Classification of Brain Tumor MRI Scans using Transfer Learning with a Comparative Analysis on Pre-Trained Networks

Ashita Shah
Department of Information Technology
Ramrao Adik Institute of Technology
Nerul, Navi Mumbai 400706, MH, India.
shahashita03@gmail.com

Pallavi Chavan
Department of Information Technology
Ramrao Adik Institute of Technology
D Y Patil Deemed to be University,
Nerul, Navi Mumbai 400706, MH, India.
pallavi.chavan@rait.ac.in

Dipti Jadhav
Department of Information Technology
Ramrao Adik Institute of Technology
D Y Patil Deemed to be University,
Nerul, Navi Mumbai 400706, MH, India.
dipti.jadhav@rait.ac.in

***Abstract*— Classification of brain MRI scans accurately to determine whether they are tumorous or not is a challenging task. Deep learning models are popularly used for the classification of data and requires lot of computational time and memory, however flexibility is highest when trained from scratch for a particular application. Where there is a constraint in availability of datasets, memory, computational time and efforts, transfer learning plays a major role to train models efficiently. In this work, transfer learning is used to classify the brain MRI scans to determine whether they are tumorous or not. Transfer learning helps to train the pre-trained models on a smaller dataset by fine-tuning the last learnable layers. Pre-trained models like GoogleNet, Alexnet, SqueezeNet, VGG16 and VGG19 are first trained and tested on a small 'Brain Tumor' dataset. Training is performed by fine-tuning the learnable layer of each network and by setting the training options. Performance is evaluated by using confusion matrices, SqueezeNet achieved highest accuracy of 92.08%. Further Alexnet and SqueezeNet are trained on the dataset consisting of 3064 T1-weighted images. Thus, modifying and fine-tuning the last layers, transfer learning helps to train a network which was pre-trained on a different dataset to be further trained on the desired dataset for making predictions. Comparative analysis is presented at the end of the study.**

Keywords*— *Brain Tumor Classification, Glioma, Meningioma, MRI Scans, Pituitary Tumor, Transfer Learning

I. INTRODUCTION

Deep learning has healthcare applications in fields like bio-signal analysis (where medical events such as seizures and cardiac arrests are predicted), making decisions in case of diagnosis and survival prediction, discovery of drugs, analysing health records and selecting therapies. The major difference between machine learning and deep learning is, machine learning uses features extracted manually from a raw dataset from other models while deep learning learn features automatically once raw data is fed into it [1]. Due to the constraints in the availability of medical dataset, often deep learning models work perfectly on the unseen dataset which is similar to the trained dataset but would result in either overfitting or false positives and negatives if trained on an unseen dataset which is completely different from the trained and the validation dataset. Here transfer learning comes into picture which helps the dataset smaller in size to train on the pre-trained model, thus saving the efforts of training the deep learning model from scratch. In [2], CAD (Computer Aided Diagnosis) system is discussed which uses CNN based architecture. Two datasets are used. First dataset for classifying tumor into three classes namely, meningioma, Glioma and pituitary tumors. Second dataset to differentiate between Glioma grades namely, Grade II, III and IV. [3] discusses some of the common deep-learning architectures used for segmenting brain structures, their speed, performance and properties are summarized and the final part discusses future trends. [4] discusses CNN architecture for distinguishing between tumors from T1 MRI images by using 4 approaches, the first two approaches using a 10-fold cross validation on record-wise and subject-wise data while the next two approaches using two databases, the original database and the database on which augmentation is performed for variability. 10 fold is safe to use as it gives less error and avoids overfitting by preventing high bias and high variability. Image database consist of all three types of tumor types (Glioma, Meningioma and Pituitary tumors) of T1 weighted contrast enhanced MRI images taken in three planes (axial, coronal and sagittal planes). Tumors which are formed in glial cells are termed as Glioma, it is important to detect the type of Glioma and its corresponding location for appropriate treatment during chemotherapy and surgery. The classification methods discussed in [5] are Multi-layered Perceptron (MLP), Support Vector Machines (SVM) and Random Forest (RF) Tree. [6] uses U-Net to segment tumors from MRI images using target magnification and data augmentation techniques. [7] discusses segmenting the MRI scans into background, whole tumor, tumor core and enhanced tumor by using a cascade of CNNs each using dilated and anisotropic filters. It shows a cascaded architecture segmenting tumors hierarchically, i.e. segmenting first the whole tumor, then the tumor core inside the whole tumor, then the enhanced tumor inside the tumor core. [8] discusses using multi-modality images in T1, T1 contrast enhanced (T1 CE, where the tumor border appears brighter due to gadolinium used), T2 (showing the edema which is the region around the tumor) and Fluid Attenuated Inversion Recovery (FLAIR, which distinguishes between the edema and the cerebrospinal fluid) for segmenting tumor sub-regions such as Whole Tumor (WT), Tumor Core (TC) and Enhanced Tumor (ET). U-Net has a drawback that the network works well on the training data but doesn't work on testing or unseen data hence a new model was proposed known as the Discriminative Adversarial Network (DAN) which consists of an embedded U-Net. [9] discusses the ensemble of 3D U-Nets in which each U-Net is trained with different hyper parameters for segmenting sub-regions of Gliomas namely peritumoral edema, necrotic core, enhancing tumor and non-enhancing tumor and in addition to it, a model is developed using features extracted for predicting the survival of the patient and reduces the regression errors. [10] discusses prediction of brain tumor grades i.e. high grade glioma (HGG) and low grade glioma (LGG) with and without transfer learning. It discusses three architectures, PatchNet, SliceNet and VolumeNet which take MRI patches, slices and volumetric images respectively and do prediction without extracting features and manual segmentations. [11] discusses a 3D CNN based model known as DeepMedic for automatic segmentation of brain tumor on Brain Tumor Segmentation (BRATS) dataset on all the four modalities i.e. T1, T1 CE, T2 and FLAIR. [12] discusses Enhanced Convolutional Neural Networks (ECNN) for MRI image segmentation using loss function optimization. [13] performs Brain Age Estimation (BAE) on MRI brain images. Two factors are used for categorizing BAE methods: first being the pixel-based, surface-based and voxel-based methods for processing MRI images, second being the deep learning methods. [14] discusses different deep learning methodologies for prognosis of eight neurological disorders, their challenges and future scope. [15] discusses the structural anatomy of the brain which includes two parts, one is the neuron which forms a highly specialized cellular brain unit and other is the glia which is a supportive cellular element. There's more glia in the brain than there are neurons. For describing physical relationship and location in the brain, standard terms are used. These terms are mainly used for research in MRI for denoting relative structure locations in the brain as well as for finding activations in the brain for MRI research.

