CS6350.002 Final Project Report Detection and Localization of Brain Tumor from MRI Scan Images. *

Mahsa Eslamialishah*, Pritom Das Radheshyam† and Novonil Das‡
The University of Texas at Dallas
*mxe190002@utdallas.edu, †pxr180033@utdallas.edu, ‡nxd180022@utdallas.edu

***Abstract*—In recent years there has been growing popularity of Deep Learning in almost every fields where decision making is involved - finance, health care, marketing, sales and what not. Deep Learning has shown promising results in the field of healthcare in many areas such as: Disease Diagnosis with Medical Imaging, Surgical Robots, Maximizing Hospital Efficiency. AI healthcare market is expected to reach $45.2 billion USD by 2026 from the current valuation of $4.9 billion USD. Deep learning has been proven to be superior in detecting diseases from X-rays, MRI scans and CT scans which could significantly improve the speed and accuracy of diagnosis.**

In this CS6350.002 final project we have applied Deep Learning to detect brain tumours from MRI Scan images using Residual Network and Convoluted Neural Networks. This automatic detection of brain tumors can improve the speed and accuracy of detecting and localizing brain tumors based on MRI scans. This would drastically reduce the cost of cancer diagnosis and help in early detection of tumors without any human involvement and would essentially be a life saver. We have also compared the accuracy of results obtained by using two different models - ResNet50 and ResNet18 and used Transfer Learning to tune or freeze weights to evaluate what gives best result.

We have 3929 brain MRI scans which are either positive or negative cases of brain tumour. We have built models using ResNet50 and ResNet18 and evaluated their performance in detecting positive or negative cases of brain tumors.

***Index Terms*—Brain Tumor Detection, Deep Learning, Residual Network, Convoluted Neural Network**

I. Introduction

II. Related Work

A. Image Segmentation

Image Segmentation is the process of classifying each pixel in the image as belonging to a specific category. Image Segmentation allows us to understand and extract information from images at the pixel-level. Image Segmentation can be used for object recognition and localization which offers tremendous value in many applications such as medical imaging and self-driving cars etc. We could use image segmentation to train a neural network to produce pixel-wise mask of the image. Modern image segmentation techniques are based in deep learning approach which makes use of common architectures such as CNN, FCNs(Fully Convolution Networks) and Deep Encoders-Decoders.

B. Convolutional Neural Network

Convolutional Neural Network (CNN) is a type of deep neural network which has proven to perform well in computer vision tasks such as image classification, object detection, object localization and neural style transfer. CNNs are able to extract the features of an image, which an algorithm like a multi-layer perceptron (MLP) or a recursive neural network (RNN) does not have the ability to do. The architecture of a convolutional neural network looks something like this:

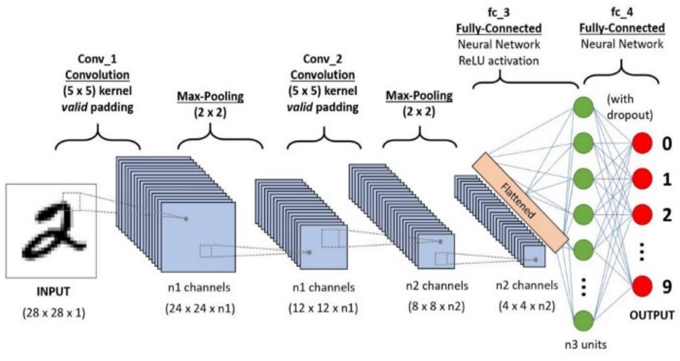


Fig. 1. Convolutional Neural Network

We should note a couple of things from this. The network starts off with 2 convolutional and max-pooling layers, followed with 2 fully connected layers which end with an output layer. The convolution +max pooling layers is where the images are recognized (features are extracted), and the fully connected layers are where the images are classified into predefined classes.

C. Residual Network

A residual neural network (ResNet) 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. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between.

In traditional neural networks, more layers mean a better network but because of the vanishing gradient problem,

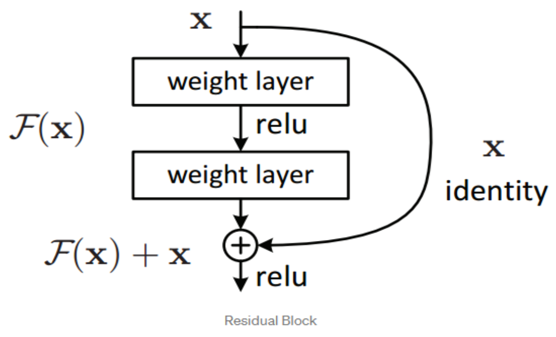


Fig. 2. Residual Network

weights of the first layer won't be updated correctly through the back-propagation. As the error gradient is back-propagated to earlier layers, repeated multiplication makes the gradient small. Thus, with more layers in the networks, its performance gets saturated and starts decreasing rapidly. Res-Net solves this problem by using the identity matrix. When the back-propagation is done through identity function, the gradient will be multiplied only by 1. This preserves the input and avoids any loss in the information.

