International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 08 | Aug 2020 www.irjet.net p-ISSN: 2395-0072

Review of Transfer Learning Methods in Deep Learning for Medical Image Classification

Gurucharan M K, Shri Harini R

Department of Biomedical Engineering
Sri Sivasubramaniya Nadar College of Engineering
Chennai, Tamil Nadu, India

Abstract - *Transfer Learning methods based on Convolutional Neural Networks (CNN) have shown promising results on various Image Classification problems. Even in Medical Images such as the X-Ray, CT Scan, MRI these methods prove to be useful. Recent Advancements in the branch of Deep Learning have made it possible to detect the presence of tumor in a Brain MRI scan. Transfer Learning is a Deep Learning method in which models that are pre-trained with specific weights are used as the starting point for training another model. It is utilized mainly because the pre-trained models are already trained on large datasets and their weights can be utilized for Medical Image Classification. In this paper, we present a detailed review on sixteen various deep learning models available with pre-trained weights on the Keras library with TensorFlow backend, that have been developed for image classification such as VGG19, Xception, InceptionV3, EfficientNet etc. These Deep Learning models are also used in a Medical Image Classification problem by incorporating Transfer Learning and their results are compared. Additionally, a brief description of the Convolutional Neural Network components are also provided.*

Key Words: **Transfer Learning, CNN, Medical Images, Deep Learning, MRI, Keras, TensorFlow**

1.INTRODUCTION

1.1 Origin

Machine Learning algorithms usually learn the underlying relationship in data and use them to make decisions without requiring explicit instructions [1]. In literature, various exciting works have been reported to understand and/or emulate the human sensory responses such as speech and vision. In 1989, a new branch of Neural Networks, called Convolutional Neural Network (CNN) was discovered, which has shown enormous potential in the field of Computer Vision. CNNs are one of the best learning algorithms for understanding image content and have shown great performance in image segmentation and classification[2]. The success of CNNs has captured attention beyond academia. In industry, technical giants such as Google, Microsoft, and Facebook are actively involved in developing new architectures of CNN. At present, most of the frontrunners of image processing and computer vision competitions are employing deep CNN based models. Convolutional neural networks are widely used in pattern- and image-recognition problems as they have a number of advantages compared to other techniques.

TensorFlow is an open-source Machine Learning library that was built by Google in 2015. Though it is relatively new compared to other libraries it has gained huge importance due to its easy-to-use APIs available and simplicity in use[3]. On the other hand, Keras is a high-level library or API that is built on top of TensorFlow. It was primarily designed to facilitate fast experimentation with Neural Networks. Keras has support for Convolutional and Recurrent Neural Networks and other basic building blocks of Neural Networks such as layers, activation functions and optimizers. Both Keras and TensorFlow are completely Python-based frameworks, which makes it easier to debug and experiment with.

1.2 CNN Architecture

In Deep Learning, CNN models are used to train and test the large dataset of images[4]. Each input image will pass it through a series of Convolutional Layers with Filters, Pooling Layers and Fully Connected Layers(FC)[5].

The network architecture of CNNs comprises various different components which are described below. Each of these layers has different parameters that can be optimized and performs different tasks on the input data.

- Convolutional Layer
- Pooling Layer
- Fully Connected Layer
- Dropout
- Activation Function

Convolutional Layer -

Each convolutional neural network consists of various convolution layers depending on the requirements. Convolution is a mathematical operation in which two inputs are the image matrix and a filter of a size NxN. The Dot Product is computed between the values of the filter and the values of the Image in that respective position of NxN which constitute the final Feature Map. In the primary stages, Convolutional Layers are used for learning the various low level features such as corners and edges. The output of these layers are then fed to other convolutional layers which learn higher level features[5][6].

Pooling Layer-

A Convolutional Layer is usually followed by a Pooling Layer. The Pooling Layer is used to decrease the size of the Image in order to decrease the number of parameters in order to

decrease the computational burden. It is done by reducing the number of connections between the convolutional layers. They are usually alternated between the convolutional layers. The most important types of Pooling are Max Pooling, Average Pooling and Sum Pooling. Max pooling takes the largest element from the rectified feature map. Average Pooling computes the average of the elements from a section of a predefined size from the Image. Sum Pooling calculated the total sum of the elements in the predefined section[5][6].

Fully Connected Layer -

Fully Connected Layers usually comprise the last few layers of the Convolutional Neural Networks. These layers are present in varying numbers in a CNN, usually before the final output layer[6]. Every neuron in this layer is connected to every other neuron in the previous layer. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer[5][6].

Dropout -

In a CNN model, the Fully Connected layer is connected to all features, and it can result in overfitting of the training data. Overfitting is the problem when a model is trained and it works so well on training data that it negatively impacts the performance of the model on new data. In order to overcome this problem of overfitting, a dropout layer can be introduced in the model in which some neurons along with their connections are randomly dropped from the network during training. Thus the network is reduced in size. This reduced network is trained on the data in the next stage[5][6].

Activation Functions -

While building a CNN, one of the important choices we need to choose is the activation function to be used. The activation functions are the main components for a neural network to learn and approximate any kind of continuous and complex relationship between variables. It adds non-linearity to the network. There are a few commonly used activation functions such as Softmax, Sigmoid, ReLU and TanH[5][6].

1.3 Transfer Learning

Transfer learning is an important tool in Deep Learning that can be used to solve the basic problem of Insufficient Training Information. Due to its wide application prospects, Transfer Learning has become a very popular[7] and promising area in machine learning. In practice, a child who has learnt to play the piano can learn to play the guitar at a faster pace compared to others. Inspired by the human beings' capabilities to transfer knowledge across domains, Transfer Learning aims to leverage knowledge from a related domain, which is the source domain to improve the learning performance or minimize the number of labelled examples required in a target domain. Leading research organisations are coming out with various Transfer Learning models every year, which after release can be downloaded and incorporated directly into our models for training purposes[8].

A range of high-performing models have been developed for image classification and demonstrated on the annual ImageNet Large Scale Visual Recognition Challenge, or ILSVRC.

This challenge, often referred to simply as ImageNet, has resulted in a number of innovations in the architecture with many leading software giants taking part in it. In addition, many models developed for this competition have been released under a permissive license.

This paper aims to review these Deep Learning models that have been released under the Keras library and utilize them in training a Medical Image Classification to detect the presence of Tumor in a Brain MRI. Keras provides access to a number of top-performing pre-trained models that were developed for image recognition tasks which are elaborately described in the next section.

2. RELATED WORKS

Magnetic resonance imaging (MRI) is a type of scan that uses strong magnetic fields and radio waves to produce detailed images of the inside of the body. The task of identifying the tumor from an MRI is a complicated one. In recent times, various methods for segmentation of MRI to localize the tumor in the brain have been developed and published.