II. Proposed Framework

A. Block Diagram

Deep learning models are pre-trained on millions of images in ImageNet including 1000 classes. If the images on which the networks are trained, doesn't match the new unseen dataset, it would lead to misclassifications. Transfer-learning helps the network to be trained again on the new dataset by fine-tuning the last learnable layers on the smaller dataset. Thus after undergoing training on the required dataset, the model's performance can be tested on an unseen new dataset for making predictions. This paper shows 3 approaches: In approach 1, the model is tested on a brain tumor dataset on 5 pre-trained models, in order to test the highest performance. As SqueezeNet achieved highest accuracy and Alexnet, SqueezeNet performed faster computations in terms of time, it was further used in approach 2. In approach 2, SqueezeNet and Alexnet were trained on the final dataset of 3064 images with 3 tumor classes without using k-cross validation, to test which of the two achieves highest accuracy.

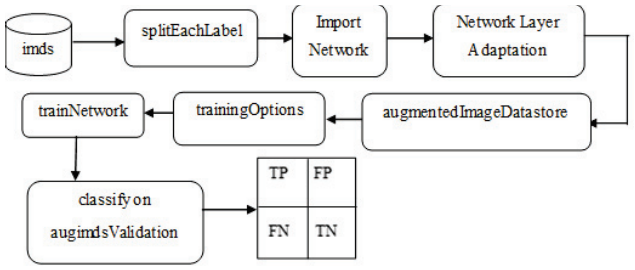


Fig. 1. Proposed framework with training using transfer-learning. Each block representing the functions used for experimentation.

As a result, SqueezeNet achieved highest accuracy of 86.29%. Hence it was further used in approach 3 with k-cross validation and with different combinations of epochs and mini-batch size which achieved the final accuracy of 94.44%. Fig. 1 represents the block diagram of proposed framework. The model is proposed in reference to the concepts of transfer-learning mentioned in [16].

B. Image Datastore (imds)

Image datastores (imds) represent the collection of images, folders and subfolders all included in a single place. For approach 1, an image datastore of 'Brain Tumor Dataset' folder is created. In mostly every classification problem, the folders have their corresponding sub-folders which indicate the number of classes. In the 'Brain Tumor Dataset' folder, there are two sub-folders which thus corresponds to two classes namely, 'yes' and 'no'. For approach 2 and 3, the original dataset, '3064' consist of 3064 images, each in .mat file format. Each file is a struct file having 5 fields, out of which the image data field and its corresponding label (1, 2, 3) is extracted. An image datastore/ folder is made consisting of three sub-folders in such a way that the images corresponding to their labels 1, 2, and 3 are stored in their respective sub-folders which also have the names 1, 2 and 3. Thus the meningioma tumor images (class 1/ label 1) are stored in sub-folder 1, the glioma tumor images (class 2/ label 2) are stored in sub-folder 2 and the pituitary tumor images (class 3/ label 3) are stored in sub-folder 3. Thus performing computation on the imds, would affect the entire datastore including the subfolders. imds is presented in Fig. 2.