Components of a network include 3X3 filters, CNN down-sampling layers with stride 2, global average pooling layer and a 1000-way fully-connected layer with softmax in the end. ResNet uses a skip connection in which an original input is also added to the output of the convolution block. This helps in solving the problem of vanishing gradient by allowing an alternative path for the gradient to flow through. Also, they use identity function which helps higher layer to perform as good as a lower layer, and not worse.

In traditional neural networks, each layer feeds into the next layer. But in a network with residual blocks, each layer feeds into the next layer and directly into the layers about some hops away. Consider a neural network block, whose input is x and we would like to learn the true distribution H(x). The residual between the output and input can be denoted as R(x) = Output - Input = H(x) - x. The layers in a traditional network learn the true output (H(x)) whereas the layers in a residual network learn the residual (R(x)).

D. Transfer Learning

A major assumption in many machine learning and data mining algorithms is that the training and future data must be in the same feature space and have the same distribution. However, in many real-world applications, this assumption may not hold. For example, we sometimes have a classification task in one domain of interest, but we only have sufficient training data in another domain of interest, where the latter data may be in a different feature space or follow a different data distribution. In such cases, knowledge transfer, if done successfully, would greatly improve the performance of learning by avoiding much expensive data-labeling efforts. Humans have an inherent ability to transfer knowledge across tasks. What we acquire as knowledge while learning about one task, we utilize in the same way to solve related tasks. Transfer Learning is a machine Learning technique in which a network that has been trained to perform specific task is being reused as a starting point of another similar task. Transfer Learning is widely used since starting from a pre-trained model can dramatically reduce the computational time required if training is performed from scratch. The more related the tasks, the easier it is for us to transfer, or cross-utilize our knowledge. Some simple examples would be,

Know how to ride a motorbike - Learn how to ride a car.

Know how to play classic piano - Learn how to play jazz piano.

Know math and statistics - Learn machine learning.

In each of the above scenarios, we don't learn everything from scratch when we attempt to learn new aspects or topics. We transfer and leverage our knowledge from what we have learnt in the past! Conventional machine learning and deep learning algorithms, so far, have been traditionally designed to work in isolation. These algorithms are trained to solve specific tasks. The models have to be rebuilt from scratch once the feature-space distribution changes. Transfer learning is the idea of overcoming the isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones. Traditional way of machine learning and deep learning approach thus far have been implemented to work in an isolated manner and solve only specific kind of tasks. Every time the models needed to be rebuilt from scratch once the feature-space distribution changes. Transfer learning is a novel approach which helps to overcome the issues of overcoming isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones [1].

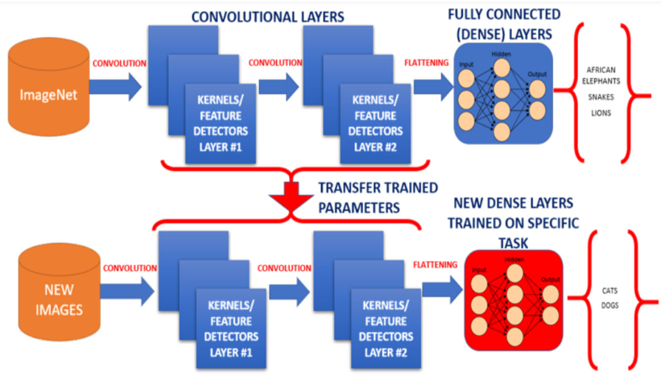

CONVOLUTIONAL LAYERS
FULLY CONNECTED (DENSE) LAYERS
ImageNet
CONVOLUTION
KERNELS/ FEATURE DETECTORS LAYER #1
CONVOLUTION
KERNELS/ FEATURE DETECTORS LAYER #2
FLATTENING
Input
Hidden
Output
AFRICAN ELEPHANTS
SNAKES
LIONS
TRANSFER TRAINED PARAMETERS
NEW DENSE LAYERS TRAINED ON SPECIFIC TASK
NEW IMAGES
CONVOLUTION
KERNELS/ FEATURE DETECTORS LAYER #1
CONVOLUTION
KERNELS/ FEATURE DETECTORS LAYER #2
FLATTENING
Input
Hidden
Output
CATS
DOGS

Fig. 3. Transfer Learning

Thus, the key motivation, especially considering the context of deep learning is the fact that most models which solve complex problems need a whole lot of data, and getting vast amounts of labeled data for supervised models can be really difficult, considering the time and effort it takes to label data points.