Emblem Ke et al. [9] performed SVMs on MRI Images. Beno et al. [10] came out with a brain tumor diagnosis system with a threshold-based method for the segmentation of MRI images. A collaboration of Artificial Bee Colony (ABC) and Genetic Algorithm (GA) was employed for predicting threshold values. El- Dahshan et al.[11] proposed a Feedback Pulse Coupling Network (FPCNN) for brain tumor diagnosis systems. Discrete Wavelet Transform (DWT), Principle Component Analysis (PCA) and Artificial Neural Network (ANN) are utilized for feature extraction, selection and classification Rahmani et al. and Akbarizadeh et al. [12] proposed an unsupervised feature learning technique which was related to the combination of both spectral clustering and sparse coding for the segmentation of SAR images. Patch-wise methods consist of repetitive convolutional calculations and examine spatially limited contextual features only. In order to avoid patches, a Fully Connected Network (FCN) with deconvolution layers was used for training an end to end and pixel to pixel CNN for pixel-wise prediction with the entire image as input. Lai [13] made use of the depth information and proposed a 3D convolution model which utilized the correlation between the slices. The 3D convolution network requires a large number of parameters. Usually, when a dataset has less number of samples, overfitting often occurs with 3D CNN. Ronneberger olaf et al.[14] and Cicek Ozgun et al.[15] applied the UNet architecture which had both down-sampling and up-sampling. Their methods in 2-dimensions ignored the information about the depth. Chang[16] proposed an algorithm which contained both FCN and Conditional Random Fields (CRF). S Pereira, A. Pinto,V.Alves et al. [17] proposed an automatic segmentation method based on Convolutional Neural Networks(CNN), exploring small 3x3 kernels. Sajid Iqbal et al. [18] presented a deep convolutional network (D-CNN) to segment brain tumors in MRIs. Saxena

et al.[19] applied the transfer learning with the CNN Architectures of VGG-16, Inceptionv3 and ResNet-50 models to classify the MRI Images.

In this paper, the transfer learning approach is utilized and the tumor class in the Brain MRI image is predicted with 16 different CNN Architectures that have been developed from the beginning such as LeNet-5 to the latest EfficientNet. These pre-trained models are built and are used to predict the presence of a tumor in a Brain MRI Scan. This paper can be used as a reference to develop a deep learning model with transfer learning on a small dataset.

3. DEEP LEARNING MODELS

3.1 LeNet-5

LeNet-5 is a simple Convolutional neural network proposed by Yann LeCun, Leon Bottou, Yosuha Bengio, and Patrick Haffner in 1998 [20] and it used backpropagation for training the network. It has 7 layers, among which there are 3 convolutional layers (C1, C3 and C5), 2 pooling layers (P2 and P4), and 1 fully connected layer (F6). It takes a 32x32 pixel image as an input (Grayscale). This input passes through the Convolutional layer(C1) with 6 filters each of size 5x5. C1 has 156 trainable parameters and 122,304 connections. The P2 layer is a pooling layer with 6 feature maps of size 14x14and with filter size 2x2. P2 contains 12 trainable parameters and 5880 connections. The C3 layer is again a convolutional layer with 16 feature maps having size 5x5 and it has 1516 trainable parameters and 151600 connections. The P4 layer is a pooling layer with 16 feature maps having size 5x5 and it has 32 trainable parameters and 2000 connections. The C5 layer is a fully connected convolutional layer with 120 feature maps having size 5x5 and it has 48120 trainable connections. Fully connected layer (F6) has 84 units [21].

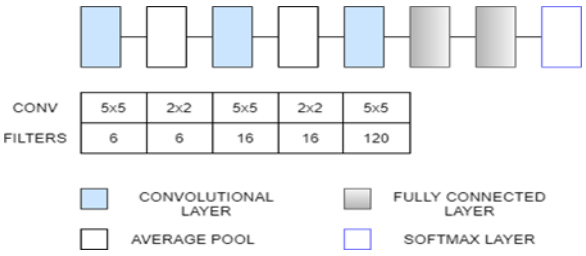


Fig. 1. LeNet-5 Architecture

3.2 AlexNet

AlexNet is a Convolutional Neural Network with 60 million parameters. It has five convolutional layers and three fully connected layers [22]. The first and second Convolutional layers are followed by the overlapping Max Pooling layers. The third, fourth and fifth convolutional layers are inter-connected. These inter-connected layer are followed by an overlapping Max Pooling layer. The output of the overlapping max Pooling layer goes into a series of two fully connected layers. The second fully connected layer is connected with a softmax classifier with 1000 class labels. ReLU (Rectified Linear Unit)[22] nonlinearity is applied after all the convolution and fully connected layers.

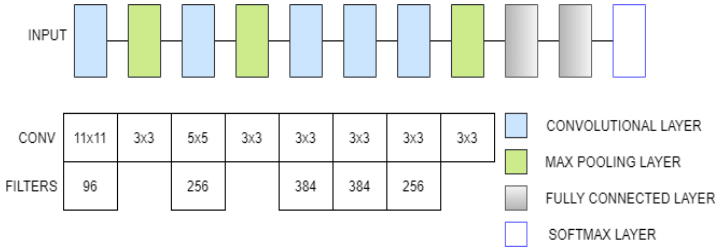


Fig. 2. AlexNet Architecture

3.3 VGG16

VGG16 is a convolutional neural network proposed by K. Simonyan et al. and A. Zisserman et al.[23] from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". It has 16 weight layers including 13 convolutional layers, 3 fully connected layers and a soft max layer. It has approximately 138 million parameters. The first and second convolutional layers have 64 filters and the size of the filter is 3×3. The input image is passed into these layers with a max pooling layer of stride 2 and the image dimension changes to 224x224x64. The third and fourth convolutional layers are of 128 filters each of size 3×3. The resulting output from these layers passed into the max pooling layer and dimensions are reduced to 56x56x128. The fifth, sixth and seventh convolutional layers have 256 filters each of size 3x3. The eighth to thirteenth convolutional layers consist of 512 filters of size 3x3. The resulting output from these layers passed into the max pooling layer. The fourteenth and fifteenth layers are fully connected hidden layers of 4096 units. The sixteenth layer is of 1000 units[23].

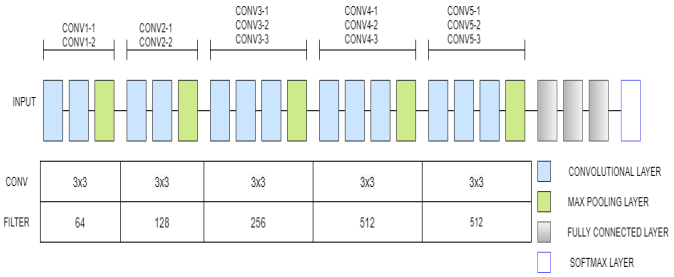


Fig. 3. VGG16 Architecture

3.4 VGG19

VGG19 is a convolutional neural network proposed by K. Simonyan et al. and A. Zisserman et al. [23]from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". It has 19 weight layers consisting of 16 convolutional layers ,3 fully connected layers and a soft max layer. It has approximately 143 million parameters. The first and second convolutional layers have 64 filters of size 3×3. Input image passed into these layers and max pooling layer of stride 2 and the image dimension changes to 224x224x64. The third and fourth convolutional layers consist of 128 filters each of size 3×3.

