Brain Tumor Classification using Convolutional Neural Network

CS 512 Computer Vision Project Report

Kinjal Kachi (kkachi@hawk.iit.edu)
A20449343
Department of Computer Science
kkachi@hawk.iit.edu

Omkar Pawar (opawar@hawk.iit.edu)
A20448802
Department of Computer Science
opawar@hawk.iit.edu

ABSTRACT

Advancements in deep learning and availability of data and processing power has made it possible to develop cutting edge use cases of neural networks in the domain of Computer Vision. These developments are being implemented in the field of medicine to speed up the process of a lot of different things. One of them is detection of Brain Tumor in its early stages and classifying it accordingly. There exist a number of such tumor segmentation algorithms with varied approaches. In this project we try to use deep learning and convolutional neural networks to perform brain tumor classification.

Keywords—Brain Tumor, Image Segmentation, Convolutional Neural Networks, Deep Learning, Computer Vision.

PROBLEM STATEMENT

In this project we try to create a neural network model which can be used to predict the class of the Magnetic resonance imaging (MRI) scan if there exists any of the following types of tumor, namely Meningioma, Glioma, Pituitary tumor. The main focus of our project will be image segmentation. It is the process of dividing an image into different regions based on the characteristics of pixels to identify objects or boundaries to simplify an image and more efficiently analyze it. Some medical applications of segmentation include the identification of injured muscle, the measurement of bone and tissue, and the detection of suspicious structures to aid radiologists (Computer Aided Diagnosis, or CAD). We will try to find tumor regions in the given image and classify the tumor in three given categories.

INTRODUCTION

Brain Tumor is an abnormal growth of brain cells caused by uncontrolled cell division. These tumors can be fatal if not treated properly. The most common type of tumor is Meningioma, Glioma, Pituitary tumor.

A meningioma develops on the brain's meninges and is a tumor of leptomeningeal origin. The meninges are the three layers of membrane that cover the brain and spinal cord. Meningiomas are one of the most common forms of brain tumors, accounting for roughly 20% of brain tumors.
Pituitary tumor occurs in the anterior body of the pituitary gland. Pituitary tumors can affect the function of the pituitary gland. Gliomas are tumors in the central nervous system (brain or spinal cord) and peripheral nervous system that form out of various types of glial cells.

In order to treat these tumors, one must detect them in their early stage. Detection of these tumors is based on the MRI (Magnetic Resonance Imaging) scans of the brain. We can leverage computer vision and deep learning techniques to classify these MRIs into different classes of brain tumor. Using deep learning techniques that learn the pattern of different kinds of brain tumor is efficient because manually segmenting images can be a time consuming process and is prone to human errors.

Medical Image segmentation is the process of detecting boundaries within the image. Image Segmentation can be used in the medical field due to its variety of applicability. It helps people to plan specific treatment according to the results. Segmentation is the process
of partitioning an image into different meaningful segments. In medical
imaging, these segments often correspond to different tissue classes, organs. We can find the boundaries of the tumor in MRIs and then train a model to learn the pattern for each class
of the tumor. This model can be used to predict the class of new MR images
which can help in early detection of the tumor.

DATA

The dataset that we will be using for our project is a brain tumor dataset posted by Jun Cheng on figshare.com. The link for the dataset is here. This data is organized in matlab data format (.mat file). Dataset Consists of 3064 T1 Weighted contrast-enhanced images from 233 patients with three kinds of a brain tumor

1) Meningioma (708 slices)
2) Glioma (1426 slices)
3) Pituitary tumour (930 slices).

Region of Interests (3 types of tumors) are segmented and provided in .mat file format as a part of the database.

Data Structure

1) cj data.label: 1 for meningioma, 2 for glioma, 3 for pituitary tumor
2) cjdata.PID: patient ID
3) cjdata.image: image data