III. Data Formatting

We have worked on 3929 Brain MRI Scans. These are image files that contain the actual data of patients brain MRI scans and the information if they were analyzed by the doctor to have tumors or not. The data was converted into .png format and we ran the machine learning algorithm in all those images. We have built two different models - one using ResNet50 and the other using ResNet18, and also tuned certain hyper parameters to assess and compare the results of all the approaches. We used 85% of the data to train the model and the rest 15% of the data was used for testing.

IV. Proposed Method

We used Brain MRI Scans images as dataset to train our machine and test our machine learning model. We used multiple features to run our model and using Residual Networks we classified if the MRI scans would be classified to have tumors or not. We used 2 different models (ResNets) - ResNet50 and ResNet18 to classify the MRI scans to either contain tumors or not and tuned multiple hyper parameters in transfer learning - a. Freeze the trained CNN network weights from the first layers, only train the newly added dense layers(with randomly initialized weights). b. Initialize the CNN network with the pre-trained weights and retrain the entire CNN network while setting the learning rate to be very small, this is critical to ensure that you do not aggressively change the trained weights.

A. Image Segmentation

The image is converted into a vector and possibly a classification head is added at the end. Classical convolutional neural networks are generally used when the entire image needs to be classified as a class label. Softmax function is applied to every pixel which makes the segmentation problem work as a classification problem where classification is performed on every pixel of the image.

B. Mask

The goal of the image segmentation is to understand the image at eh pixel level. It associates each pixel with a certain class. The output produce by image segmentation model is called a ”mask” of the image.

Masks can be represented by associating pixel values with their coordinates. For example if we have a black image of shape (2,2), this can be represented as [[0,0],[0,0], If the output mask is as follows [[255,0],[0,255]]. To represent this mask we have to first flatten the image into a 1-D array. This would result in something like [255,0,0,255] for the mask. The, we can use the index to create the mask. Finally we would have something like [1,0,0,1] as our mask.

C. Convolutional Neural Networks

Neural networks are a powerful technology for classification of visual inputs arising from documents. The first CNN layers are used to extract high level general features. The last couple of layers are used to perform classification (on a specific task). Local respective fields scan the image first searching for simple shapes such as edges/lines. These edges are then picked up by the subsequent layer to form more complex features.

D. ResNet(Residual Neural Network)

Convolutional Neural Network (CNN) based models have achieved great success in Single Image Super-Resolution (SISR). Owing to the strength of deep networks, these CNN models learn an effective nonlinear mapping from the low-resolution input image to the high-resolution target image, at the cost of requiring enormous parameters. As convolutional neural networks grow deeper, vanishing gradient tend to occur which negatively impact network performance. Vanishing gradient descent problem occurs when the gradient is back-propagated to earlier layers which result in a very small gradient. Residual Neural Network includes ”skip connection” feature which enables training of 152 layers without vanishing gradient issues. ResNet works by adding ”identity mappings” on top of the CNN. ImageNet contains 11 million images and 11000 categories. ImageNet is used to train ResNet deep network.

E. ResNet50

ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [1]. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224. This can be used to classify new images using the ResNet-50 model.

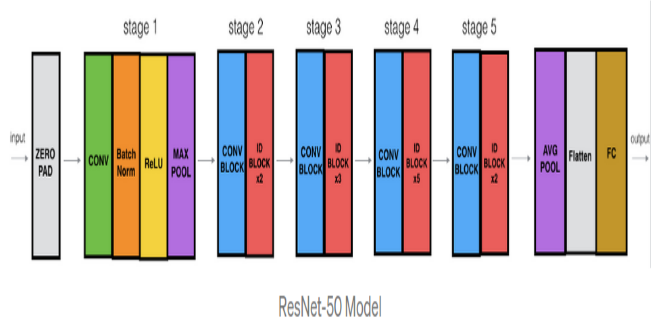


Fig. 4. ResNet50

F. ResNet18

ResNet-18 is a convolutional neural network that is 18 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [1]. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224. This can be used to classify new images using the ResNet-50 model.

G. Transfer Learning

Transfer Learning is a machine learning technique in which a network that has been trained to perform a specific task is being reused or re-purposed as a starting point for another similar task. Transfer Learning is widely used since starting from a pre-trained model can dramatically reduce the computational time required if training is performed from scratch.

1) Transfer Learning Training Strategies:

- Strategy 1 Steps:
  - Freeze the trained CNN network weights from the first layers.
  - Only train the newly added dense layers(with randomly initialized weights).
- Strategy 2 Steps:
  - Initialize the CNN network with the pre-trained weights.
  - Retrain the entire CNN network while setting the learning rate to be very small, this is critical to ensure that you do not aggressively change the trained weights.
- Transfer Learning Advantages are:
  - Provides fast training progress, you don't have to start from scratch using randomly initialized weights.
  - You can use small training data set to achieve incredible results.

