Brain Tumor Detection

Authors:- Anonymous 1 and Anonymous 2.

Abstract

In this project, we have worked on tumor detection in brain MRI images. We have experimented with different models to achieve better results. We have achieved an accuracy of 89% on our test data. We have also explored pruning as an experiment on how it effects our model and accuracy for classification. We have also tried using ResNet to obtain good results. We have also experimented with UNets and added a classification head to its architecture as an experiment to see if we can get any reliable results. We will discuss the use of few metrics and their need in the end of this paper.

Introduction

Brain Tumor detection has gathered a lot of interest in the computer vision community and medical image processing lately. This has particularly helped doctors and surgeons to detect tumors that might have been overseen. The dataset is taken from Kaggle. The link to the dataset is https://www.kaggle.com/mateuszbuda/lgg-mri-segmentation. The data contains thousands of brain MRI scan images. We have explored various models like ResNets, UNets and we have developed our own CNN model. We are looking for good accuracy and have used evaluation metrics like f1 score, ROCAUC for our test data. Figure 1 shows sample images from our dataset. Figure 2 illustrates the distribution of instances among the two classes.

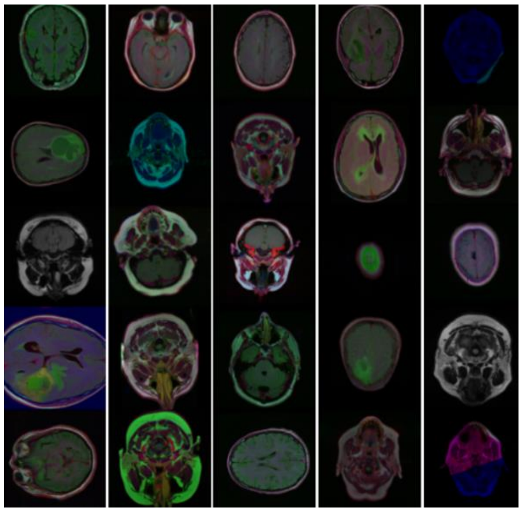
Figure 1

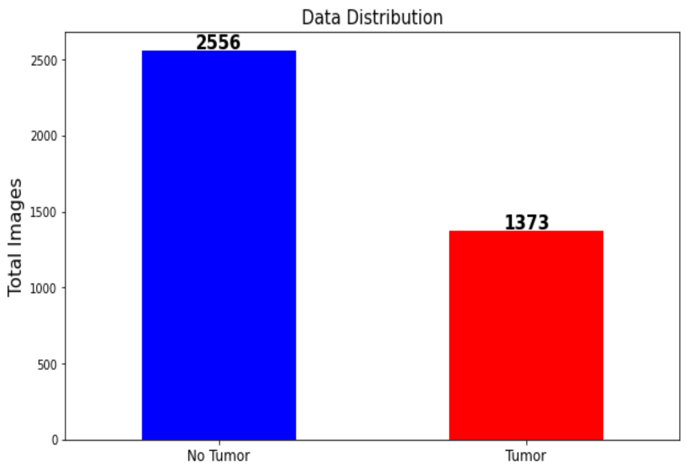

Figure 2

Pruning has significantly helped in reducing the size of deep neural architectures. We have leveraged pruning in our CNN model to make it light weight at the same time improving the performance. We will explore all its effects in the later part of this paper. We will first explain the models and their related concepts in the background/Related work section. We will then explain the approach and architecture of each model in the approach section. The experimental results of all our models will be shown in the experimental section. We will end the paper with all our learnings and takeaways in the conclusion followed by references.

Background/Related Work

We have tried using ResNet [2] for our project and get good results. We have used resnet with only 3 layers. We have not extended our experiment on ResNets as our dataset contains images that are very similar to each other (that is the complexity of data to be learned is very low). ResNet also takes too much time for training and experiments on this model is tedious. Nonetheless, we have shown our result with ResNet in the experiment session.