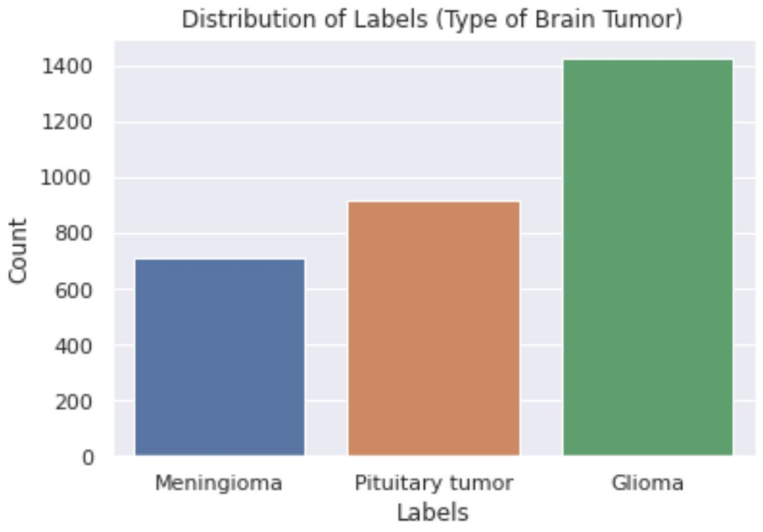
PROPOSED SOLUTION AND IMPLEMENTATION DETAILS

The data that we have at hand is in .mat format. We have a total of 3064 image files. We have adopted the following steps: Data Pre-processing, Model building and Fitting Test data.

DATA PREPROCESSING:

1) First task will be to transform data to python readable format.
2) Make use of two distinct python lists to store the features and the labels.
3) Convert the python lists into Numpy arrays to make it usable in Computer Vision models.
4) Divide the data in train and test datasets with the ratio of 70-30 for training and testing sets respectively.
5) Use stratified sampling on data to obtain a sample population that best represents the entire population being studied, making sure that each subgroup of interest is represented.
6) Pickle the divided data in order to increase the reliability and reusability of the data.

In the plot below, we can see the labels of the data are distributed unevenly. This is the reason we have used stratified sampling while creating training and validation sets.



Model Building:

Implement a basic convolutional neural network to classify the types of Brain Tumour types. We have undertaken following steps:

1. Construct a CNN
2. HyperParameter Tuning
3. Inference

Construct CNN

The solution for this Project is written in python as a programming language and utilizing Keras, Tensorflow GPU framework to implement convolutional neural networks. We start with a basic network and evaluate its performance. Then using Hyperparameter Tuning, we increase the performance of the model to make more accurate predictions.

CNN Model layers:

- Two Convolution Layers with Pooling.
- One Dropout Layer.
- Two Fully connected layers.

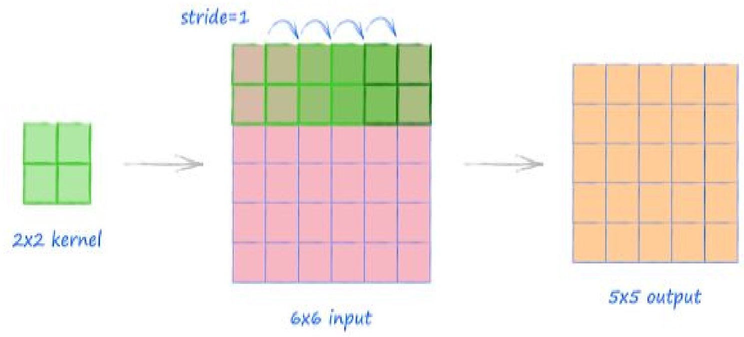
Following are the elements of our Model.

Convolution Neural Networks:

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. Convolutional networks were inspired by biological processes in the connectivity pattern between neurons.

Strides:

Stride is the number of pixels shifted over the input matrix. When the **stride** is 1 then we move the filters to 1 pixel at a time. When the **stride** is 2 then we move the filters to 2 pixels at a time and so on. We have used a Stride of (2,2)



Flattening:

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We **flatten** the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.

Activation Function:

An activation function is a function that is added into an artificial neural network in order to help the network learn complex patterns in the data. We have used two activation functions namely, Softmax and ReLU (Rectified Linear Unit).

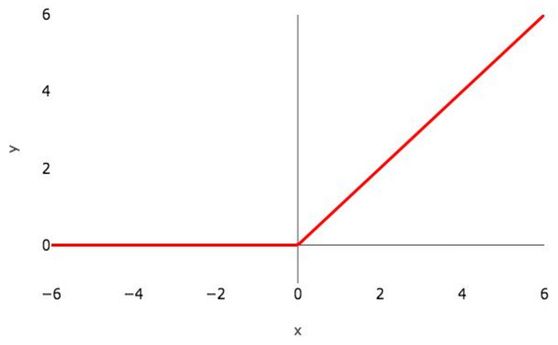
Softmax:

The softmax is a more generalised form of the sigmoid. It is used in **multi-class classification problems**. Similar to sigmoid, it produces values in the range of 0–1 therefore it is used as the final layer in classification models. **Here we have three classes at the output, hence it's usage at the end layer is justified.**

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

ReLU(Rectified Linear Unit) :

Networks are trained better with Relu function. Relu does not saturate unlike step and sigmoid function. It is a linear function hence useful in training the network. This is a widely used activation function, especially with Convolutional Neural networks. It is easy to compute and does not saturate**.**



Loss Function:

Categorical_Cross_Entropy: Categorical cross entropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one**.**We have used the Softmax activation function before using Categorical Cross Entropy, this ensures the output is Probability Distribution.