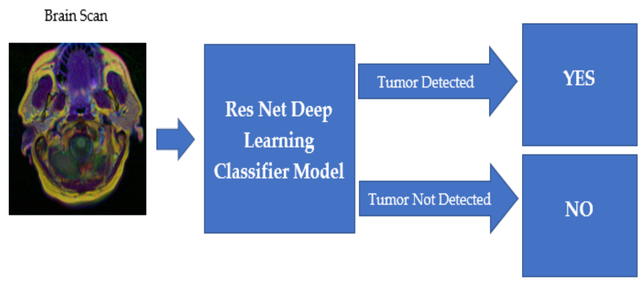


Fig. 5. Architecture

V. Experiments

In this study, we explore different approaches for data augmentation and transfer learning. For data augmentation, we have leveraged random horizontal flipping on the training data. For transfer learning, we have investigated leveraging the pre-training models as feature extractors and an initial model. In the feature extractor scenario, the pre-trained layers are frozen during training the new task and only the added classification layers are trained. In the initialization scenario, the entire model, including the ResNet layers, is trained to learn the new task.

Since our training data is small compared to the number of trainable parameters, we hypothesized that a small pre-trained model would perform better for the initialization scenario. Therefore, we examine the ResNet50 and ResNet18 as our pre-trained models.

A. Experiment 1

The images files contain brain MRI scans and those are used as inputs to the model to train and test the model and in this experiment the images are resized.

B. Experiment 2

The images files contain brain MRI scans and those are used as inputs to the model to train and test the model and in this experiment the images are resized and normalized.

C. Experiment 3

The images files contain brain MRI scans and those are used as inputs to the model to train and test the model and in this experiment the images are resized and random horizontal flip.

D. Experiment 4

The images files contain brain MRI scans and those are used as inputs to the model to train and test the model and in this experiment the images are resized and random horizontal flip and normalized.

E. Experiment 5

VI. Result

We conducted a series of experiments and the results of those experiments have been attached here at the end. The results were then compared, and we found the following - By comparing experiment 1 and 3, where we changed the model from ResNet50 to ResNet18, we show that we achieved 1% improvement. this improvement is because ResNet18 has less number of parameters and since our data is small, ResNet50 gets overfitted to the data. By tuning the Neural Network hyperparameters we achived 96% accuracy for ResNet18 and 94% accuracy for ResNet50. In the next experiments we have fixed these hyperparameters and normalized our data with mean = [0.485,0.456,0.406] and std= [0.229,0.224,0.225]. we show that our accuracy dropped from 96% to 92% for ResNet18 and improved by 1% for ResNet50. comparing experiment 13 and 15 shows that by adding random horizantal flipping and removing the normalization, the ResNet18 acieves a better performance. For the last experiment we have added random horizontal flipping and normalization. As the result of experiment 19 and 20 shows, we have achived our best performance for both ResNet50 and ResNet18. We achieved the best results for a train batch size of 8 and test batch size of 8 with 30 epochs and Learning rate of 0.0001 and momentum of 0.5, with added layers of Linear RELU Dropout Linear Dropout Linear, and a transform on the dataset with a resize factor of 224 and Random Horizontal Flip we achieved an accuracy of 97% using a ResNet18 model and an accuracy of 96% using ResNet50 model.

Table: 1 RESULTS

Transfer Learning using ResNet50 and ResNet18

Experiment Number	Parameters	Transfer Learning model	Added layers	Transforms On Dataset	Result
1	Train Batch size = 32 Test batch size = 32 Epochs = 20 Learning Rate = 0.01 Momentum = 0.5	ResNet18 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.45 Test average loss = 0.46 Test accuracy = 75%
2	Train Batch size = 32 Test batch size = 32 Epochs = 20 Learning Rate = 0.01 Momentum = 0.5	ResNet18 as fixed feature extractor	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.46 Test average loss = 0.43 Test accuracy = 81%
3	Train Batch size = 32 Test batch size = 32 Epochs = 20 Learning Rate = 0.01 Momentum = 0.5	ResNet50 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.46 Test average loss = 0.47 Test accuracy = 74%
4	Train Batch size = 32 Test batch size = 32 Epochs = 20 Learning Rate = 0.01 Momentum = 0.5	ResNet50 as fixed feature extractor	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.53 Test average loss = 0.47 Test accuracy = 65%
5	Train Batch size = 16 Test batch size = 16 Epochs = 20 Learning Rate = 0.01 Momentum = 0.5	ResNet18 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.65 Test average loss = 0.66 Test accuracy = 65%
6	Train Batch size = 16 Test batch size = 16 Epochs = 20 Learning Rate = 0.01 Momentum = 0.5	ResNet18 as fixed feature extractor	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.47 Test average loss = 0.46 Test accuracy = 73%
7	Train Batch size = 16 Test batch size = 16 Epochs = 20 Learning Rate = 0.01 Momentum = 0.5	ResNet50 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.47 Test average loss = 0.43 Test accuracy = 78%