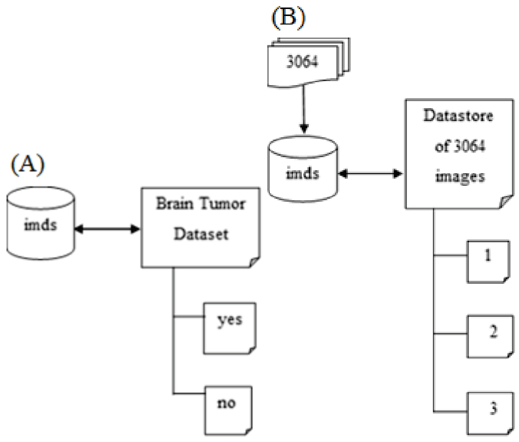


Fig. 2. Image Datastore, (A) image datastore for approach 1 (B) image datastore for approach 2, 3

C. Splitting imds into Training and Validation/ Testing Datastores (splitEachLabel)

In approach 1, the datastore is being split into training datastore and a validation/ testing datastore. 60% of the data is being selected as the training data and the remaining 40% as the validation data. So there are 152 images in the training dataset from both the subfolders and 101 images in the validation dataset respectively. Often the behaviour of the network depends on the ratio of how many images are been taken into the training and validation datasets. If the training dataset contains images from one class more than the other, the network would be trained more on that particular class thus leading to over-fitting or under-fitting issues and misclassifications. Thus we can even divide the datasets in such a way that it takes almost equal amount of images from both the classes in training and validation datasets. Store these training and validation datasets in the training datastore

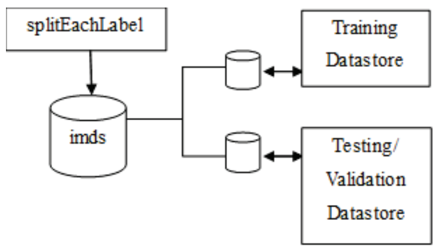


Fig. 3. Splitting the imds datastore into training and validation datastores using splitEachLabel function (approach 1, 2)

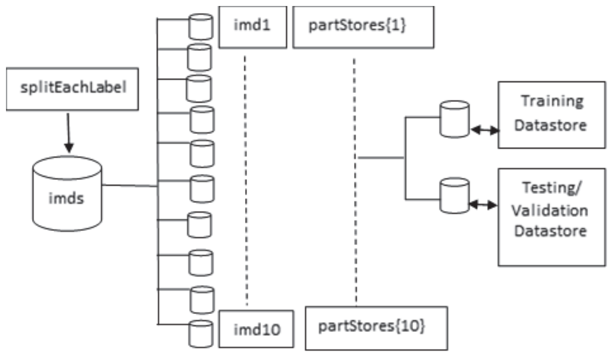


Fig. 4. Splitting the imds datastore into training and validation datastores using splitEachLabel function (approach 3)

and validation datastore as shown in Fig. 3. In approach 2, 70% of the data is being selected as the training data and the remaining 30% as the validation data. So there are 2145 out of 3064 images in the training dataset from all the three subfolders and 919 images in the validation dataset respectively.

In approach 3, the image datastore (imds) is first being split into 10 equal parts namely imd1-10 using splitEachLabel and randomize function which randomly selects the data in each of the 10 datastores. Each of the 10 files (imd1-10) are stored in partitions in partStores variable which is a partition 1×10 array storing 10 imd files in 10 partitions respectively as shown in Fig. 4.

D. Import Network

In approach 1, five pre-trained networks are used such as alexnet, googlenet, squeezenet, vgg16 and vgg19 for performing transfer-learning. In approach 2, alexnet and squeezenet are used. In approach 3, squeezenet is used with 5 cross and 10 cross validations.

E. Network Layer Adaptation

In CNN, the layers after the input layer, i.e. convolution (Conv), RelU and max-pooling layers are used for extracting features and the further layers, i.e. the fully connected layer (fc), soft-max and final classification layer is used for classifying images into categories based on the extracted features. Thus in transfer-learning, the last few layers are modified and fine-tuned for carrying out the classification as required. It involves replacing the learnable layers with a new layer and the classification layer with a new classification output layer. The learnable layers can be fc or Conv. In networks such as Alexnet, GoogleNet, VGG16 and VGG19, the learnable layer is the fully connected layer. In SqueezeNet,

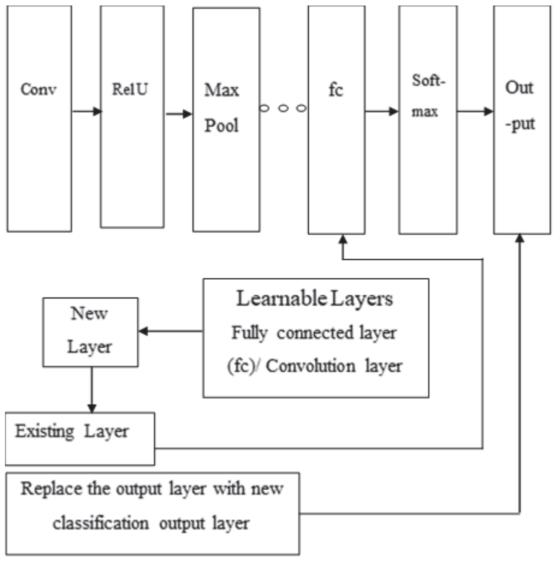

Conv
RelU
Max Pool
fc
Soft-max
Out -put
New Layer
Learnable Layers
Fully connected layer (fc)/ Convolution layer
Existing Layer
Replace the output layer with new classification output layer