The resulting output from these layers passed into the max pooling layer and dimensions are reduced to 56x56x128. The fifth, sixth, seventh and eighth convolutional layers consist of 256 filters with size 3x3. The ninth to sixteenth convolutional layers consist of 512 filters each of size 3x3.The output of these layers passed into max pooling layer. The seventeenth and eighteenth layers are fully connected hidden layers of 4096 units. The nineteenth layer is of 1000 units[24][25].

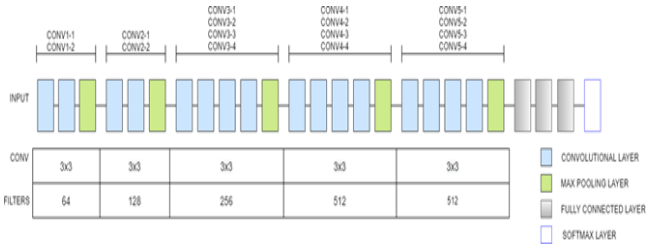


Fig. 4. VGG19 Architecture

3.5 ResNet-50

ResNet50 is a 50-layer Residual Network. This network introduces a new terminology called residual learning. ResNet performed well in classification, detection and localization, winning the ILSVRC 2015. The ResNet-50 model consists of 4 stages each with a Convolution and Identity block. Each convolution block and each identity block has 3 convolution layers . ResNet architecture performs the initial convolution using 7x7 size filters and max- pooling using 3×3 size filter. After this, the size of kernels used to perform convolution operation in all 3 layers of blocks of 4 stages are 64,64 and 128. The curved arrows and the dashed arrows refer to the identity and convolution operation in residual blocks respectively. It has approximately 23 million trainable parameters. The three layers are 1×1, 3×3, 1×1 convolutions. The 1×1 convolution layers are used for reducing and then restoring the dimensions. The 3×3 convolutional layer is left as a bottleneck with smaller input/output dimensions.After the 4 stages,The network has an Average Pooling layer followed by a fully connected layer having 1000 neurons[24].

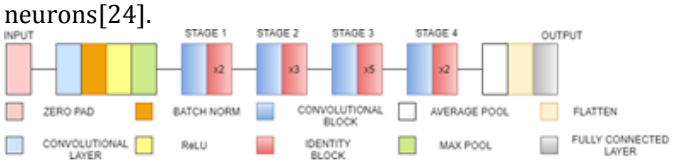
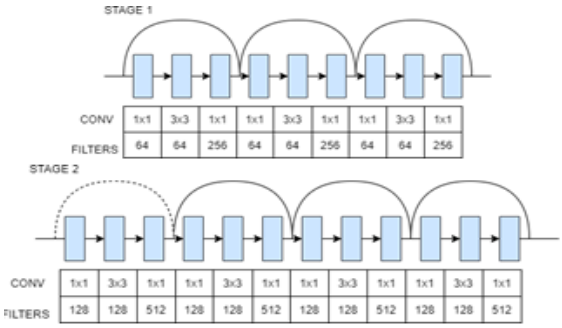


Fig. 5(a). ResNet-50 Architecture



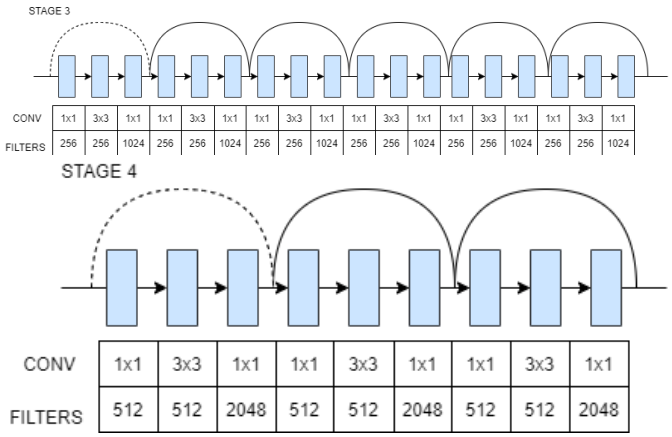


Fig. 5(b). ResNet-50 Architecture(Stages)

3.6 InceptionV3

InceptionV3 is the improved version of InceptionV1 which is also known as GoogleNet. Compared with InceptionV1, InceptionV3 has a superior performance in object recognition. This model is trained on a subset of the ImageNet database, which is used in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)[26]. This model adapts the idea of multi scale. The InceptionV3 model includes three parts: The basic convolutional block, Improved Inception module and The classifier. It has 48 layers with over 24M parameters. There are three kinds of inception modules which are Inception-A, Inception-B and the Inception-C. Each module has multiple branches with different sizes of filters (1x1,3x3,5x5,7x7). The 1×1 convolutional kernel is widely used to reduce the number of feature channels and accelerate training speed[27]. In InceptionV3, 5 Inception-A modules, 4 Inception-B modules and 2 Inception-C modules are stacked in series. After the Convolutional layers and Inception modules, the feature map dimensions are 8×8 with 2,048 channels. Then, there are 3 fully connected layers at the end of the Inception modules which allow us to utilize the pre-trained model and fine-tune the parameters for our own task. Finally, there is a softmax layer[28].

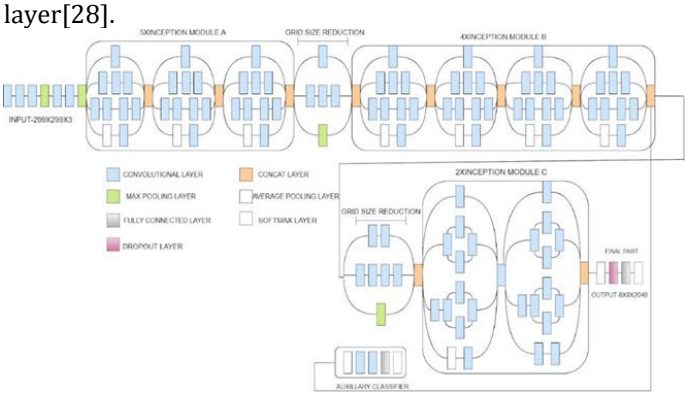


Fig. 6. InceptionV3 Architecture

3.6 InceptionResNet-V2

InceptionResNetV2 is a 164-layer deep convolutional neural network. It is also known as a hybrid inception model. It is a combination of both the Inception structure and Residual connection. Compared with inception model,this model has

better training speed.It also has Batch Normalization but present only on the top of traditional layer not on summations.They have a set of filters with sizes 1x1,3x3, 5x5, etc. that are merged with concatenation in each branch. Each Inception block is connected with a filter expansion layer of size 1x1 which is used for scaling up the dimensions of the filter bank before the addition . This model includes Inception ResNet-A,Inception ResNet-B,Inception ResNet-C[29].