8	Train Batch size = 16 Test batch size = 16 Epochs = 20 Learning Rate = 0.01 Momentum = 0.5	ResNet50 as fixed feature extractor	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.65 Test average loss = 0.66 Test accuracy = 65%
9	Train Batch size = 8 Test batch size = 8 Epochs = 20 Learning Rate = 0.001 Momentum = 0.5	ResNet18 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.07 Test average loss = 0.16 Test accuracy = 96%
10	Train Batch size = 8 Test batch size = 8 Epochs = 20 Learning Rate = 0.001 Momentum = 0.5	ResNet18 as fixed feature extractor	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.32 Test average loss = 0.25 Test accuracy = 90%
11	Train Batch size = 8 Test batch size = 8 Epochs = 20 Learning Rate = 0.001 Momentum = 0.5	ResNet50 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.19 Test average loss = 0.17 Test accuracy = 94%
12	Train Batch size = 8 Test batch size = 8 Epochs = 20 Learning Rate = 0.001 Momentum = 0.5	ResNet50 as fixed feature extractor	Linear ReLU Dropout linear Dropout Linear	Resize -> 224	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.35 Test average loss = 0.30 Test accuracy = 85%
13	Train Batch size = 8 Test batch size = 8 Epochs = 20 Learning Rate = 0.001 Momentum = 0.5	ResNet18 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224 Normalize(mean = [0.485,0.456,0.406] , std= [0.229,0.224,0.225])	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.07 Test average loss = 0.2 Test accuracy = 92%
14	Train Batch size = 8 Test batch size = 8 Epochs = 20 Learning Rate = 0.001 Momentum = 0.5	ResNet50 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224 Normalize(mean = [0.485,0.456,0.406] , std= [0.229,0.224,0.225])	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.17 Test average loss = 0.14 Test accuracy = 95%
15	Train Batch size = 8 Test batch size = 8 Epochs = 20 Learning Rate = 0.001 Momentum = 0.5	ResNet18 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224 Random Horizontal Flip	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.09 Test average loss = 0.14 Test accuracy = 96%

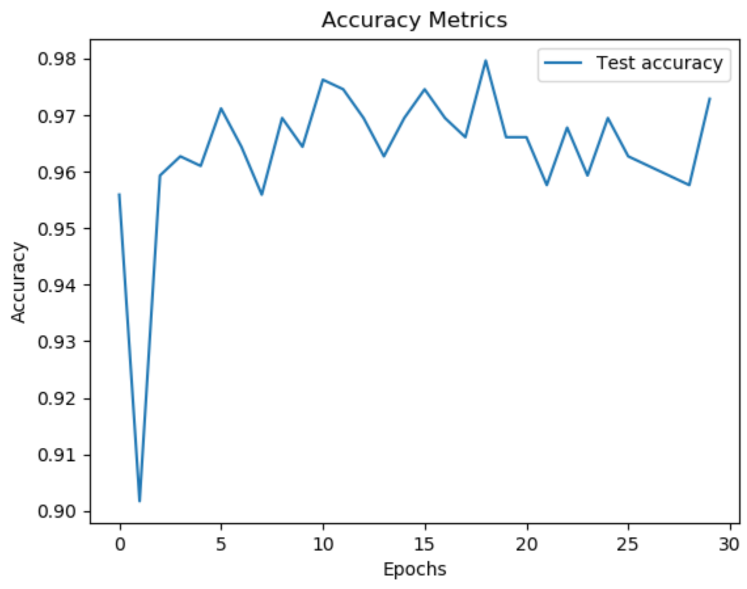
16	Train Batch size = 8 Test batch size = 8 Epochs = 20 Learning Rate = 0.001 Momentum = 0.5	ResNet50 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224 Random Horizontal Flip	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.16 Test average loss = 0.15 Test accuracy = 95%
17	Train Batch size = 8 Test batch size = 8 Epochs = 20 Learning Rate = 0.001 Momentum = 0.5	ResNet18 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224 Random Horizontal Flip Normalize(mean = [0.485,0.456,0.406] , std= [0.229,0.224,0.225])	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.10 Test average loss = 0.14 Test accuracy = 97%
18	Train Batch size = 8 Test batch size = 8 Epochs = 20 Learning Rate = 0.001 Momentum = 0.5	ResNet50 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224 Random Horizontal Flip Normalize(mean = [0.485,0.456,0.406] , std= [0.229,0.224,0.225])	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.16 Test average loss = 0.21 Test accuracy = 92%
19	Train Batch size = 8 Test batch size = 8 Epochs = 30 Learning Rate = 0.0001 Momentum = 0.5	ResNet18 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224 Random Horizontal Flip Normalize(mean = [0.485,0.456,0.406] , std= [0.229,0.224,0.225])	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.02 Test average loss = 0.14 Test accuracy = 97%
20	Train Batch size = 8 Test batch size = 8 Epochs = 30 Learning Rate = 0.0001 Momentum = 0.5	ResNet50 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224 Random Horizontal Flip Normalize(mean = [0.485,0.456,0.406] , std= [0.229,0.224,0.225])	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.013 Test average loss = 0.15 Test accuracy = 96%