Fig. 5. Network Layer Adaptation

the learnable layer is the convolution layer. If the networks were pre-trained on ImageNet, the output size would be 1000, but in approach 1, the required output size is 2 and in approach 2,3 the required output size is 3 as it contains 2 and 3 classes respectively. So a new fc/ Conv layer is given the number of classes which are required to be classified and then the new layer replaces the initial learnable layer. Similarly, the output from fc is given to the classification output layer which also has an output size of 1000 if trained on the ImageNet. Thus it is replaced by the new output layer whose output size is not defined. Thus it would reset the size of the output depending on the input from the fc layer. The learning rates of the first few layers are frozen [17], because they are already trained and updated on the basis of their previous weights. The learning rates of the replaced layers are increased as they need rapid learning on the new dataset for appropriate classification, and the rates of the middle layers are reduced so that they don't change their weights rapidly. This balancing of learning rate helps the network to converge faster with less training time needed.

F. Augmented Image Datastore (augmentedImageDatastore)

Data augmentation helps to modify images in the imds to suit the parameters of the pre-trained network. As shown in Fig. 6. the images in the imds are grayscale images but the pre-trained network for example, VGG16 have an image input size of 224-by-224-by-3. Thus it takes an RGB image as its input. Data augmentation converts the images in the imds by making 3D arrays of each image and resizing it according to the input size of the network on which the training needs to be performed. If the input size of the images in the dataset is not adjusted according to the network, it would give an error of inappropriate input size. Image data augmenter (imageDataAugmenter) helps to perform various morphological augmentation operations such as image flipping along vertical axis, translating them to 30 pixels and scaling them up to 10% horizontally and vertically. All these data augmentation operations are performed on both the training datastore and the validation datastore to convert it into an augmented training datastore and an augmented validation datastore respectively. The augmented training datastore undergoes training and the trained network is then tested on the augmented validation datastore.

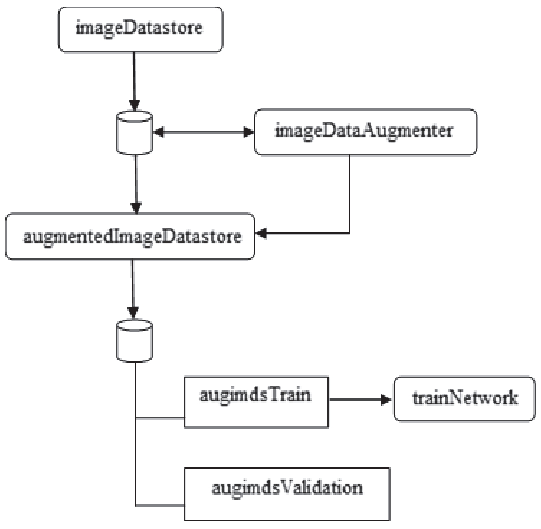


Fig. 6. Augmented Image Datastore

G. Set Training Options (trainingOptions)

Setting the training options involves setting the initial learning rate, maximum epochs, mini-batch size, validation accuracy, learning algorithm, etc. The following sub-sections explain each in detail.

1) Stochastic Gradient Descent with Momentum (sgdm) and Learning Rate

Stochastic Gradient Descent is a learning algorithm which helps to optimize the weights for the next iteration. Here gradient indicates the steepness of a ground or the slope to the very bottom. The height of the slope is often determined by the loss function and we try to improve the accuracy by going towards the slope's bottom to reach the bottom value of the ground. If the network is giving a bad performance, the initial learning rate is divided by 10 from its default value 0.01 and trained again. Momentum helps to take small steps towards the gradient, but at the next step it does not directly change the direction to a completely new direction but turn towards it. Thus it turns according to the weighted average between the previous direction and the new direction. This weighing of how much to turn is set by the momentum option in the training options. In this paper, the initial learning was set to a very small value for the middle layers to slow down its training, so that it does not update the weights rapidly. The initial layer's learning rate was already frozen so that it used the pre-trained weights and the new layer's learning rate was increased for increasing the classification accuracy. This balancing of learning rate leads to faster convergence of the network.

2) Mini-batch Size and Maximum Epochs

The training datastore is divided into small groups called the mini-batches set by specifying the size of the mini-batch in the training options. During an iteration, the network forms a mini-batch randomly by shuffling the data if the option is being set in the training options. One epoch corresponds to the traversal of the entire training set by the network.