When we were researching on Tumor Detection, we came to know that the location of the brain tumor will affect the function of the nervous system. Hence, we have not limited our work on just Brain Tumor detection but has extended it to segmentation which tells in which location of the brain the tumor is located. In a traditional UNet [6], we have a contracting path which has general convolution layers and an expansive path which has a transpose2d layers. We have taken the UNet model and have modified it to perform both the classification and segmentation tasks.

Convolution Neural Networks (CNN) have been very useful in a wide variety of tasks [1]. The most basic uses of CNNs are classification problems, where some data is provided with the class in which it belongs, and we need to learn the data and predict the class of new unseen data. We are trying to experiment on different approaches for classification and the effect of various changes to our model. Although our project is on just 2 classes (tumor or no tumor), it can be extended to multi-class classification scenario like we have seen for ImageNet dataset.

We can use very big CNN with many layers and train it for many epochs to get very good results. This will inevitably increase the forward pass time as the number of layers increase. We call this delay between the time of providing the input and the result of the CNN as "latency". We can decrease this latency by a method called pruning [3] where we zero out few weights and biases from our CNN model. This is like dropout used for regularization in our CNN but in case of dropout we randomly drop weights each iteration during training. In case of pruning, we remove the connections with lower weights permanently of a trained model. This of course results in accuracy loss as a lot of information is lost when we miss out few connections. For this reason, after pruning, we retrain the model so that the other weights can learn and compensate the absence of the lost weights. We will explore the performance of this technique further in the paper.

The metrics used in this experiment are accuracy, f1 score, ROCAUC. The reason we are not just considering accuracy is that in case of 2 class classification model. We can easily get an accuracy of at least 50% by predicting all the examples as tumor or non-tumor. Hence, it is not a good metric for a 2-class problem. To get an estimate of how good the performance really is, we need to use the f1 score. F1 score can also be computed in case of a multi-class model. In the multi-class and multi-label case, this is the average of the F1 score of each class.

Approach

We do not have a dedicated test data, so we have split our data into train, validation, and test data. The images in the dataset are all different in size, resized all the input images to 3X128X128. We used albumentation library that helps in preprocessing of the data and transforms like Horizontal flip, vertical flip and Random Rotate. [7]

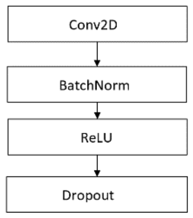

Figure 3

CNN

The architecture of CNN is shown in the Figure 3 and Figure 4. We use techniques like BatchNorm[4] and Dropout[5] following the convolutional layers in our network. BatchNorm makes sure that the values of weights do not explode or converge to zero as the depth of the network increases. Dropout helps in regularization and avoids overfitting. For this model, we have incorporated pruning where weights less than a threshold are ignored. Initially, we expected a drop in accuracy with the pruned model. Conversely, our experiments reveal a boost in the accuracy after pruning. We will get almost the same accuracy that we get from the unpruned model. We will also save some time in forward pass and retraining will be significantly faster as we will backpropagate through fewer nodes and weights.

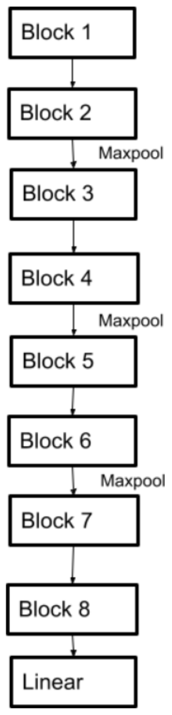

Figure 4

Resnet

We used a ResNet3 architecture, that has 3 residual blocks which is shown in Figure 6 and Figure 7. We used these residual blocks and an average pooling layer followed by linear layer. Unlike the ImageNet dataset, our dataset has only two classes, and the structure of brain is almost similar across all the images. Hence, our dataset is lesser complexity. We do not need deeper architectures like ResNet18 or higher as we need not learn a very complex target function. The model only needs to learn the presence of a tumor, which is relatively easy compared to learning complex features. We have built our own custom CNN and modified UNet models. However, we find that ResNet3 gives the best results on test set.