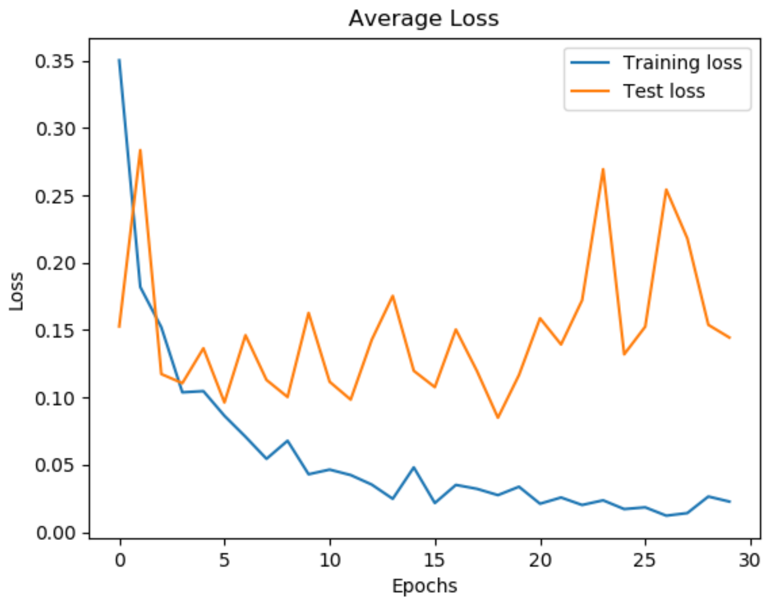
Table: 2 BEST RESULTS

Experiment Number	Parameters	Transfer Learning model	Added Layers	Transforms On Dataset	Result
19	Train Batch size = 8 Test batch size = 8 Epochs = 30 Learning Rate = 0.0001 Momentum = 0.5	ResNet18 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224 Random Horizontal Flip Normalize(mean = [0.485,0.456,0.406] , std= [0.229,0.224,0.225])	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.02 Test average loss = 0.14 Test accuracy = 97%

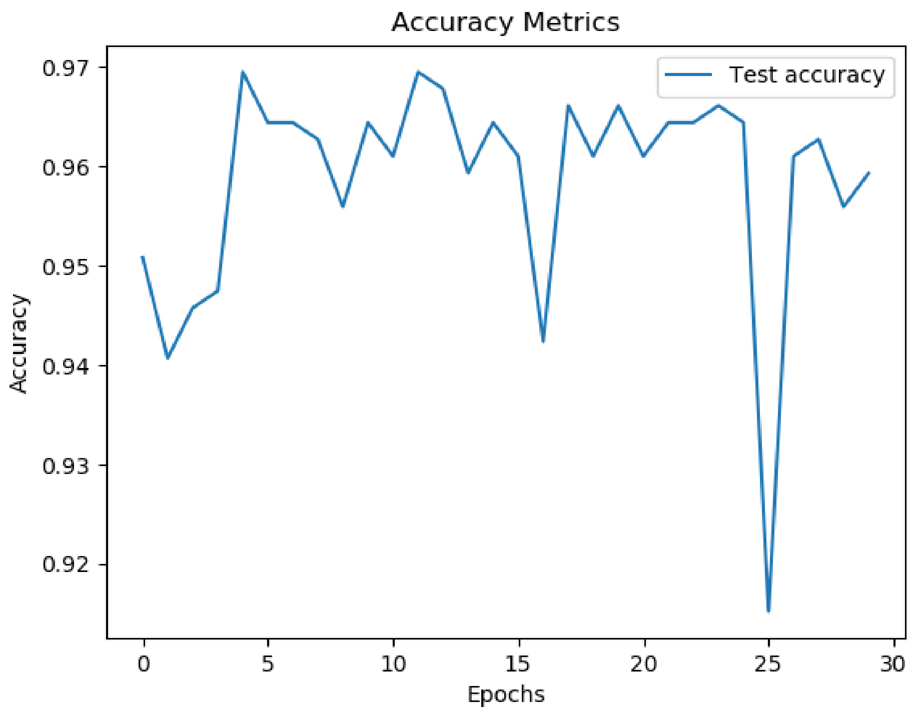
20	Train Batch size = 8 Test batch size = 8 Epochs = 30 Learning Rate = 0.0001 Momentum = 0.5	ResNet50 Fine tuning the convnet	Linear ReLU Dropout linear Dropout Linear	Resize -> 224 Random Horizontal Flip Normalize(mean = [0.485,0.456,0.406] , std= [0.229,0.224,0.225])	Train/Test split = 85:15 Size of dataset = 3,929 Train average loss = 0.013 Test average loss = 0.15 Test accuracy = 96%