3) Accuracy and Loss

Validation accuracy is calculated at each epoch. In the training progress, accuracy of the network while training should either keep increasing or remain constant. Similarly, the loss while training should keep on decreasing. Train the network until its loss keep decreasing, if the training loss curve is seen to be increased, stop training the network otherwise it would lead to network over-fitting.

4) trainNetwork and Transfer Learning

Transfer-learning have requirements of network layer modification, training data and training options. After acquiring these three requirements, the network is trained on the training dataset. If the results are sufficient and fulfilling, the training is stopped. But if the results are insufficient, modification are done in the training options to re-train the network. In approach 1, SqueezeNet achieved highest accuracy among all the five networks with 92.08% validation accuracy. In approach 2, SqueezeNet achieved highest accuracy as compared to Alexnet with 86.29% validation accuracy. In approach 3, SqueezeNet achieved highest accuracy among 5-cross and 10-cross validations with 94.44% validation accuracy.

5) Classifying on Augmented Validation Dataset (classify function on augimdsValidation)

The trained network is evaluated on the validation dataset and the accuracy is tested by comparing the predictions made by the network and the validation ground truth labels. The predicted labels are then evaluated using performance parameters like confusion matrix, number of correctly classified and misclassified fraction, etc.

III. Experiments and Results

A. Approach 1

The 'Brain MRI Images for Brain Tumor Detection' dataset [18] consists of two subfolders labelled 'yes' and 'no', indicating tumorous and non-tumorous images respectively. The images are axial MRI 2D scans. There are in total 253 images, 155 and 98 images of yes and no categories respectively. Experiments were performed using MATLAB tool (Deep Learning Toolbox, Image Processing Toolbox, Computer Vision Toolbox) and training was performed on Single CPU. The dataset was first transformed into an image datastore. It was then split into 60% training datastore and 40% validation datastore. The input size of each network imported was evaluated. The learnable layer and the classification output layer were replaced with new layers on the basis of the number of classes into which the classification should be done, 2 in this case. The weights of the first 10 layers were being frozen so that the network considered the weights on which it was pre-trained. Image data augmenter performed augmentations like flipping and scaling vertically and horizontally. Thus the training datastore and the validation datastore were converted into augmented training datastore and augmented validation datastore respectively by considering the input size and augmentation options. The training options were being set such as sgdm was used as the learning algorithm, mini-batch size was set to 10, maximum epochs was set to 6 and initial learning rate was set to a very small value. The network was trained on the three necessary parameters which are required for training such as the modified network layers, augmented training datastore and the training options. The trained network was tested for classification on augmented validation datastore. Accuracy was calculated by comparing the predicted labels and the ground truth validation datastore labels by taking its mean. Table I. compares the results of all the five pre-trained networks.

TABLE I. APPROACH 1: COMPARING THE RESULTS ACQUIRED AFTER TRAINING AND TESTING ON THE FIVE PRE-TRAINED NETWORKS AS MENTIONED BELOW, FOR VIEWING WHICH OF THE NETWORKS GIVE HIGHEST VALIDATION ACCURACY AND LEAST COMPUTATIONAL TIME.

Network	Depth	Input Size	Learnable Layers	Validation Accuracy	No. of Correct Labels	Elapsed Time on Single CPU (in seconds)
googlenet	22	224-by-224	Fully connected layer, Classification Output Layer	88.12%	89	122
alexnet	8	227-by-227	Fully connected layer, Classification Output Layer	85.15%	86	100
squeezenet	18	227-by-227	Convolution layer, Classification Output Layer	92.08%	93	98
vgg16	16	224-by-224	Fully connected layer, Classification Output Layer	90.10%	91	369
vgg19	19	224-by-224	Fully connected layer, Classification Output Layer	87.13%	88	457

Out of all the five networks, SqueezeNet achieved highest performance in terms of accuracy, number of correct labels and time required to train the network. As shown in the Table. 1, SqueezeNet [19] achieved highest accuracy with the least computational time to train the network with the maximum number of correct labels identified. This is because of the design of the Squeezenet which includes 'Fire Modules' in the network architecture, thus fitting well in the computer memory. Squeezenet adapts to three major strategies, the first strategy reduces the 3×3 filters to 1×1 filters given in the squeeze layer, the second strategy uses expand layer in which 1×1 and 3×3 filters are fed with less input parameters from the squeeze layer and the third strategy down-samples late (having smaller stride values), so that the last layers have larger activation maps which results in better accuracy. In the networks like Alexnet, in the last layers, the activation maps are also reduced with smaller activations, thus Squeezenet achieves higher accuracy. It includes a convolution layer, followed by fire modules (squeeze and expand layers), followed by the last learnable convolution layer. Hence with less number of parameters, the network fits well into the memory and converges faster as compared to the other networks.