UNets

We have used the UNet model to perform image segmentation. As illustrated in Figure 5, we have added a classification head (Z) to the our model to handle two disparate objectives simultaneously. You can see the model architecture in Figure 5. Y is the predicted segmentation mask and X is the input image. We have added both the segmentation and classification losses to compute the final loss for our learning objective. This final loss is backpropagated to update model weights in every iteration.

$$L^{net} = \Theta L^{y} + (1 - \Theta) L^{z}$$

$$L^{y} = \text{Segmentation Loss}$$

$$L^{z} = \text{Classification Loss}$$

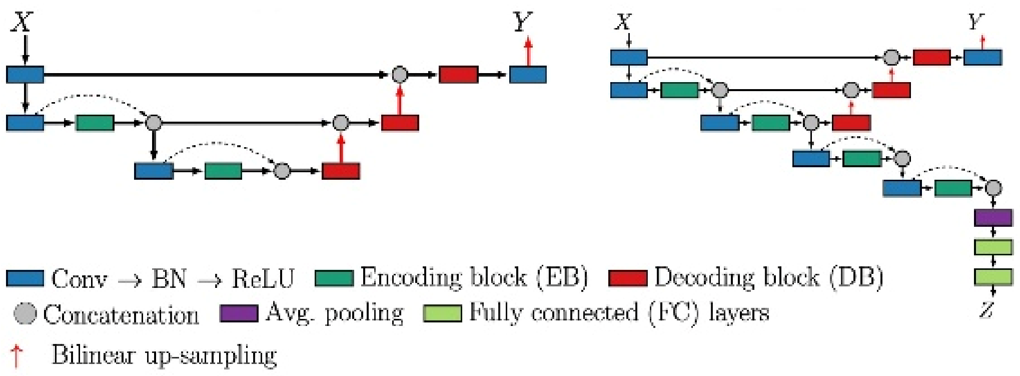

Figure 5 [8]

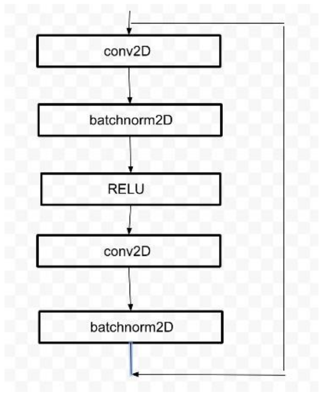

Figure 6

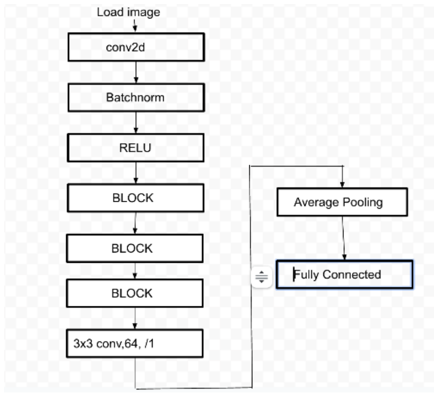

Figure 7

Experiments

We used a batch size of 32 while implementing our CNN model. We have searched extensively for hyper-parameters to fine tune our models. After much experimentation, we found that using Adam Optimizer along with a learning rate of 10^{-3} and momentum of 0.9 gives the best results. Our models are initialized using Gaussian Normal distribution with zero mean and standard deviation of 0.1. We also experimented with Xavier uniform distribution for our ResNet3 model. Initially, we have used 3-layer CNN for the milestone and achieved an accuracy of 74% on validation and 71% on the test data. We later tried 4-layer CNN to see get about 81% on validation and 79% on test data. Aiming to get a test accuracy of over 85%, we used 9-layer CNN and achieved an accuracy of 94% on validation and 89% on test data. We decided to keep the 9-layer CNN and stopped enhancing the model. We then pruned the model to reduce its size while preserving the accuracy metric. We implemented one shot pruning with different sparsity ratios 30%, 50%, 70% and 90%. And, for iterative pruning, we used sparsity ratios 50%, 75% and 90%. The details of pruning and their results are explained further below.

