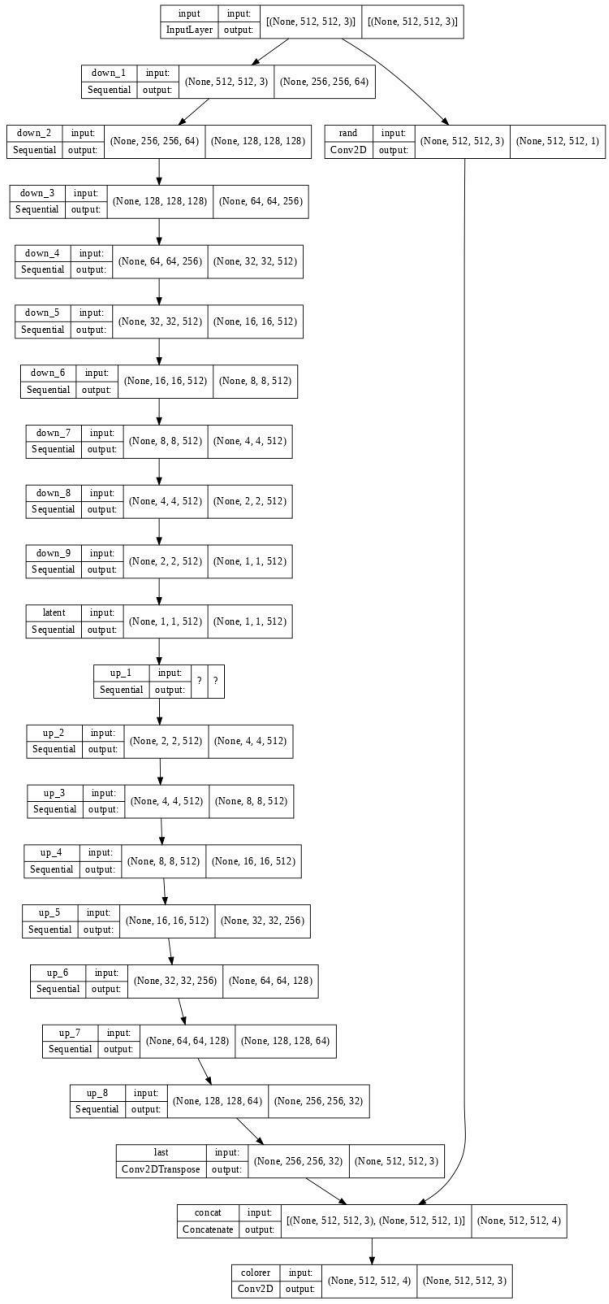

Figure 5: This is the Left side only model. This model has a ‘rand’ layer on the right instead of the ‘location’ layer in the original UNet. This is so when the weights are loaded into this model, new weights are created in that layer. This is so we only use the weights trained on the right sub-layers only. This is to see the contribution of the left layers.

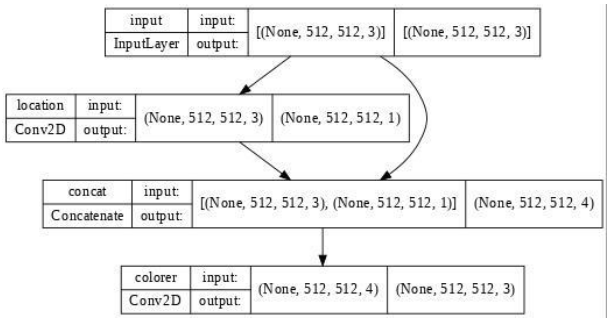

Figure 6: This is the right side of the original UNet model. The weights from the location layer are transferred from the trained UNet and the other weights are new random weights. This is to see the contribution of the location weights and prove they don't contribute to the color of the output image.

4.1.2 Gradient Tape

A common method for creating a saliency map for a CNN classifier is by using Tensorflow's built-in gradient tape functionality. The gradient tape saves the operations and gradients done through the network during a classification task. This gradient tape functionality is mainly used for the autograd functionality but can be hijacked to create saliency maps.