B. Approach 2

As in approach 1, SqueezeNet achieved highest accuracy of 92.08% on a small dataset and Alexnet, SqueezeNet took least computational time, 1 min 40 sec and 1 min 38 sec respectively as compared to other networks, both Alexnet and Squeezenet were thus used in further experimentation for approach 2 (without k-cross validation). The final brain tumor dataset [20] consists of 3064 images from 233 patients with 3 types of tumors: meningioma, glioma and pituitary tumors, each with 708, 1426 and 930 images respectively. Experiments were performed using MATLAB tool (Deep Learning Toolbox, Image Processing Toolbox, Computer Vision Toolbox) and training was performed on Single CPU. An image datastore/ folder is made consisting of three sub-folders in such a way that the images corresponding to their labels 1, 2, and 3 are stored in their respective sub-folders which also have the names 1, 2 and 3. It was then split into 70% training datastore and 30% validation datastore. Thus there were 2145 and 919 images in training and validation/testing datastore respectively. The learnable layer and the classification output layer were replaced with new layers on the basis of the number of classes into which the classification should be done, 3 in this case.

As shown in Table. II, SqueezeNet achieved highest accuracy of 86.28% with 10-7 mini batch size - no. of epochs combination. Hence in approach 3 (with k-cross validation), SqueezeNet was used, the first combination for experimentation being 10-7 for mini-batch size and no. of epochs. However, in approach 2, highest accuracy was achieved with rms prop being the optimization algorithm, but as the network's efficiency is tested both on accuracy and the computational time, sgdm performs better than rms prop in SqueezeNet in terms of computational time (80 min 57 sec) with not much difference in the validation accuracy (84.98%). Hence for final experimentation in approach 3, SqueezeNet with 10-7 mini-batch size and no. of epochs, 0.0001 as the learning rate and sgdm as the optimization algorithm were the parameters defined in the first step combining it with k-cross validation.

C. Approach 3

In approach 3, 5-cross and 10-cross validations were used with SqueezeNet with different combination of mini-batch size and no. of epochs. In 5-cross validation, splitEachLabel splits the imds datastore into 5 stores namely imd1 to imd5. These are then stored in partitions in partStores variable containing 5 partitions, one for each imd store. 1 part out of the 5 parts corresponding to index 5 is taken in the testing/validation datastore and the remaining 4 parts are taken in the training datastore. Similarly, in 10-cross validation, splitEachLabel splits the imds datastore into 10 stores namely imd1 to imd10. These are then stored in partitions in partStores variable containing 10 partitions, one for each imd store. 1 part out of the 10 parts corresponding to index 10 is taken in the testing/validation datastore and the remaining 9 parts are taken in the training datastore. In approach 3, 0.0001 and sgdm are used as the learning rate and optimization algorithm respectively. Table. II below shows the results of the experiments performed in approach 2, 3.

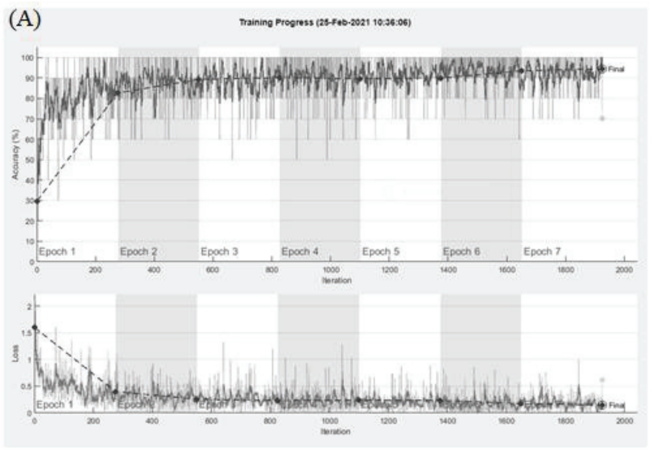
TABLE II. APPROACH 2: COMPARING THE RESULTS ACQUIRED AFTER TRAINING AND TESTING ON SQUEEZENET AND ALEXNET, WITH DIFFERENT COMBINATIONS OF MINI-BATCH SIZE, NO. OF EPOCHS AND OPTIMIZATION ALGORITHM. APPROACH 3: COMPARING THE RESULTS ACQUIRED AFTER TRAINING AND TESTING ON SQUEEZENET WITH K-CROSS VALIDATION. USING 5-CROSS AND 10-CROSS VALIDATION WITH DIFFERENT COMBINATION OF NO. OF EPOCHS.