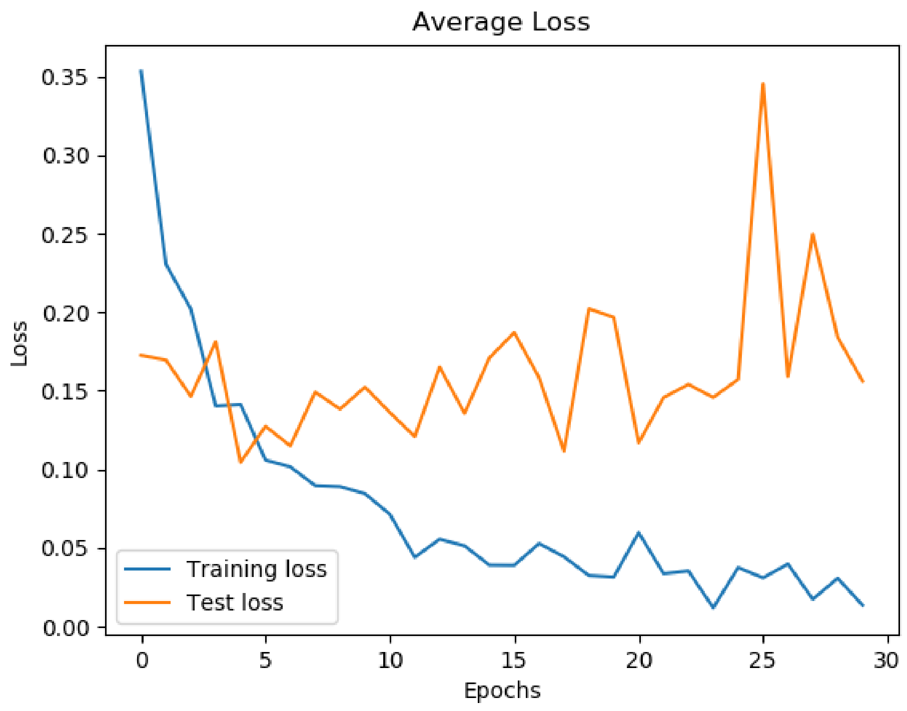
Experiment 19 plots:(ResNet18)





Experiment 20 plots:(ResNet20)

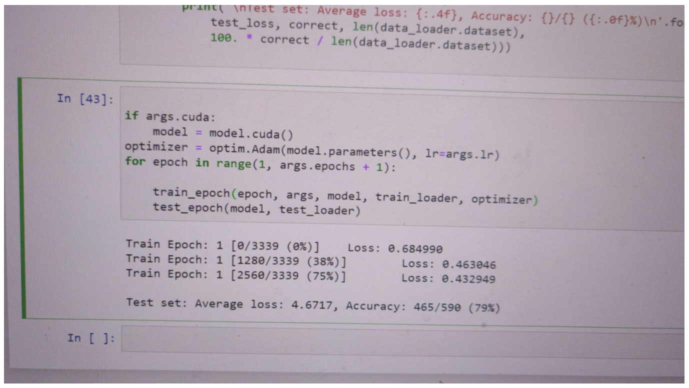




VII. Running Pyspark jobs in AWS EMR Cluster vs AWS EMR Notebook

A. *AWS EMR Notebook*

One can use Amazon EMR Notebooks along with Amazon EMR clusters running Apache Spark to create and open Jupyter Notebook and JupyterLab interfaces within the Amazon EMR console. An EMR notebook is a ”serverless” notebook where the commands are executed using a kernel on the EMR cluster. Notebook contents are also saved to Amazon S3 separately from cluster data for durability and flexible re-use. One can start a cluster, attach an EMR notebook for analysis, and then terminate the cluster. One can also close a notebook attached to one running cluster and switch to another. Multiple users can attach notebooks to the same cluster simultaneously and share notebook files in Amazon S3 with each other [4].We observed the following output when we choose ”1” as the parameter for epoch.



```
    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.fo
        test_loss, correct, len(data_loader.dataset),
        100. * correct / len(data_loader.dataset)))

In [43]:
if args.cuda:
    model = model.cuda()
optimizer = optim.Adam(model.parameters(), lr=args.lr)
for epoch in range(1, args.epochs + 1):

    train_epoch(epoch, args, model, train_loader, optimizer)
    test_epoch(model, test_loader)

Train Epoch: 1 [0/3339 (0%)]    Loss: 0.684990
Train Epoch: 1 [1280/3339 (38%)]        Loss: 0.463046
Train Epoch: 1 [2560/3339 (75%)]        Loss: 0.432949

Test set: Average loss: 4.6717, Accuracy: 465/590 (79%)

In [ ]:
```