To create a saliency map using gradient tape, we can look at the gradients between the highest activated neuron, a.k.a the classification, in the last layer and the original image. This gradient represents which neurons from the original image contributed the most to the chosen neuron. We can then plot the neurons that had the most contribution in image shape to see which area in the original image the model was looking at when it decided on a classification. The results for this are shown in figures 13 and 14.

4.2 Convolutional Neural Network

4.2.1 CNN Structure and Parameter Settings

The CNN model in this case is of the Sequential type. In Keras, the simplest technique to build a model is sequential. It allows one to layer-by-layer construct a model. The 'add()' function is used to add layers to our model. For this CNN we have applied six layers of Conv2D layers which are convolution layers that will deal with the input images. 64 in the first layer, 256 in the second layer, 512 in the 3rd layer and 1024 in the last four layers are the number of nodes in each layer. This number is adjustable to be higher or lower, depending on the size of the given dataset. For CNN, this combination works well, so we stuck with this sequence. We have used ReLU as our activation function for the first six layers. In neural networks, this activation function performs extremely well. Our first layer takes in an input shape of image size 512,512,3 where 3 signifies that the images are RGB. We have added a 'Flatten' layer between the last Conv2D layer and the Dense layer, which serves as a connection between the convolution and dense layers. We have used a Dense layer for our output layer as well. Dense is a standard layer type that is used in many cases for neural networks. We have 3 nodes in our output layer, one for each outcome (0–2). We have used activation 'Softmax' for the output layer. After initializing the CNN, we used the maxpooling2D to down sample the images and lower the size of the feature maps by reducing the number of parameters to learn and the computation in the neural network. We have used a L2 regularizer of 0.001 in the first Dense layer in order to minimize the adjusted loss function and prevent overfitting or underfitting and a dropout of 0.1 after the first dense layer which is a regularization technique to prevent overfitting in the model.

4.2.1 Compile

To compile the CNN we passed 3 parameters: optimizer, loss and metrics. We have used 'adam' as our optimizer which adjusts the learning rate throughout training. We have used 'SparseCategoricalCrossentropy' for our loss function. A lower loss score indicates that the model is performing better. We have used the 'accuracy' metric to see the accuracy score on the validation set while training the model.

4.3 Transfer Learning

Transfer learning is a method where knowledge learnt from one task (source task) is transferred to learn a new task (target task) [14]. This approach can save time in modeling and improve the performance accuracy with less labeled data. In computer vision, several pretrained models such as VGG19, Inceptionv3, ResNet50 are available to leverage the knowledge transfer technique from one domain to another.

In this project, we applied transfer learning with the pretrained VGG16 model. We included one fully connected layer of 128 units followed by the classification layer with 3 units at the end of the model architecture. For this, we imported the VGG16 pretrained model and the necessary libraries using TensorFlow and Keras and all the implementation has been done in Python. The input image size is 224x224 and batch size is 32. Moreover, to optimize the learning, we used 'Adam' optimizer with 0.001 learning rate. We chose these hyperparameters by

trying different combinations of values. However, to prevent the overfitting we applied the early stopping approach.

5. Results and Analysis

5.1 Saliency Maps

We trained the UNet for 30 epochs and saved a UNet model every 5 epochs to later analyze. In figure 7, we can see the UNet's training progress and in figure 8, we can see the result of each UNet, showing it was better at coloring the image as it trained, saturating at around 25-30 epochs.

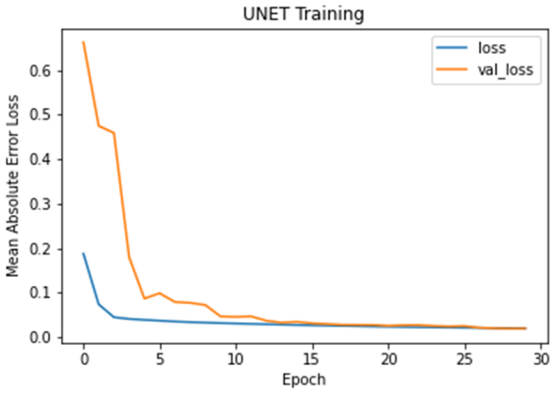

Figure 7: This is the Loss vs. Epoch graph of the UNet model.