Approach	**Value of k**	**Network**	**Mini-batch Size**	**Iterations per epoch**	**No. of Epochs**	**Elapsed Time (in seconds)**	**Validation Accuracy**	**Algorithm**
2	NA	SqueezeNet	10	214	6	4857	84.98%	sgdm
		SqueezeNet	10	214	7	7473	86.28%	rms prop
		SqueezeNet	10	214	6	6281	75.73%	adam
		Alexnet	10	214	6	5820	81.07%	sgdm
		Alexnet	10	214	6	10879	77.58%	rms prop
		Alexnet	8	268	7	11553	81.83%	adam
3	5	SqueezeNet	10	275	6	6334	92.48%	sgdm
		SqueezeNet	10	275	7	7488	94.14%	sgdm
		SqueezeNet	10	275	8	9206	94.12%	sgdm
		SqueezeNet	10	275	9	11895	94.44%	sgdm
3	10	SqueezeNet	10	275	6	6764	93.46%	sgdm
		SqueezeNet	10	275	7	7462	94.44%	sgdm
		SqueezeNet	10	275	8	8423	94.44%	sgdm
		SqueezeNet	10	275	9	9205	93.46%	sgdm

In Table. II, 5-cross and 10-cross validation results are shown with 10 as the mini-batch size and 6, 7, 8, 9 as the no. of epochs. Here, first the network was tested on 10-cross validation with 10-7 mini batch size - no. of epochs. It gave 94.44% accuracy. Then the network was tested on combination 10-8, in which the accuracy remained the same. Further on increasing the no. of epochs to 9, the accuracy started to decrease. Similarly, after using 10-6 combination with 10-cross validation, the accuracy was decreased. Hence 10-cross validation achieved highest accuracy of 94.44%. Then the network was tested on 5-cross validation with 10-7 combination. Further by increasing the no. of epochs, 94.44% was achieved with 10-9 combination. Further after increasing the epoch size to 10, the accuracy started to decrease. Also, 10-6 combination showed decrease in the accuracy as compared to 10-7 with 5-cross validation. Hence 5-cross validation also achieved highest accuracy of 94.44%.

Fig. 7. shows the training progress and the output of 10-cross validation performed on Squeezenet for 10-7 mini batch size - no. of epochs.

However, after comparing the time taken by both 5-cross validation and 10-cross validation on 10-7 mini batch size – no. of epochs combination, the time taken by 10-cross validation to achieve 94.44% was less than the time taken by 5-cross validation to achieve the same accuracy. The final results are shown in Table III.



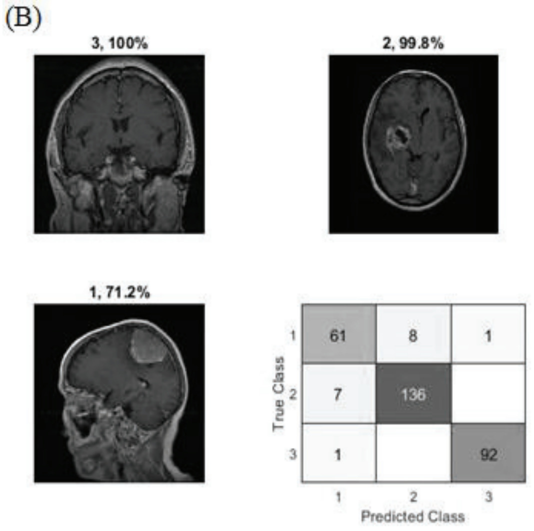


Fig. 7. (A) Training progress and (B) confusion matrix of 10-cross validation performed on SqueezeNet, k=10, mini-batch size=10, no. of epochs=7

TABLE III. FINAL RESULTS: AS SQUEEZENET ACHIEVED HIGHEST ACCURACY WITH 10-CROSS VALIDATION ON THE FINAL DATASET, ALL THE PARAMETERS USED FOR TRAINING AND TESTING ARE SUMMARIZED IN THE TABLE AS SHOWN BELOW.

Network	**SqueezeNet**
k-cross validation	10
Training-Validation Datastore Images	2758-306
Mini-batch Size	10
No. of Epochs	7
Learning Rate	0.0001
Optimization Algorithm	Stochastic Gradient Descent with Momentum (sgdm)
Iterations per Epoch	275
Maximum Iterations	1925
Elapsed Time	7462 seconds
No. of Correct Labels	289/ 306
Validation Accuracy	94.44%

IV. CONCLUSION

In this paper, we proposed a method to train on an unseen dataset which requires less memory, less computational time, less efforts with reasonable amount of flexibility and accuracy by only fine tuning the last layers of a pre-trained network as suggested in [2] for training on a smaller dataset and avoiding training from scratch. In approach 1, network classifies the brain tumor dataset into tumorous and non-tumorous categories trained on pre-trained networks such as Alexnet, GoogleNet, SqueezeNet, VGG16 and VGG19 networks. As in approach 1, SqueezeNet achieved highest accuracy, whereas Alexnet and SqueezeNet performed faster computations as compared to other three pre-trained networks, Alexnet and SqueezeNet were used in approach 2. In approach 2, the network was tested on the final dataset containing 3064 images from 233 patients with three types of brain tumors namely the meningioma, glioma and pituitary tumors. A comparative analysis was performed on Alexnet and SqueezeNet with different combinations of mini-batch size, no. of epochs and optimization algorithms. SqueezeNet achieved highest accuracy of 86.28% with rms prop as the optimization algorithm. Whereas, sgdm performed faster computations, hence SqueezeNet was used in approach 3 with sgdm as the optimization algorithm. In approach 3, 5-cross and 10-cross validations were performed, out of which 10-cross validation achieved highest accuracy of 94.44% with faster computational time.