Fig. 6. Result with 1 epoch

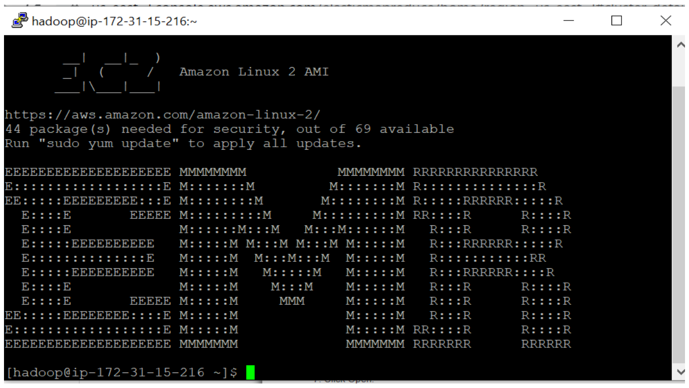
B. *AWS EMR through SSH in Name Node*

There is another way to run pyspark applications in AWS EMR. After a cluster is created, SSH into the Master Node's EC2 Instance. The EC2 key pair that was used in creating the cluster is needed for using SSH into the name node machine. The steps are to copy the python code from AWS S3 to the Name Node's HDFS Storage. The steps to be followed are given on the following images:-

VIII. Conclusion & Future Work

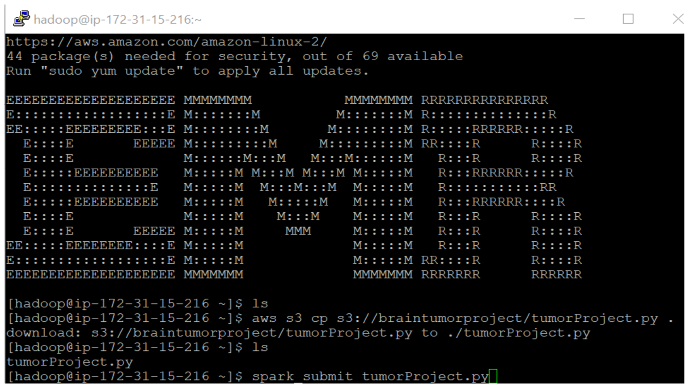
We notice that ResNet18 has achieved an accuracy of 97% and ResNet50 has acheived an accuracy of 96% in the best setting when it comes to accurately predicting whether the Brain MRI scans have tumors in them.

With this approach we were successful to accurately classify whether an MRI Scan has a tumor in it. We could take this further and use a segmentation model such as a ResUNet to determine and localize the tumor in the MRI Scan. We could also classify the type of tumor that exists in the MRI Scan to take it one level forward. When we don't have much data to train our model, we can use data augmentation to add to data to get higher accuracy. We could also use data augementation by making some changes in the existing dataset like flipping, rotating, scaling to achieve better results.



```
hadoop@ip-172-31-15-216:~

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
44 package(s) needed for security, out of 69 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-15-216 ~]$
```

Fig. 7. Access EMR after SSH inside Master Node



```
hadoop@ip-172-31-15-216:~
https://aws.amazon.com/amazon-linux-2/
44 package(s) needed for security, out of 69 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-15-216 ~]$ ls
[hadoop@ip-172-31-15-216 ~]$ aws s3 cp s3://braintumorproject/tumorProject.py .
download: s3://braintumorproject/tumorProject.py to ./tumorProject.py
[hadoop@ip-172-31-15-216 ~]$ ls
tumorProject.py
[hadoop@ip-172-31-15-216 ~]$ spark_submit tumorProject.py
```

Fig. 8. Submitting a PySpark Job in the Master Node

References

[1] Dipanjan Sarkar. ”A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning”

[2] S. J. Pan and Q. Yang, ”A Survey on Transfer Learning,” in IEEE Transactions on Knowledge and Data Engineering

[3] Ying Tai, Jian Yang, Xiaoming Liu; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017

[4] https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-managed-notebooks.html

[5] Simard, Patrice Y., David Steinkraus, and John C. Platt. ”Best practices for convolutional neural networks applied to visual document analysis.” Icdar. Vol. 3