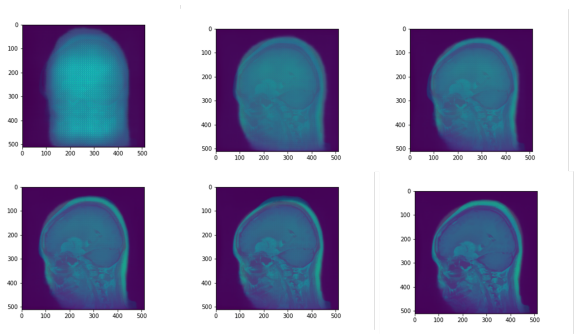

Figure 8: This is the resulting images from the UNet model through several epochs. From top left to bottom right epochs: 5, 10, 15, 20, 25, 30.

To confirm only the left side of the network was contributing to the training, we also looked at the output of individual sides of the network. Figure 9 shows the results of the right network with random values in the layers of the left side. The results show that the right network did not contribute to the coloring of the image, and just passed localization information. The left side's results are shown in figure 10. In these results, the left autoencoder side got the weights from the previously trained UNet model and random weights for the layers on the right side. The results show that this side of the network learned the coloring of the image.

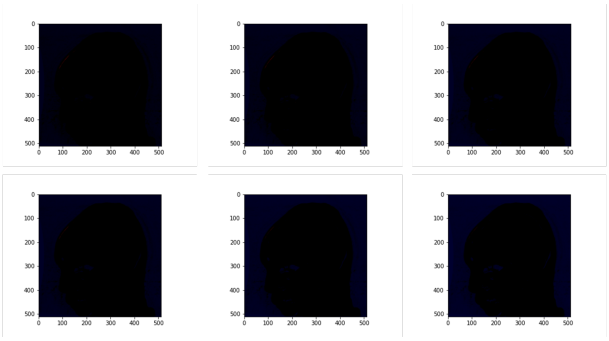

Figure 9: This is the resulting images from the right sublayers of the UNet model through several epochs. From top left to bottom right epochs: 5, 10, 15, 20, 25, 30.

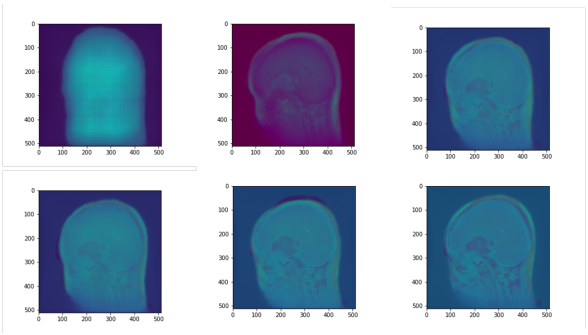

Figure 10: This is the resulting images from the left sublayers of the UNet model through several epochs. From top left to bottom right epochs: 5, 10, 15, 20, 25, 30.

The encoder was then trained as a CNN for 160 epochs. The 80-100 epochs are shown in figure 11. On approximately the 20th epoch, the model was performing at an 80-90 percent training accuracy, but the confusion matrix was not very good. We trained the model for the extra 140 epochs to make sure the encoder is hyper-tuned on the details in the image that will classify each tumor type. Figure 12 shows the confusion matrices of different stages in CNN's training.

In figures 13 and 14, we compare the UNet and gradient tape saliency methods. We see that the UNet method shows a more general region of where the model is looking and the gradient tape method shows more varied and exaggerated regions of where the model was looking.

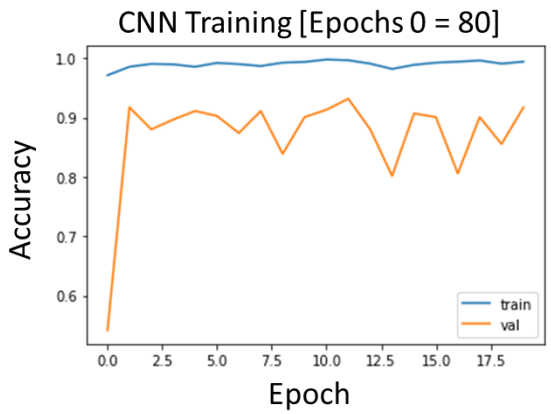

Figure 11: This is the CNN/encoder training accuracy form epoch 80-100.