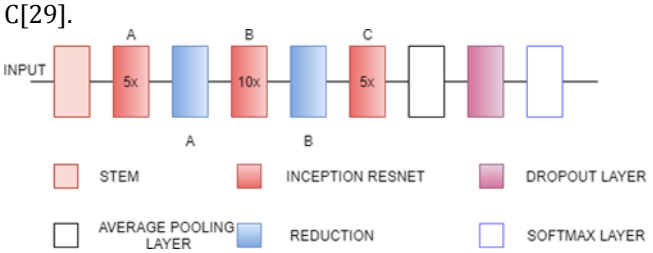


Fig. 7(a). InceptionResNetV2 Architecture

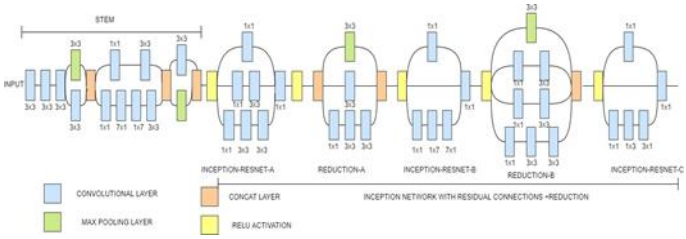


Fig. 7(b). InceptionResNetV2 Architecture detailed

3.8 Xception

Xception stands for Extreme Inception and it is 71 layers deep. Xception was developed by Francois Chollet, the creator and chief maintainer of the Keras library. Xception contains modified inception block whixh is wide and replaced the different spatial dimensions such as 1x1, 5x5, 3x3 with a single dimension (3x3) followed by a 1x1 convolution to regulate computational complexity. The Xception model is based on inception module with modifications in convolutional and depthwise separable convolution layers.It has three major blocks: Entry Flow, Middle Flow, and Exit Flow. It has 36 convolutional layers. These layers are structured into 14 modules, Except the first and last module,all these modules have linear residual connectiona around them. The data first enters the entry flow, then goes through the middle flow which is repeated eight times, and finally through the exit flow. All Convolution and depthwise separable Convolution layers are followed by batch normalization[30].

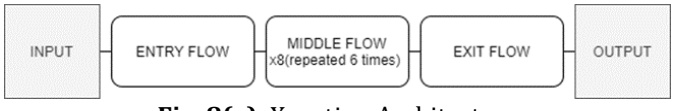


Fig. 8(a). Xception Architecture

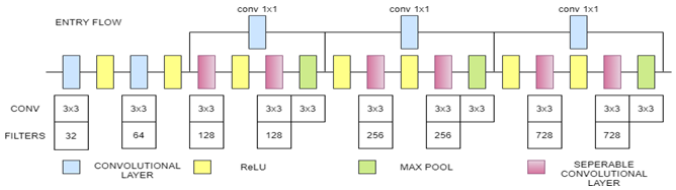


Fig. 8(b). Xception Architecture(Entry flow)

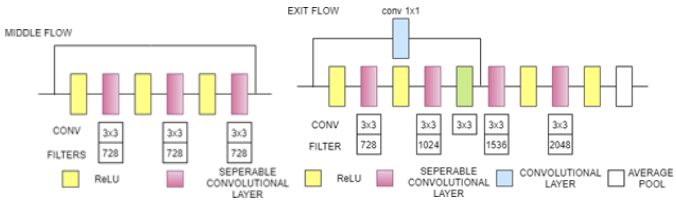


Fig. 8(c). Xception Architecture(Middle and exit flow)

3.9 DenseNet121

DenseNet121 is 121 layers deep, densely connected convolutional network. There are three Sections in the DenseNet121 architecture. The first is convolution block, which is a basic block in the DenseNet architecture. Convolution block in DenseNet is similar to the identity block in ResNet. The second section is the dense block, in which the convolution blocks are concatenated and densely connected. Dense block is the main block in DenseNet architecture[27]. The last is the transition layer, which connects two contiguous dense blocks. The size of the feature maps are the same in the dense block. Transition block reduces the dimension of feature maps. It has approximately 8M parameters.[31]

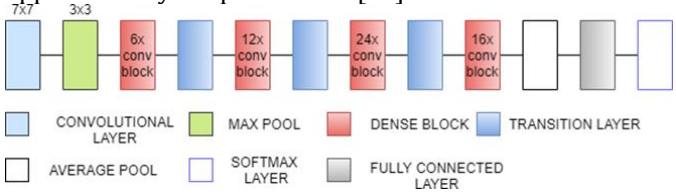


Fig. 9(a). DenseNet121 Architecture

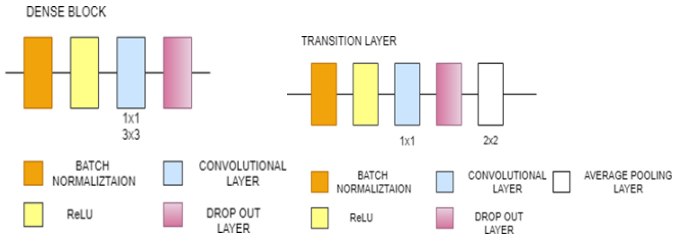


Fig. 9(b). DenseNet121 Architecture(Dense block and transition layer)

3.10 DenseNet169

DenseNet169 is 169 layers deep, densely connected convolutional network. There are three Sections in the DenseNet169 architecture. The first is convolution block, which is a basic block in the DenseNet architecture. Convolution block in DenseNet is similar to the identity block in ResNet. The second section is the dense block, in which the convolution blocks are concatenated and densely connected. Dense block is the main block in DenseNet architecture. The last is the transition layer, which connects

two contiguous dense blocks. The size of the feature maps are the same in the dense block. Transition block reduces the dimension of feature maps. It has approximately 14.3M parameters[31].

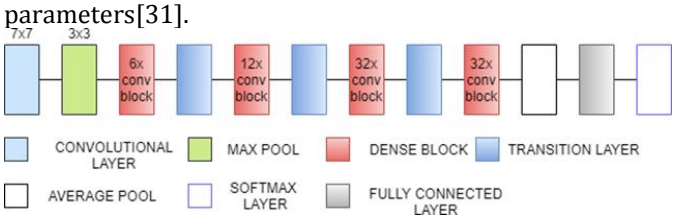


Fig. 10. DenseNet169 Architecture

3.11 DenseNet201

DenseNet201 is 201 layers deep, densely connected convolutional network. There are three Sections in the DenseNet201 architecture. The first is convolution block, which is a basic block in the DenseNet architecture. Convolution block in Densenet is similar to the identity block in ResNet. The second section is the dense block, in which the convolution blocks are concatenated and densely connected. Dense block is the main block in DenseNet architecture. The last is the transition layer, which connects two contiguous dense blocks. The size of the feature maps are the same in the dense block. Transition block reduces the dimension of feature maps. It has approximately 20.2M parameters[31].

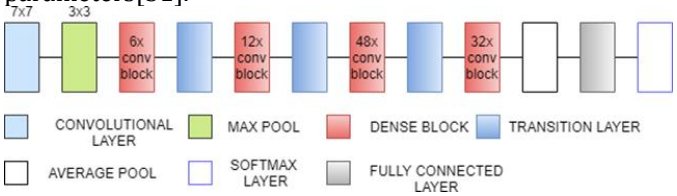


Fig. 11. DenseNet201 Architecture

3.12 MobileNetV2

MobileNetV2 is based on an inverted residual structure and a linear bottleneck layer[32]. There are two types of blocks in MobileNetV2. One is the residual block with stride of 1. Another one is the residual block with stride of 2 for downsizing with each block having three layers. First layer is a 1x1 convolutional layer with ReLU6 [32]. The second layer is the depth wise convolution which uses 3x3 depth-wise separable convolution and the third layer is linear 1x1 convolutional layer. It has 3.4M parameters[33].

