Classification of Cervical Cancer using various Deep Learning Algorithms

ABSTRACT

‘Cervical cancer’ is one of the highly prevalent ailments in women ranking fourth in worldwide, mostly occurring in less-developed countries. This is perceived when certain vagaries occur in a woman's cervix. These cancer cells can also spread to other vital organs like lungs, liver and bladder which complicates the problem. Previous discoveries, tests and careful monitoring showed high levels of recovery rates at early detection of cancerous cells. But distinguishing cervical cells in ‘Pap-smear’ is demanding piece of work due to a few constraints. Some of the constrictions include complications of the ‘morphological’ changes in the physical elements of the cells. Although there are two methods to obtain the cells which are the process of Colposcopy and pap-test but, Pap-smear test are most favoured due to low cost and pain free diagnosis. This paper presents deep learning classification methods applied on the SIPAKMED ‘papilloma-smear’ image dataset in order to establish a benchmark point for the assessment of forthcoming classification techniques. With ResNet-152 highest classification accuracy was 94.89%. It is evidently proven that early detection increased the probability of proper treatment of the cancerous cells and thereby increasing the chances of survival of the patient. This paper aims to put a step in the direction of using Deep Learning in diagnostic procedures.

Keywords— Pap-Smear, Deep Transfer Learning, Classification Algorithm, Resnet-50, Vgg-16, cancerous cells, SIPAKMED

Table of Contents

Table of Tables

CHAPTER -1 INTRODUCTION

Cancer is the most apprehensive and widespread disease these days, with more than ten million people dying every year. Taking into consideration, worldwide health affect, cervical cancer stands in the 2nd position after breast cancer which occurs in the region of cervix due to irregular augmentation of cells [1]. Roughly 1500 people die because of cancer every day according to some statistics.

There are several peril circumstances. Around 90% of cervical cancers are predominantly triggered by 'Human Papilloma virus' (HPV) of types 16 and 18 which are generally sexually transmitted and cause warts. If these warts and blisters are left unattended may make the cells malignant. About half a million women worldwide are recognized with this type of cancer and more than 0.28 million women's lives are determined with cervical cancer each year [2]. Cervical cancer is ranked first among other cancers in Tripura, a North-East Indian state of India corresponding to the central sites of cancer [3]. Cervical cancer prevents cells in the cervical region from performing their normal functions and building tissue that kills normal surrounding cells [4]. Normal cells do not quickly turn into cancer instead, it grows gradually into premature mutations that turn into cancer [5]. The death rate of cervical cancer is higher in rural areas compared to urban areas. Various cancer prevention programs have been developed by the Indian government, but these have not had much impact on rural areas due to issues related to lack of awareness, poor health amenities, lack of framework, unskilled workforce and quality assurance. There has been a gradual/ steady crusade in opposition of cervical cancer for the last 35 years in India, but this has had slight influence on the illness and death from the disease [6].

When all the preventive procedures and treatment are given, the chances of maintaining or supporting oneself become possible. But still one cannot say that a sustained fight is stopped, as many cases outline that cancer has a possibility of occurring again. Person surviving from cancer become weaken as it directly affects its immune system.

This is where the arrival of artificial intelligence plays it role. Investigators and data Scientists are trying to enlarge the gain of its implementation in all possible fields, from weather prognosticating to legislation. Apparently in this large set of implementations, the medical sector promotes a large part of it. Machine and Deep-learning principles have publicized their efficiency in a range of medical imaging modalities such as MRI scan, X-Rays, CT scan, etc. Cervical cancer is curable if detected in early stage with proper medication [7].

The intention of this project report is to prepare a classification model on several deep learning algorithms to categorize the cells among various level/stages of cancerous cell growth. Since we suggested to use deep learning models so there is no need for feature extraction step as deep learning model is to learn appropriate internal depiction by design, given raw input RGB values. The dataset chosen is from one of the latest studies (discussed in the following sections) in this field and hence is more fitting with current environmental and genetic changes when compared to datasets which were prepared over a decade ago. Although an ample research have been done with older datasets, but this dataset being more relevant has not been much examined for its functioning with Deep Learning Algorithms.

CHAPTER-2 LITERATURE REVIEW

The purpose of this project is to figure out whether the sample comprises any cancer traits or not. Previous work [8] done on this kind of project employed K-NN (K Nearest Neighbours), a machine learning algorithm for cell classification on Pap smear with the percentage accuracy achieved over 82%. Another work presented by T. Chankong et al., [9] achieved 93.78% accuracy for classification using ANN (Artificial Neural Network). R. Kumar et al. [10] proposed a framework for classifying cervical cells images using KNN and obtained the accuracy of about 92%. More related work on Pap Smear is illustrated in the form of table below at Table.