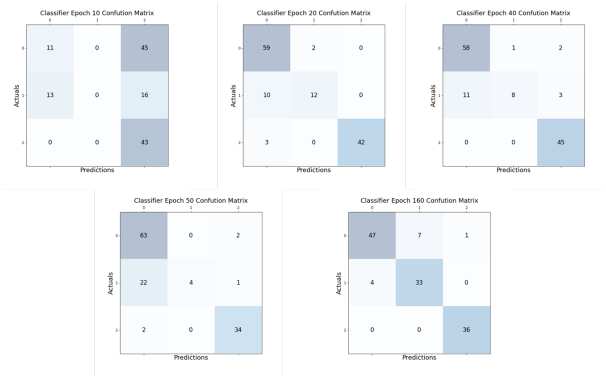

Figure 12: The confusion matrices for the CNN/encoder at several epochs.

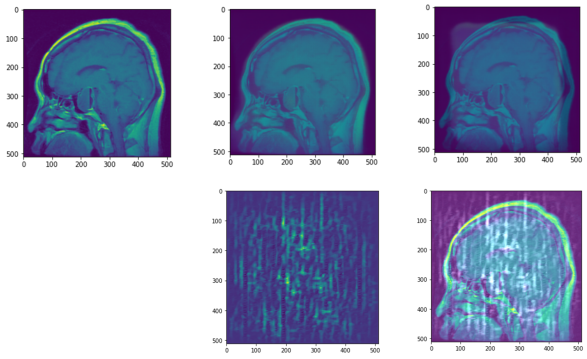

Figure 13: From top left to bottom right: original image, original UNet output, UNet with encoder trained as classifier, gradient tape saliency map, gradient tape saliency map on top of original image.

The gradient tape method also shows bright lines indicating the CNN was looking for vertical lines heavily during the classification task. The UNet method has a main brighter rectangular area around the whole head indicating the classification model was generally looking at the head, but more detail is not visible.

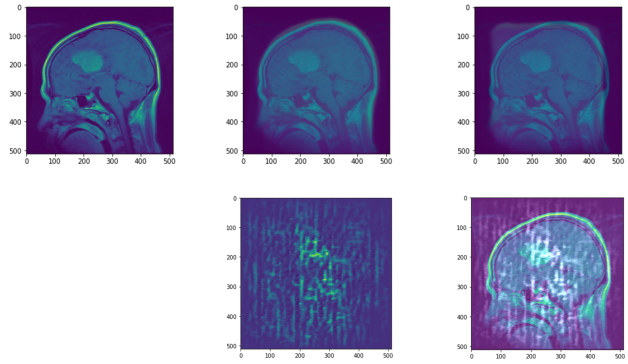

Figure 14: From top left to bottom right: original image, original UNet output, UNet with encoder trained as classifier, gradient tape saliency map, gradient tape saliency map on top of original image.

5.2 Convolutional Neural Network

5.2.1 Epochs and Accuracy and Loss Graphs

We have run the CNN for 20 epochs and we got a training accuracy of 0.98 and training loss of 0.47. We have achieved a Val accuracy of 0.93 and a val loss of 0.67.

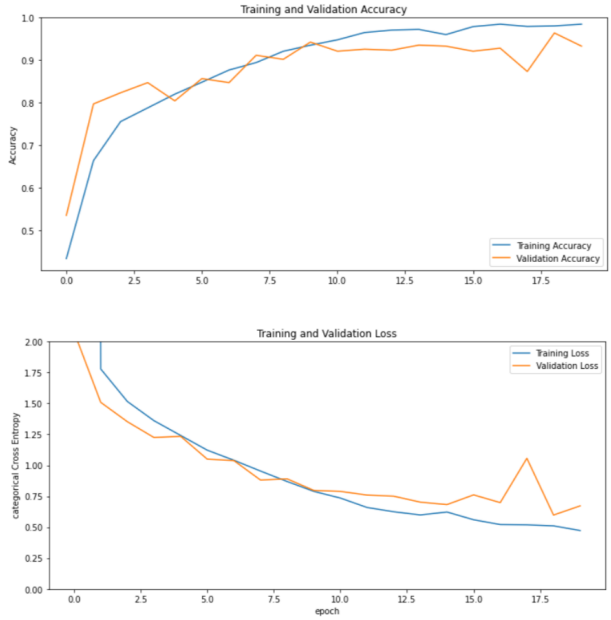

Figure 15. Accuracy and loss vs epochs

From the graph we can see that the loss and accuracy for both training and testing were almost consistent. We got a test accuracy of 92% and test loss of 0.74 over 32 test samples.