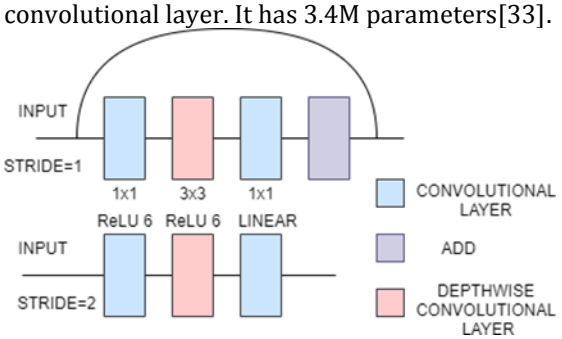


Fig. 12. MobileNetV2 Architecture

3.13 NasNetMobile and NasNetLarge

NasNetMobile is a Scalable Convolutional Neural Network. It consists of a few basic building blocks. Each block consists of a few basic operations that are repeated multiple times according to the required capacity of the network. It has 12 blocks and about 5.3M parameters. The default input size is 224x224. Block is the smallest unit in NASNet. Cell is a combination of blocks[34][33].

NasNetLarge is a convolutional neural network that is trained on more than a million images from the ImageNet database which has over 88M parameters. It consists of two repeated motifs termed as Normal cell and Reduction cell[35]. In normal cells, convolutional cells return a feature map of the same dimension. In reduction cells, convolutional cells return a feature map with the dimensions reduced by a factor of 2[36]. NAS-Neural Network Search is an algorithm that is used to search for the best neural network architecture[37].In the NAS algorithm, controller Recurrent Neural Network (RNN) samples the blocks and puts them together to create end-to-end architecture. The default input size is 331x331.

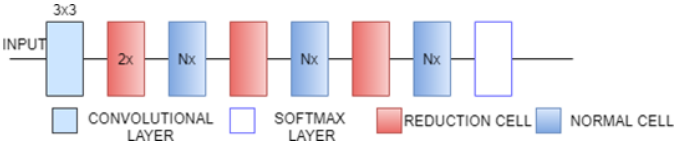


Fig. 13(a). NasNet Architecture

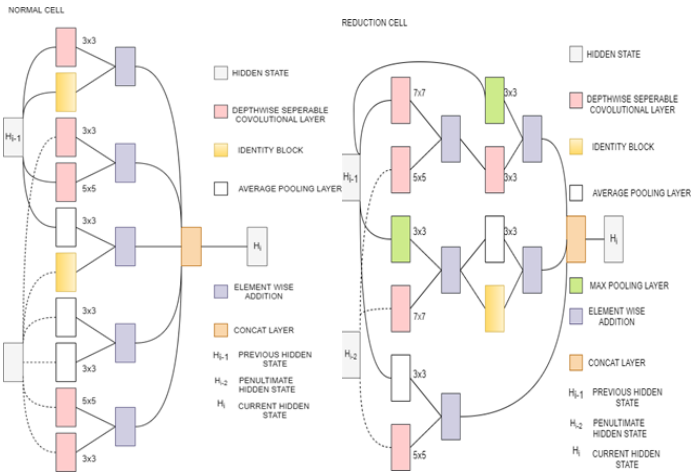


Fig. 13(b). NasNet Architecture(Normal and Reduction cell)

3.14 EfficientNetB0

The EfficientNet group consists of 8 models from B0 to B7. In this model a new scaling method called" Compound Scaling" was introduced. Compound scaling method uses a compound co-efficient to uniformly scale all the three dimensions(depth,width, and resolution) together. This new family of models was created using the NAS (Neural Architecture Search) algorithm[37]. This NAS Algorithm is used to find the optimum baseline network. In all the 8 models, the first (stem) and final layers are common. After this, each model consists of seven blocks. These blocks have varying numbers of sub-blocks whose number is increased as we move from

EfficientNetB0 to EfficientNetB7. EfficientNetB0 has approximately 5.3M trainable parameters[38].

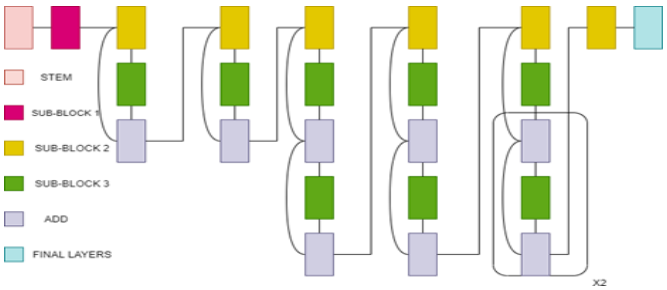


Fig. 14(a). EfficientNetB0 Architecture

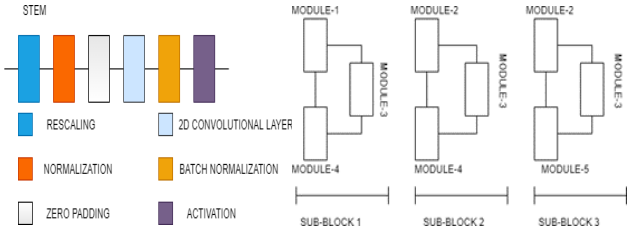


Fig. 14(b). EfficientNetB0 Architecture(Stem and module)

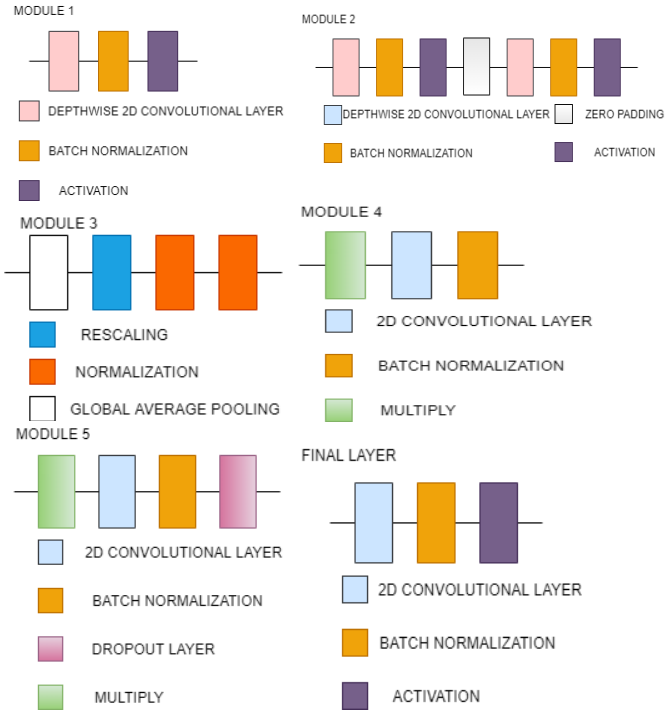


Fig. 14(c). EfficientNetB0 Architecture(Modules and Final layer)

4. METHODOLOGY

4.1 Data Pre-processing –

In our experiments, we evaluate the performance of our CNN transfer learning models on a Medical Image Classification to detect the presence of Tumor in a Brain MRI. The original dataset of MRI Images consisted of a total of 253 images with 153 images with Tumor labelled as 'yes' and 98 images without Tumor labelled as 'no'. These images are obtained from Brain MRI images for tumor detection dataset by Navoneel Chakrabarty[39]. The dataset was built by experienced radiologists using real patient's data. All the images are resized to 150X150. Figure 15(a) and 15(b) show the MRI Images with tumor and without tumor respectively.