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i$$

Optimizers :

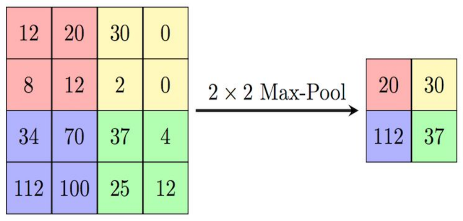
Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses. We have used RMSprop in our network design.

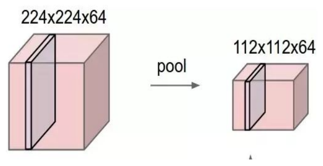
RMSPROP:

RMSprop optimizer restricts the oscillations in the vertical direction. Therefore, we can increase our learning rate and our algorithm could take larger steps in the horizontal direction converging faster.

MaxPooling:

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.





Batch Normalization:

We normalize the input layer by adjusting and scaling the activations. For example, when we have features from 0 to 1 and some from 1 to 1000, we should normalize them to speed up learning. Batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers. To increase the stability of a neural network, batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.

Weight Initialization or Kernel Initializer:

The neural network needs to start with some weights and then iteratively update them to better values. The term kernel_initializer is a fancy term for which statistical distribution or function to use for initialising the weights. When weights are small the activation is less precise and when weights are large the activation gets saturated. Hence choose the weights randomly near zero.

Glorat Kernel Initializer:

Basically it tries to make sure the distribution of the inputs to each activation function is zero mean and unit variance. To do this, it assumes that the input data has been normalized to the same distribution. The more number of inputs a neuron has, the smaller the initial weights should be, in order to compensate for the number of inputs. In a word, the Xavier initialization method tries to initialize weights with a smarter value, such that neurons won't start training in saturation.

$$\mathrm{Var}(W) = \frac{2}{n_{\text{in}} + n_{\text{out}}}$$

It's initializing the weights in your network by drawing them from a distribution with zero mean and a specific variance, where W is the initialization distribution for the neuron in question, and
n(in) is the number of neurons feeding into it. The distribution used is typically Gaussian or uniform.
n(out) is the number of neurons the result is fed to.

Regularization Methods:

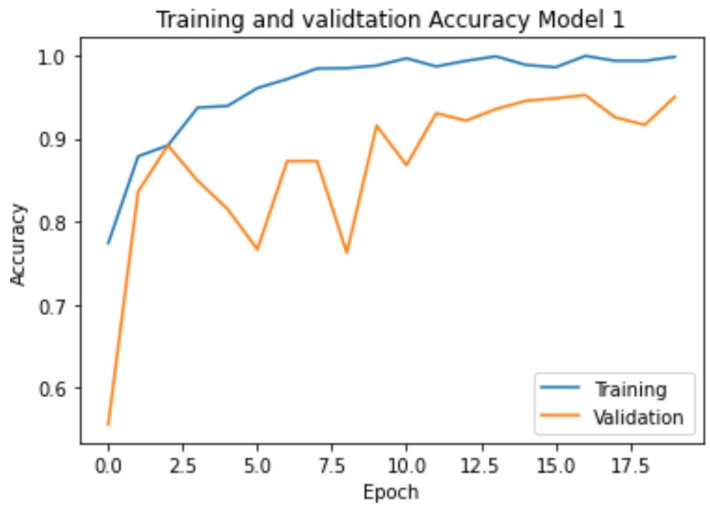
Large numbers of parameters tend to overfit. Need to prevent overfitting

Dropouts:

Dropouts help to avoid co-adaptation. At each fully connected stage dropouts are units with probability of (1-p), where p is a hyperparameter. We have dropped out by the probability of 0.3.

Evaluation Metrics :Accuracy: This metric signifies how many data points are predicted correctly. Accuracy is the proportion of true results among the total number of cases examined.