	RESNET	UNET	CNN
EPOCH1	69.92	65.88	67.84
EPOCH10	85.80	88.91	79.73
EPOCH20	92.37	91.34	89.31
EPOCH30	95.17	83.12	92.11

We later performed an experiment observing the validation accuracies for different models like ResNet, UNet and CNN. From the values shown in the table above we can clearly see that the validation accuracy is increasing for each epoch for ResNet and CNN models. But when we consider the case of UNet, the accuracy was increasing till 20 epochs, and began to drop substantially after 20 epochs.

In One Short pruning we remove a particular percentage of weights in one stretch from the model and train the pruned model and test it. In iterative pruning, the model is pruned and trained iteratively. This approach is considered because sometimes if we prune a model too much at once the network might lose important features that are crucial for our classification task. So generally iterative model is used in practice.

In our model the Figure 8 are the results of one short pruning for sparsity ratio 30%,50%,70% and 90% and the Figure 9 are the results of iterative pruning with sparsity ratio 50%, 75% and 90%. We can clearly see from the below graphs that the results of one-shot pruning, and iterative pruning are similar which is good. Moreover, the accuracy of the pruned model is like the accuracy of the unpruned model. This is because as we increase the sparsity ratio, we are removing weights which causes the model not to evaluate certain features hence we can see that there is a decrease in accuracy as we go beyond 70 % accuracy.

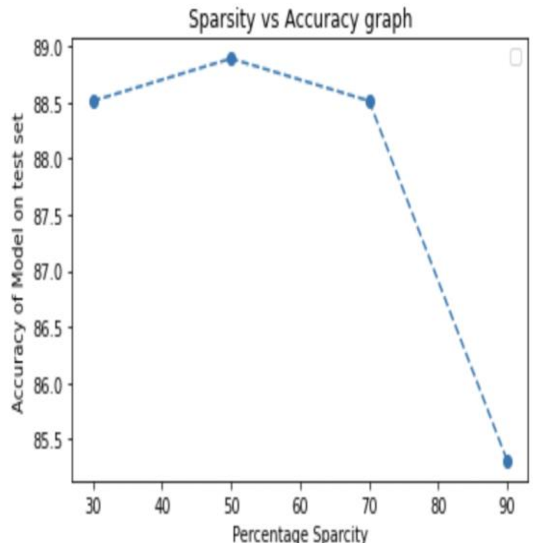

Figure 8

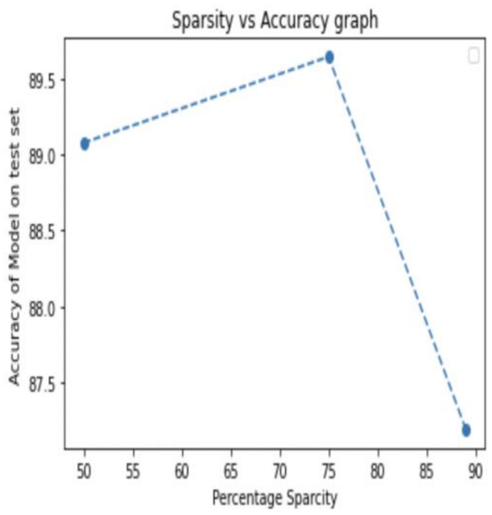

Figure 9

The inference time in this experiment is calculated to fetch one image. From Figure 10, we can clearly see that as the sparsity ratio increases the inference time decreases. This is because when you are pruning a model you are basically converting the parameters in the model to zero parameters. Hence the number of computations decreases as you increase the sparsity ratio. As the number of computations decreases the inference time decreases. That is the reason the inference time is decreasing with the increase of sparsity ratio.

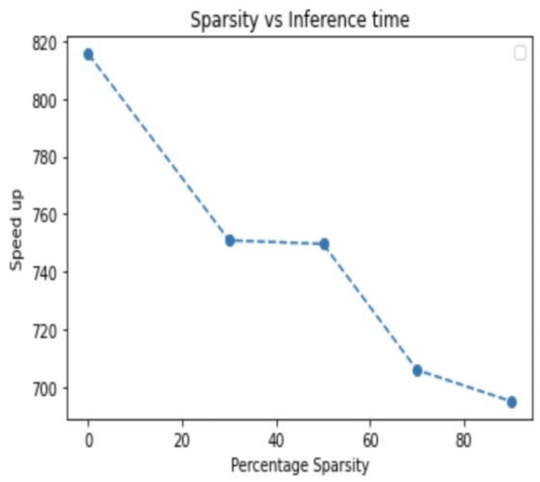

Figure 10

The table shown below tells us the test accuracies of the model after pruning with sparsity ratios 30%, 50%, 70% and 90% and their accuracies before and after training of the pruned model.