5.2.2 Confusion Matrix

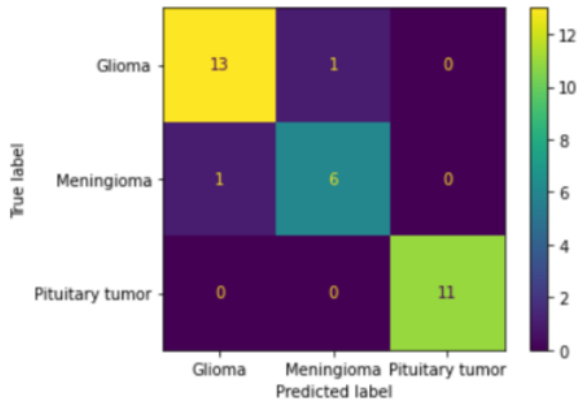

Figure 16. Confusion matrix

5.2.3 Precision, Recall and F1-Score per Class Using CNN

Table 1. Precision, recall and F1-score for each class using custom CNN.

Class	Precision	Recall	F1-score
Glioma	93%	93%	93%
Meningioma	86%	86%	86%
Pituitary tumor	100%	100%	100%
Average	**93%**	**93%**	**93%**

5.3 Transfer Learning

We divided the dataset into training, validation and testing dataset. The training, validation and testing set contain 2452, 420 and 192 images respectively. Transfer learning has the power to show better accuracy in very early epochs and so, the model showed preferable accuracy and was trained for only 9 epochs. After training, the model was evaluated on the test set and the approach got 93.23% accuracy. We have also provided the confusion matrix (Figure 17) and the precision, recall and F1 scores in Table 1 to quantify the recognition of the trained classifier for each of the 3 classes.

Table 2. Precision, recall and F1-score for each class using transfer learning

Class	Precision	Recall	F1-score
Glioma	95%	91%	93%
Meningioma	78%	84%	81%
Pituitary tumor	95%	97%	96%
Average	**90%**	**91%**	**90%**

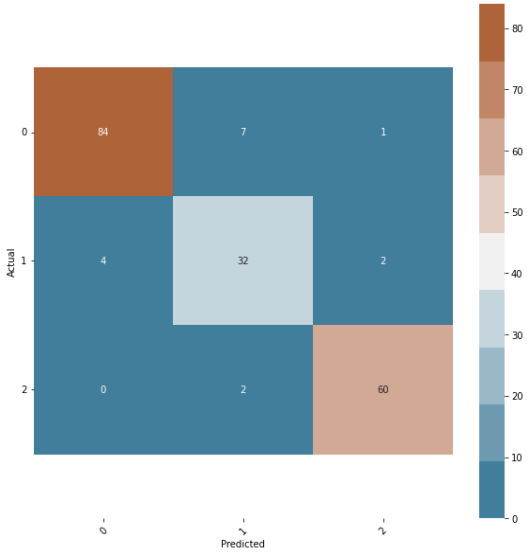

Figure 17. Confusion matrix.

6. Conclusion and Future Work

Brain tumor classification is one of the challenging tasks according to clinicians as the misclassification can occur due to bias and errors. In this project, we implemented three techniques in classifying three types of brain tumor. We also created several saliency maps using a UNet method and a gradient tape method. Our saliency maps provided insight into what the CNN model was learning but did not prove an adequate tool to localize the tumors in the image. Our custom CNN and transfer learning approach acquired 92% and 93.23% accuracy on the test set respectively. Our approaches provided an acceptable performance in the classification task and can assist the radiologists to categorize the brain tumor from the MRI.