ACCURACY = (TP + TN) / (TP + FP + TN + FN)

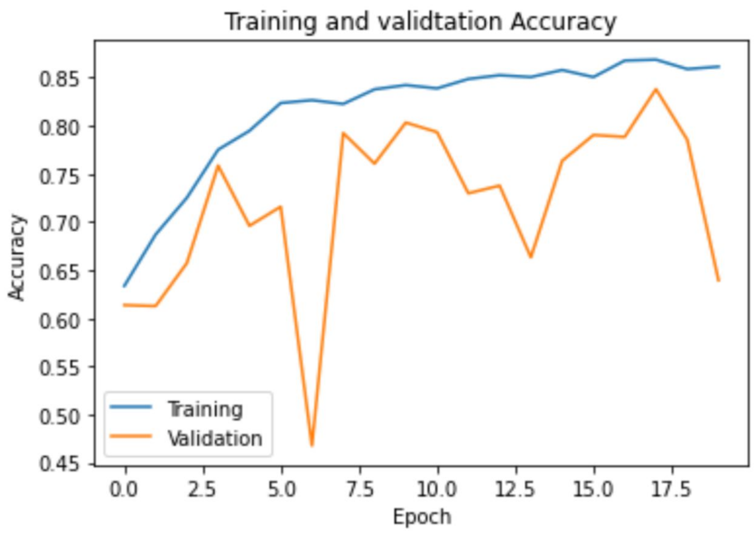


DATA AUGMENTATION

This step is generally done in data preprocessing where we make changes to existing features like flip, rotation, skew, shear, etc to generate new samples of the same image to get different points of views. We have done data augmentation using ImageDataGenerator in keras where it augments the data randomly in the runtime, while the model is being trained.

We tried implementing data augmentation in hope that the model will generalize better. In the above results we can see that even though the model performs good on validation data, it seems to overfit the training data. Hence anticipating better results using different views of the image, we used data augmentation.

We have defined a data generator which randomly flips the image horizontally and vertically and also rotates it by the scale of 90. Following are the results of the model which shows the validation and training accuracy.



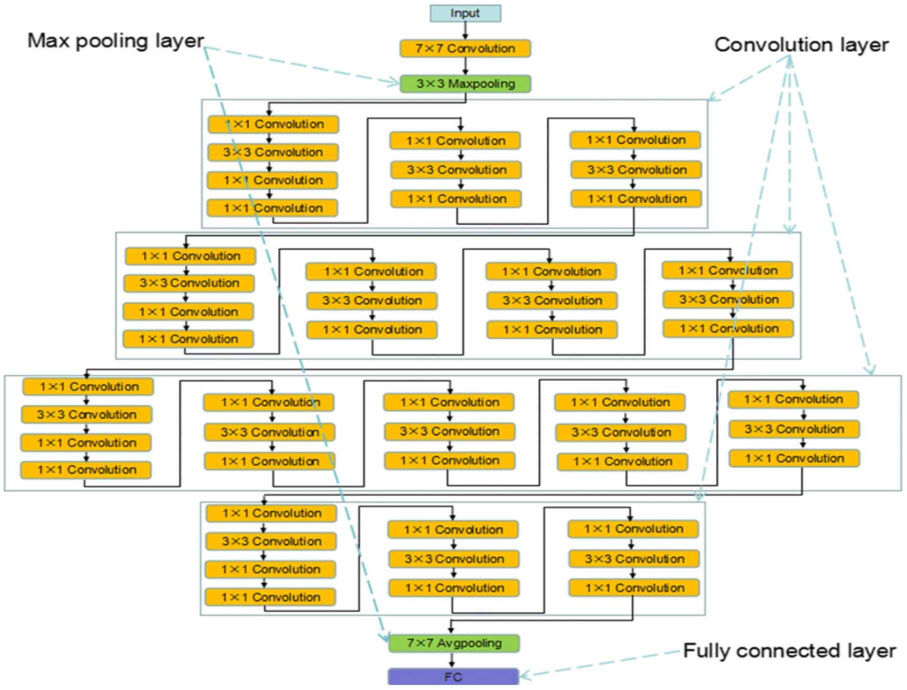
After taking a look at the results, it can be observed that the model performance was degraded. We were anticipating a more robust model but it turned out to be poor performing. This made us curious to dive more deeper in the process of MR Imaging. When we read more about how the MR Images are taken, we can conclude that if we randomly make changes to the image, we are changing the perspective of the model. The MRIs are taken from a certain angle and have specific interpretations. So this step won't help improve the performance, instead it will degrade the model.