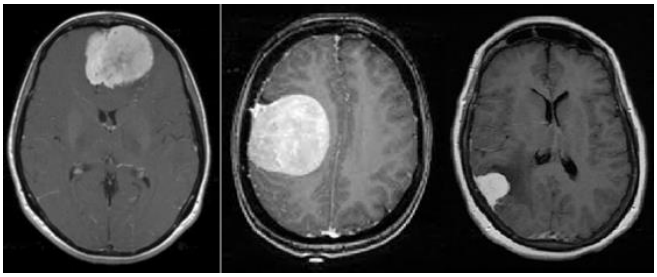


Fig. 15(a). With Tumor

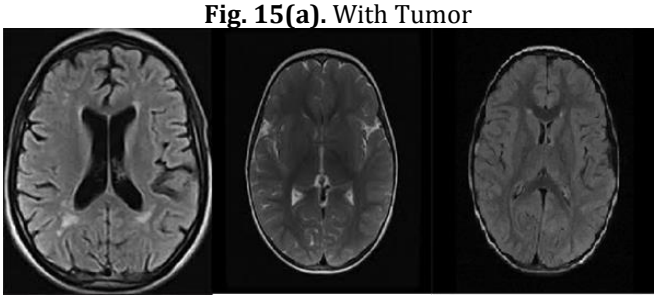


Fig. 15(b). Without Tumor

4.2 Data Augmentation–

Data Augmentation is a technique in which the size of the training dataset is artificially expanded by creating modified versions of images in the dataset which will enhance the capability of the model to learn and generalize better on future unseen data[40]. In order to account for the data imbalance and more number of images, these images were augmented to get a total of 1085 images with Tumor and 979 images without Tumor. Figure 16(a) and 16(b) show the ways in which an image is augmented to account for more number of data. Table 1 gives the details of the dataset before and after data augmentation.

Table 1. Data Analysis

S.no.	Tumor Class	Original Dataset (Total-253)		After Data Augmentation	
		Number	Percentage	Number	Percentage
1	Yes	155	61.26	1085	52.57
2	No	93	38.74	979	47.43
Total		253	100	2064	100

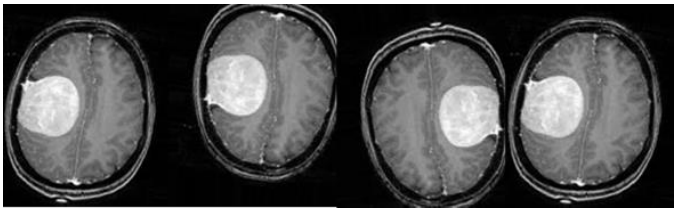


Fig. 16(a). Data Augmentation

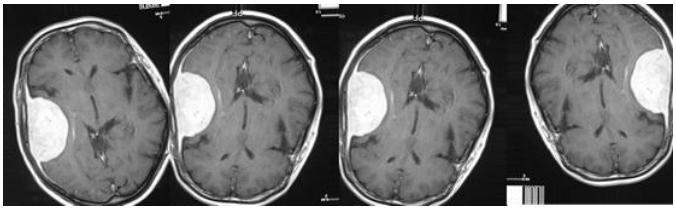


Fig. 16(b). Data Augmentation

4.3 Data Splitting –

After the data has been augmented as per the requirement, the data is then split into the training set and the test set. In this experiment, the Test-Train ratio has been set at 0.2 indicating that 80% (1651 images) of the images will go to the training set which will be used by the neural network to get trained and the remaining 20% (413 images) of the images will go to the test set on which the trained neural network will be applied and to check the validation (test) accuracy of the Neural Network. Table 2 gives the number of images split to the training and the test set.

Table 2. Data Splitting

S.no.	Brain MRI Tumor	Test-Train Split Ratio: 0.2	
		Number	Percentage
1	Training set	1651	80
2	Test set	413	20
Total		2064	100

4.4 CNN Architectures –

Transfer learning is a technique in which a pre-trained model is taken that has already been trained on a related task of Image Classification and reusing the weights in the new model to be trained in this experiment.

The CNN Architectures on which fine tuning of parameters is performed to apply Transfer Learning are given in Table 3.

Table 3. Pre-Trained Models

Architecture Name	Year Published	Number of Parameters
LeNet-5	1998	0.060 M
AlexNet	2012	60 M
VGG16	2014	138.3 M
VGG19	2014	143.6 M
ResNet-50	2015	25.6 M
InceptionV3	2015	23.8 M
InceptionResNet V2	2016	55.8 M
ResNet152V2	2016	60.3 M
Xception	2016	22.9 M
DenseNet121	2017	8 M
DenseNet169	2017	14.3 M
DenseNet201	2017	20.2 M
MobileNetV2	2018	3.5 M
NasNetMobile	2018	5.3 M
NasNetLarge	2018	88.9 M
EfficientNetB0	2019	5.3 M

For this experiment, we decided to keep all the major parameters such as the optimizer, loss function constant for all the CNN Architectures so that the results can be compared on equal grounds.

The following are the hyperparameters that have been used to train all the Transfer Learning models used in this Brain MRI Tumor Image Classification.

Weights: imagenet

Optimizer: SDG (Stochastic Gradient Descent) with Learning Rate of 0.001

Loss Function: Binary Cross Entropy

Metrics: "acc"

4.5 Layers Used–

1. Output Layer of the Pre-trained Model (VGG19, InceptionV3 etc.)
2. Dropout Layer with probability of 0.3
3. Flatten Layer
4. Dropout Layer with probability 0.5
5. Dense Layer with 128 units and ReLU activation function
6. Dense Layer with 1 unit and Sigmoid activation function

4.4 Training and Validation Generator details -

Image Size: (150,150,3) *

Batch size: 64

Class Mode: Binary

Number of Epochs: 30

* NasNetMobile required an image size of (227,227,3) and NasNetLarge required an image size of (331,331,3)

4.7 Transfer Learning Model Architecture

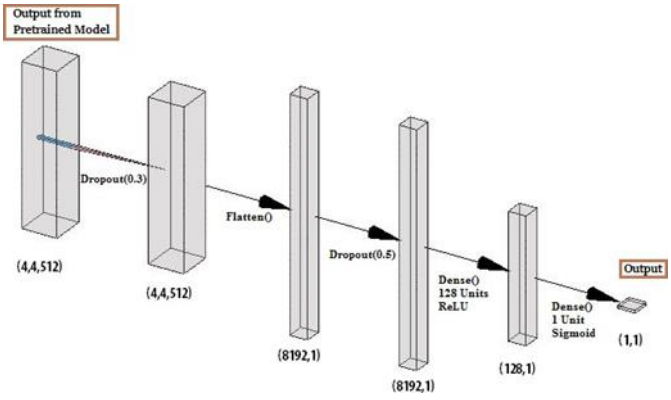


Fig. 17. Transfer Learning Model Architecture

5. RESULTS

These pre-trained models were used to classify the Brain MRI images to detect the presence of tumor in it. All the models were trained for 30 epochs and the results are tabulated below in Table 4.