	Sparsity30	Sparsity 50	Sparsity 70	Sparsity 90
Unpruned model	85.87	85.87	85.87	85.87
Pruned model (before training)	83.99	79.09	36.53	35.03
Pruned model (after training)	88.5	88.89	88.51	85.31

The table below shows the f1 score, recall, precision and ROCAUC for the CNN model we used for the classification.

	f1 score	recall score	precision	ROCAUC
epoch1	0.73	0.78	0.68	0.79
epoch10	0.75	0.61	0.99	0.80
epoch20	0.88	0.83	0.93	0.90
epoch30	0.93	0.89	0.97	0.94

Conclusion

Metrics

Our problem is a binary classification problem. For such datasets the accuracy is not a good measure of performance as we can easily get over 50% by predicting any class for all the images. For this reason, we use the metrics f1 score, recall and precision. ROCAUC is basically used for classification problems. The ROC curve summarizes the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds. The AUC function takes both the true outcomes (0,1) from the test set and the predicted probabilities for the 1 class.

Pruning

We can clearly see that pruning has almost no effect on validation and testing accuracy (after we retrain the model). But we can see a significant speedup in test time. This test time does not even show the total performance enhancements of this method. As the pytorch library stores the removed weights using zeros instead of not performing computations with them, that is there is an actual multiplication happening here which can be ignored. Instead, if we can store the weights as a spare matrix, we can do only the necessary computation which will result in slightly faster test times. Now, let us justify why pruning has no effect on our model. The presence or absence of tumor in the data depends on a very small region of the MRI images. Hence, we can ignore most of the weights which have little to no effect on predicting the tumor. We can also see that sometimes there is a small increase in test accuracy when we prune the model. This can be because pruning adds some regularization to the model like drop- out. But when we prune it extremely like 90% then we lose some information that is valuable for the classification. Overall, this method is very useful when we have very time sensitive computations. For example, tesla uses image recognition to analyze its environment and saving a few milliseconds can significantly improve its safety and the drive experience.

UNets

We see that the performance of the UNets for doing both the classification and segmentation is very poor compared to the one with only segmentation. This is because the gradients propagating back from the classification head and the segmentation head are adding up once they reach the common layers of the model. This will change the direction of the step taken by our weights to neither improve our classification nor the segmentation. This phenomenon is shown in figure 11. Initially our classification head is performing very well as we reach a local minimum. Soon after this our model will try to improve the segmentation of the model. After many epochs, we will reach a stability where both the heads perform weaker than if the losses are backpropagated individually. We have backpropagated the losses individually we get result we expect that are very good.

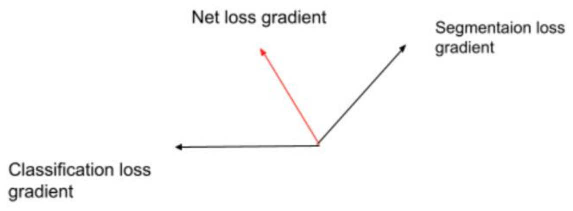

Figure 11

References

1) AlexNet-https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.
2) ResNet- https://arxiv.org/pdf/1512.03385.pdf.
3) Pruning - Han, Song, Pool, Jeff, Tran, John, and Dally, William J. Learning both weights and connections for efficient neural networks. In Advances in Neural Information Processing Systems, 2015.
4) BatchNorm - https://arxiv.org/pdf/1502.03167.pdf
5) Dropout - https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf
6) UNets - https://arxiv.org/pdf/1505.04597.pdf

7) Preprocessing - https://www.kaggle.com/lqdisme/brain-mri-segmentation-unet-pytorch#7.-Train-History
8) Y-Nets - https://arxiv.org/pdf/1806.01313.pdf.