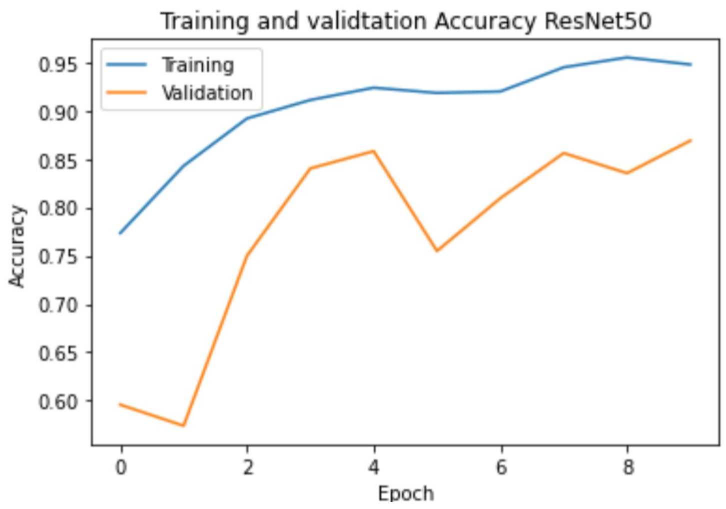
TRANSFER LEARNING

The networks that we implemented above had few layers and if we wish to build deeper neural networks, it would take a lot of time to train and find the optimal weights and biases. One step to cut this process short and make it more efficient is transfer learning. In this approach, we try to use the weights of already trained models and just change the input and output layers to make it work for our problem. One such network for image classification is ResNet 50 (Residual Network).

- **ResNet50**

A residual neural network is an artificial neural network of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers. We load the pretrained ResNet model using imagenet weights and discard the output layers. We also changed the input shape parameter to (512,512,3). Also we have converted our training and testing features to 3-channeled images as ResNet works only on RGB. Also, we freeze most of the layers to reduce the training time. If not done, the notebook crashes as it can't handle such large parameter computations. Following is the architecture of ResNet 50 which shows how each layer is connected to the other.





RESULTS AND INFERENCES:

Now that we have gone through the approaches, here are the results of each approach. We evaluated the models based on accuracy. Hence, the table below shows the comparison of each model. We compare the training and validation loss, accuracy and training time for each model and note each value is the best in all the epochs.

Metric	**Model 1**	**With Data Augmentation**	**ResNet 50**
Training Accuracy	0.995	0.8688	0.9549
Training Loss	0.0048	0.4140	0.4828
Validation Accuracy	0.9523	0.8381	0.8689
Validation Loss	0.1754	0.4877	1.1077
No. of Epochs	20	20	10

- From the results we can see that the model which we define performs better than other approaches that we implement. Other models are more complex as compared to Model1.
- The model that uses augmented data fails as the classifier also depends on the orientation of the images and changing the images won't help get better performance of the model.
- The approach of transfer learning using ResNet 50 architecture surely gave better results. We freezed most of the layers to keep the trainable parameters less, but still it takes longer to run. And being a more complex and deeper model, it tends to overfit the data. We hope this would be different if we let all the layers train, but we didn't have the resources to do so at the present time. In future, we plan to train the whole ResNet model allowing all the layers to update their values.

The final approach that we would go with is the simpler model, Model 1. This model gives better results and does not overfit the training data as much as the other two models do.

REFERENCES

[1] BRAIN TUMOR SEGMENTATION USING DEEP LEARNING BY TYPE SPECIFIC SORTING OF IMAGES
[2] COMPUTER VISION BASED BRAIN TUMOR CLASSIFICATION AND IDENTIFICATION IN CT SCANNING

[3]https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253

[4]https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a

[5]https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy

[6]https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

[7]https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253

[8]https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b

[9]http://www.cs.iit.edu/~agam/cs512/share/dlearc.pdf

[10]http://www.cs.iit.edu/~agam/cs512/share/dnnkeras.pdf

[11]http://www.cs.iit.edu/~agam/cs512/share/cnna.pdf

[12]http://www.cs.iit.edu/~agam/cs512/share/cnnb.pdf

Link for the datasets

- **Original Data**

https://drive.google.com/drive/folders/1q3_9SgW8VqUEijLaEDrGj90QE_SUnnU_?usp=sharing

- **Preprocessed Data**

https://drive.google.com/file/d/1X96IRHzIEm6ICd9YY-09y77-SfxOfl03/view?usp=sharing