Table 4. Results

CNN Architecture	Validation Accuracy (%)	Validation Loss	Training Accuracy (%)	Training Loss
DenseNet121	94.18	0.19	96.42	0.08
InceptionV3	92.73	0.28	98.18	0.04
Xception	92.73	0.29	98.84	0.03
ResNet152V2	91.76	0.36	99.33	0.02
NasNetMobile	91.28	0.49	99.15	0.02
VGG19	90.55	0.24	86.73	0.3
DenseNet201	90.55	0.37	99.03	0.02
MobileNetV2	90.31	0.4	99.45	0.02
NasNetLarge	90.31	0.49	100	0
DenseNet169	89.83	0.47	97.81	0.05
InceptionResNetV2	87.89	0.35	96.97	0.07
AlexNet	87.89	0.42	82.98	0.38
VGG16	87.16	0.31	87.4	0.28
ResNet-50	77.23	0.52	68.07	0.59
LeNet-5	72.88	0.43	86.67	0.31
EfficientNetB0	52.3	1.3	89.7	0.24

From the above tabulated results, it can be seen that the DenseNet121 with 8M parameters has the highest validation accuracy of 94.18% among all the other models that were built with the same hyperparameters and trained on the same Medical Image dataset of Brain MRI Scan images. InceptionV3 and Xception models also show accuracies of 92.73% on the validation set but the higher training accuracies denote that there is a tendency towards over fitting. The ResNet152V2 and NasNetMobile show validation accuracies of 91.76% and 91.28% respectively. The training accuracies of above 99% in both the models also denotes the possibility of overfitting. The VGG19 and DenseNet201 architectures have an accuracy of 90.55% on the validation set but the training set accuracy of the VGG19 architecture is 86.73% which denotes that more training is required to achieve higher accuracy. On the other hand, DenseNet201 has the training accuracy of 99.03% which shows overfitting. The MobileNetV2 and NasNetLarge architectures have validation accuracies of 90.31% and have high training accuracies of 99.45% and 100% respectively. This clearly denotes that the model has over fit the training data and can be rectified by fine-tuning the hyperparameters. The DenseNet169 and InceptionResNetV2 architectures have shown a validation accuracy of 89.83 and 87.89 respectively which can also be improved by fine tuning the hyperparameters. The AlexNet architecture with over 60M parameters is one of the old architectures developed which performed well with a validation accuracy of 87.89% and training accuracy of 82.98%. The VGG16 model also gave a validation accuracy of 87.16% which can be improved by fine-tuning the hyperparameters. The ResNet-50 model showed a very poor training accuracy of 68.07% and hence produced a lower validation accuracy of 77.23%. This means that the hyperparameters need to be fine-tuned to improve the training accuracy which will improve the validation set accuracy. The LeNet-5 Architecture which was developed in 1998 managed to give a validation accuracy of 72.88% and a training accuracy of 86.67%. Though this model's architecture has a very low number of parameters, it managed to learn the features of the training classes to produce reasonable accuracies. The latest EfficientNetB0, which was published in 2019 failed to achieve a good validation accuracy and suffered from heavy overfitting on training with the above mentioned hyperparameters. The validation accuracy remained very low at 52.30% while the training accuracy was 89.70%. This can be corrected by adjusting the hyperparameters to prevent overfitting.

The results of these pre-trained models with transfer learning are available at https://github.com/mk-gurucharan/Transfer-Learning-Methods for reference.

The Bar plot of the accuracies and losses of the various CNN architectures are shown in Figure 18 and 19 respectively.

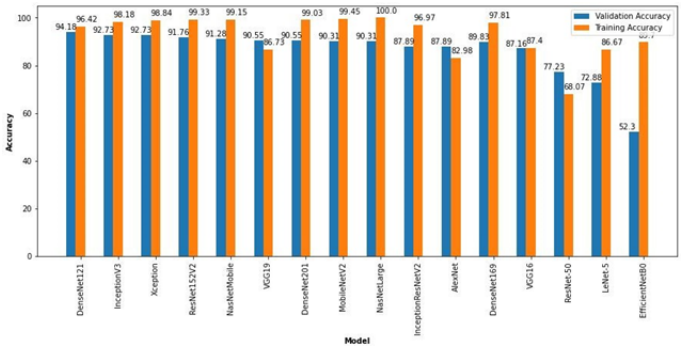


Fig 18. Accuracy Bar Plot

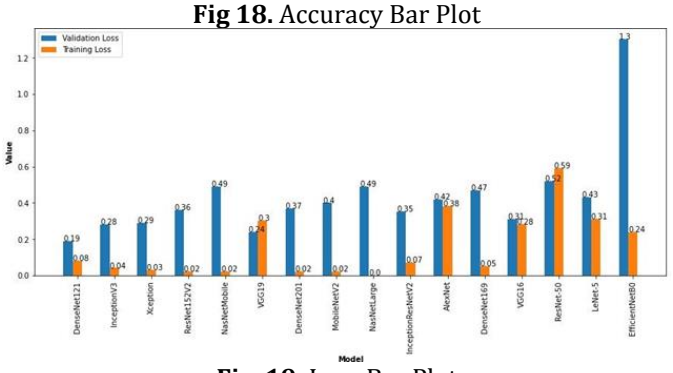


Fig. 19. Loss Bar Plot

6. CONCLUSION AND FUTURE SCOPE

In this paper, sixteen pre-trained CNN models have been built using Transfer Learning and have been used to classify the Brain MRI Images if they have a Tumor or not. The Data is pre-processed with a standard size and augmented to increase the size of the training set to prevent overfitting. Among all the models, keeping the various hyperparameters such as optimizer and loss function constant, DenseNet121 showed the highest validation accuracy of 94.18% with the lowest validation loss of 0.19. The other models such as

Xception, InceptionV3 and ResNet50 also performed well with accuracies higher than 90%. Some of the models such as NasNetLarge and EfficientNetB0 suffered from overfitting. From these results, the pre-trained models that can be used by implementing Transfer Learning for Medical Image Classification on a small number of images can be comprehended. The future possibility could be to use a larger dataset to control the occurrence of overfitting. Additionally, extensive tuning of the hyperparameters such as optimizer and loss on the models can be performed to result in a high accuracy for Medical Image Classification with Deep Learning.

5. REFERENCES

[1] J. G. Carbonell, "Machine learning research," ACM SIGART Bull., vol. 18, no. 77, pp. 29–29, 1981, doi: 10.1145/1056743.1056744.