Neither the UNet method nor the gradient tape method provided much insight into where the tumor might be in the image. The UNet method provided a generalized area of where it was looking in the image without specifying very deeply what subregions it was looking at. The

gradient tape method provided very specific and varied areas in the image. These regions did not help specify where the brain tumor could be.

Although there are similar ideas to it, we have not found our UNet method in other literature. It might not have performed very well at localizing the brain tumor in our data, but it did show the general area the model was focusing on. This means that with images where the more obvious information in the image leads to the classification, the UNet method would be able to show which region the model looked at. Thus, the model has potential as a region proposal method trained on only the classifications of an image. This model can generalize to give regions that affected the classification and propose where the objects in the image are. The main benefit for this method as a region proposal method is that the regions of the data do not need to be labeled ahead of time. Furthermore, the regions proposed by the UNet will not be restricted to specific shapes and can be much more flexible.

The UNet method can also be used as a label creation technique for another object detection model to train on. Instead of having a person label the location of objects in an image. The UNet can be trained with just the classification of the object and it will generate the location that was affecting this classification, thus creating some general localization regions which could be used as labels for an object detection model.

To improve the UNet model, we could also introduce more dropout layers to force brighter smaller regions. The decoder layers might be too well tuned at recreating the original image and in a sense ignore the differences in the latent space from the new classification information. To improve on this, we would need to make the model more sensitive to the latent space layer.

References

1. Dougherty G. Digital Image Processing for Medical Applications. 2009. doi:10.1017/cbo9780511609657

2. de Bruijne M. Machine learning approaches in medical image analysis: From detection to diagnosis. Med Image Anal. 2016;33: 94–97.

3. Khan AH, Abbas S, Khan MA, Farooq U, Khan WA, Siddiqui SY, et al. Intelligent Model for Brain Tumor Identification Using Deep Learning. Applied Computational Intelligence and Soft Computing. 2022;2022. doi:10.1155/2022/8104054

4. Mantha T, Eswara Reddy B. A Transfer Learning method for Brain Tumor Classification using EfficientNet-B3 model. 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS). 2021. pp. 1–6.

5. Díaz-Pernas FJ, Martínez-Zarzuela M, Antón-Rodríguez M, González-Ortega D. A Deep Learning Approach for Brain Tumor Classification and Segmentation Using a Multiscale Convolutional Neural Network. Healthcare (Basel). 2021;9. doi:10.3390/healthcare9020153

6. Irmak E. Multi-Classification of Brain Tumor MRI Images Using Deep Convolutional Neural Network with Fully Optimized Framework. Iranian Journal of Science and Technology, Transactions of Electrical Engineering. 2021;45: 1015–1036.

7. Soumik MFI, Hossain MA. Brain Tumor Classification With Inception Network Based Deep Learning Model Using Transfer Learning. 2020 IEEE Region 10 Symposium (TENSYMP). 2020. pp. 1018–1021.

8. Cheng J, Huang W, Cao S, Yang R, Yang W, Yun Z, et al. Enhanced Performance of Brain Tumor Classification via Tumor Region Augmentation and Partition. PLOS ONE. 2015. p. e0140381. doi:10.1371/journal.pone.0140381

9. Cheng J, Yang W, Huang M, Huang W, Jiang J, Zhou Y, et al. Retrieval of Brain Tumors by Adaptive Spatial Pooling and Fisher Vector Representation. PLoS One. 2016;11: e0157112.

10. Kavi D. Brain Tumor Image Dataset. Available: https://www.kaggle.com/denizkavi1/brain-tumor

11. Meningioma diagnosis and treatment. In: National Cancer Institute [Internet]. 17 Sep 2018 [cited 23 Feb 2022]. Available: https://www.cancer.gov/rare-brain-spine-tumor/tumors/meningioma

12. Gliomas. [cited 23 Feb 2022]. Available: https://www.hopkinsmedicine.org/health/conditions-and-diseases/gliomas

13. Pituitary tumors. [cited 23 Feb 2022]. Available: https://www.hopkinsmedicine.org/health/conditions-and-diseases/pituitary-tumors

14. Pan SJ, Yang Q. A Survey on Transfer Learning. IEEE Trans Knowl Data Eng. 2010;22: 1345–1359.