REFERENCES

[1] Lundervold AS, Lundervold A. An overview of deep learning in medical imaging focusing on MRI. Zeitschrift fur Medizinische Physik. 2019 May;29(2):102-127. DOI: 10.1016/j.zemedi.2018.11.002.

[2] X H. H. Sultan, N. M. Salem and W. Al-Atabany, "Multi-Classification of Brain Tumor Images Using Deep Neural Network," in IEEE Access, vol. 7, pp. 69215-69225, 2019, doi: 10.1109/ACCESS.2019.2919122.

[3] Akkus, Zeynettin & Galimzianova, Alfiia & Hoogi, Assaf & Rubin, Daniel & Erickson, Bradley. (2017). Deep Learning for Brain MRI Segmentation: State of the Art and Future Directions. Journal of digital imaging. 30. 10.1007/s10278-017-9983-4.

[4] Badza, Milica & Barjaktarovic, Marko. (2020). Classification of Brain Tumors from MRI Images Using a Convolutional Neural Network. Applied Sciences. 10. 1999. 10.3390/app10061999.

[5] Latif, Ghazanfar & Iskandar, Dody & Alghazo, Jaafar. (2018). Multiclass Brain Tumor Classification using Region Growing based Tumor Segmentation and Ensemble Wavelet Features. ICCBD '18: Proceedings of the 2018 International Conference on Computing and Big Data. 67-72. 10.1145/3277104.3278311.

[6] Ho, Te-Wei & Qi, Huan & Lai, Feipei & Xiao, Fu-Ren & Wu, Jin-Ming. (2019). Brain Tumor Segmentation Using U-Net and Edge Contour Enhancement. ICDSP 2019: Proceedings of the 2019 3rd International Conference on Digital Signal Processing. 75-79. 10.1145/3316551.3316554.

[7] Wang, Guotai & Li, Wenqi & Ourselin, Sébastien & Vercauteren, Tom. (2018). Automatic Brain Tumor Segmentation Using Cascaded Anisotropic Convolutional Neural Networks. 10.1007/978-3-319-75238-9_16.

[8] Liu, Jing & Yin, Pengyu & Wang, Xiaohui & Yang, Wei & Cheng, Kun. (2019). Glioma Subregions Segmentation with a Discriminative Adversarial Regularized 3D Unet. ISICDM 2019: Proceedings of the Third International Symposium on Image Computing and Digital Medicine. 269-273. 10.1145/3364836.3364891.

[9] Feng X, Tustison NJ, Patel SH, Meyer CH. Brain Tumor Segmentation Using an Ensemble of 3D U-Nets and Overall Survival Prediction Using Radiomic Features. Front Comput Neurosci. 2020;14:25. Published 2020 Apr 8. doi:10.3389/fncom.2020.00025

[10] Banerjee, S., Masulli, F., & Mitra, S. (2017). Brain Tumor Detection and Classification from Multi-Channel MRIs using Deep Learning and Transfer Learning.

[11] Perkuhn M, Stavrinou P, Thiele F, et al. Clinical Evaluation of a Multiparametric Deep Learning Model for Glioblastoma Segmentation Using Heterogeneous Magnetic Resonance Imaging Data From Clinical Routine. Invest Radiol. 2018;53(11):647-654. doi:10.1097/RLI.0000000000000484

[12] Thaha, M.M., Kumar, K.P.M., Murugan, B.S. et al. Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images. J Med Syst 43, 294 (2019).

[13] Sajedi, H., Pardakhti, N. Age Prediction Based on Brain MRI Image: A Survey. J Med Syst 43, 279 (2019).

[14] Gautam, R., Sharma, M. Prevalence and Diagnosis of Neurological Disorders Using Different Deep Learning Techniques: A Meta-Analysis. J Med Syst 44, 49 (2020).

[15] Fundamental Neuroscience for Neuroimaging, available at: https://www.coursera.org/learn/neuroscience-neuroimaging

[16] Deep Learning Onramp, available at: https://in.mathworks.com/learn/tutorials/deep-learning-onramp.html

[17] Train Deep Learning Network to Classify New Images, available at: https://in.mathworks.com/help/deeplearning/ug/train-deep-learning-network-to-classify-new-images.html

[18] Brain MRI Images for Brain Tumor Detection, available at: https://www.kaggle.com/navoneel/datasets

[19] Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., & Keutzer, K. (2017). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. ArXiv, abs/1602.07360.

[20] 3064 images from 233 patients containing 3 types of brain tumor images: meningioma, glioma and pituitary tumors available at: https://figshare.com/articles/dataset/brain_tumor_dataset/1512427