[2] N. Jmour, S. Zayen, and A. Abdelkrim, "Convolutional neural networks for image classification," in 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET), 2018, pp. 397–402, doi: 10.1109/ASET.2018.8379889.

[3] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," 2016, [Online]. Available: http://arxiv.org/abs/1603.04467.

[4] T. Zhou, S. Ruan, and S. Canu, "A review: Deep learning for medical image segmentation using multi-modality fusion," Array, vol. 3–4, no. August, p. 100004, 2019, doi: 10.1016/j.array.2019.100004.

[5] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," Artif. Intell. Rev., pp. 1–70, 2020, doi: 10.1007/s10462-020-09825-6.

[6] M. A. Wani, F. A. Bhat, S. Afzal, and A. I. Khan, "Basics of Supervised Deep Learning," pp. 13–29, 2020, doi: 10.1007/978-981-13-6794-6_2.

[7] Y. Zhu et al., "Heterogeneous transfer learning for image classification," 2011.

[8] M. Lagunas and E. Garces, "Transfer Learning for Illustration Classification," 2018, doi: 10.2312/ceig.20171213.

[9] K. E. Emblem et al., "Predictive modeling in glioma grading from MR perfusion images using support vector machines.," Magn. Reson. Med., vol. 60, no. 4, pp. 945–952, Oct. 2008, doi: 10.1002/mrm.21736.

[10] M. M. Beno, V. I. R, S. S. M, and B. R. Rajakumar, "Threshold prediction for segmenting tumour from brain MRI scans," Int. J. Imaging Syst. Technol., vol. 24, no. 2, pp. 129–137, Jun. 2014, doi: 10.1002/ima.22087.

[11] E. A. S. El-Dahshan, H. M. Mohsen, K. Revett, and A. B. M. Salem, "Computer-aided diagnosis of human brain tumor through MRI: A survey and a new algorithm," Expert Syst. Appl., vol. 41, no. 11, pp. 5526–5545, 2014, doi: 10.1016/j.eswa.2014.01.021.

[12] M. Rahmani and G. Akbarizadeh, "Unsupervised feature learning based on sparse coding and spectral clustering for segmentation of synthetic aperture radar images," IET Comput. Vis., vol. 9, no. 5, pp. 629–638, 2015, doi: 10.1049/iet-cvi.2014.0295.

[13] M. Lai, "Deep Learning for Medical Image Segmentation," 2015, [Online]. Available: http://arxiv.org/abs/1505.02000.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9351, pp. 234–241, 2015, doi: 10.1007/978-3-319-24574-4_28.

[15] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger,"3D U-net: Learning dense volumetric segmentation from sparse annotation," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9901 LNCS, no. June 2016, pp. 424–432, 2016, doi: 10.1007/978-3-319-46723-8_49.

[16] P. D. Chang, "Fully Convolutional Deep Residual Neural Networks for Brain Tumor Segmentation," in Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries, 2016, pp. 108–118.

[17] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, "Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images," IEEE Trans. Med. Imaging, vol. 35, no. 5, pp. 1240–1251, 2016, doi: 10.1109/TMI.2016.2538465.

[18] S. Iqbal, M. U. Ghani, T. Saba, and A. Rehman, "Brain tumor segmentation in multi-spectral MRI using convolutional neural networks (CNN)," Microsc. Res. Tech., vol. 81, no. 4, pp. 419–427, Apr. 2018, doi: 10.1002/jemt.22994.

[19] P. Saxena, A. Maheshwari, S. Tayal, and S. Maheshwari, "Predictive modeling of brain tumor: A Deep learning approach," 2019, [Online]. Available: http://arxiv.org/abs/1911.02265.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Ha, "LeNet," Proc. IEEE, no. November, pp. 1–46, 1998, doi: 10.1109/5.726791.

[21] A. El-Sawy, H. EL-Bakry, and M. Loey, "CNN for Handwritten Arabic Digits Recognition Based on LeNet-5 BT - Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016," 2017, pp. 566–575.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Handb. Approx. Algorithms Metaheuristics, pp. 1–1432, 2007, doi: 10.1201/9781420010749.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., pp. 1–14, 2015.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[25] M. Mateen, J. Wen, Nasrullah, S. Song, and Z. Huang, "Fundus image classification using VGG-19 architecture with

PCA and SVD," Symmetry (Basel)., vol. 11, no. 1, 2019, doi: 10.3390/sym11010001.
[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016- Decem, pp. 2818–2826, 2016, doi: 10.1109/CVPR.2016.308.
[27] Q. Ji, J. Huang, W. He, and Y. Sun, "Optimized deep convolutional neural networks for identification of macular diseases from optical coherence tomography images," Algorithms, vol. 12, no. 3, pp. 1–12, 2019, doi: 10.3390/a12030051.
[28] X. Xia, C. Xu, and B. Nan, "Inception-v3 for flower classification," in2017 2nd International Conference on Image, Vision and Computing (ICIVC), 2017, pp. 783–787, doi: 10.1109/ICIVC.2017.7984661.
[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," 31st AAAI Conf. Artif. Intell. AAAI 2017, pp. 4278–4284, 2017.
[30] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 1800–1807, 2017, doi: 10.1109/CVPR.2017.195.
[31] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 2261–2269, 2017, doi: 10.1109/CVPR.2017.243.
[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 4510– 4520, 2018, doi: 10.1109/CVPR.2018.00474.
[33] F. Saxen, P. Werner, S. Handrich, E. Othman, L. Dinges, and A. Al- Hamadi, "Face attribute detection with mobilenetv2 and nasnet- mobile," Int. Symp. Image Signal Process. Anal. ISPA, vol. 2019-Septe, no. October, pp. 176–180, 2019, doi: 10.1109/ISPA.2019.8868585.
[34] K. Radhika, K. Devika, T. Aswathi, P. Sreevidya, V. Sowmya, and K.
P. Soman, Performance analysis of NASNet on unconstrained ear recognition, vol. SCI 871, no. January. Springer International Publishing, 2020.
[35] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 8697–8710, 2018, doi: 10.1109/CVPR.2018.00907.
[36] B. Zoph, V. Vasudevan, J. Shlens, and Q. V Le, "Zoph_Learning_Transferable_Architectures_CVPR_2018_paper.pdf," Proc. IEEE Conf. Comput. Vis. pattern Recognit., pp. 8697–8710, 2018, [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/papers/Zoph_Learning
_Transferable_Architectures_CVPR_2018_paper.pdf.
[37] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc., pp. 1–16, 2019.
[38] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 36th Int. Conf. Mach. Learn. ICML 2019, vol. 2019-June, pp. 10691–10700, 2019.
[39] "Brain MRI Images for Brain Tumor Detection | Kaggle." https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor- detection (accessed Jul. 03, 2020).
[40] M. A. Tanner and W. H. Wong, "The Calculation of Posterior Distributions by Data Augmentation," J. Am. Stat. Assoc., vol. 82, no. 398, pp. 528–540, Jul. 1987, doi: 10.2307/2289457.