Table 1 Literature Review

Author/Year	**Method**	**No. of images**	**Image resolution**	**Accuracy**
M.E. Plissiti et. al./ 2006 [11]	Morphological operator, deformable model	Not provided	2048x1536 pixels	94.15%
Rahmadwati et. al./ 2011 [12]	Median filter, K-means clustering, Matching methods, feature extraction	475 labelled cervical biopsy images	4080x3072 pixels	89%
Marina E. Plissiti et al./ 2018 [13]	VGG-19	966 images	2048x1536 pixels	95.35

CHAPTER-3 METHODOLOGY

The illustrated block-diagram (Figure 1) explains the pipeline of the course of implementation. The 1st objective as to obtain a newer dataset which is more relevant to all the environmental conditions. The said dataset was created in Greece and was publicly available. The source is cited in the references. After the dataset was procured the next step was to divide it into training and testing subsets. The ration of split between training and testing was 8:2. Out of 966 total images, 196 cluster images were put into testing subset and the rest were used for the training process.

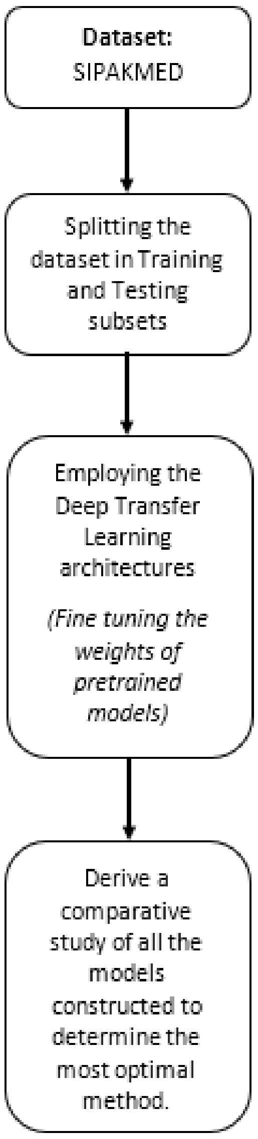

Dataset: SIPAKMED
Splitting the dataset in Training and Testing subsets
Employing the Deep Transfer Learning architectures *(Fine tuning the weights of pretrained models)*
Derive a comparative study of all the models constructed to determine the most optimal method.

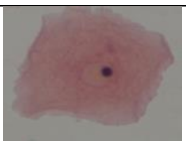
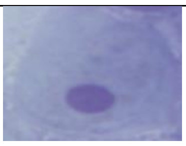
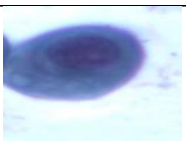
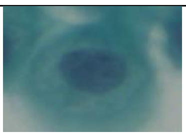
Figure 1 Methodology

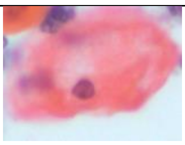
Residual Network-50 was the first model that was considered for the objective. The results obtained were encouraging (these results are explained in the next sections of the paper). Before employing other models, tweaking of some parameters was done manually initially, in the hope to get some improvement. It is imperative to note that learning rate was cyclically varied to acquire the highest performance.

A. Dataset

The publicly available pap-smear SIPaKMed Database consist of 966 manually cropped cluster images from the 4049 isolated cells images. These images were captured through a camera that used charge coupling with infinity of 1 Lumenera distributed on 5 different classes of cervical cells. The 5 types of sets can be bifurcated into further two sub classes, i.e., Normal cells and Abnormal cells.

Table 2 Five types of classes for single pap-smear cells. (to fit the table, some images are slightly scaled)

Class	Cell type	Cell image	No. of Images	No. of Cells
1	Superficial/ Intermediate (Normal)		126	813
2	Parabasal (Normal)		108	787
3	Koilocytotic (Abnormal)		238	825
4	Metaplastic (Abnormal)		271	793

5	Dyskeratotic (Abnormal)		223	813
Total			966	4049

a) **Normal cells** are the squamous epithelial cells defined by their location in the epithelium layers and their growth level. Superficial-Intermediate and Parabasal comes under this category.

i. Superficial-Intermediate cells: These types of cells are most found in Pap test and are usually flat with round or oval shape with typically cynophile or eosinophilic. They are found with central thickening or condensation nucleus. Refer figure number 2.

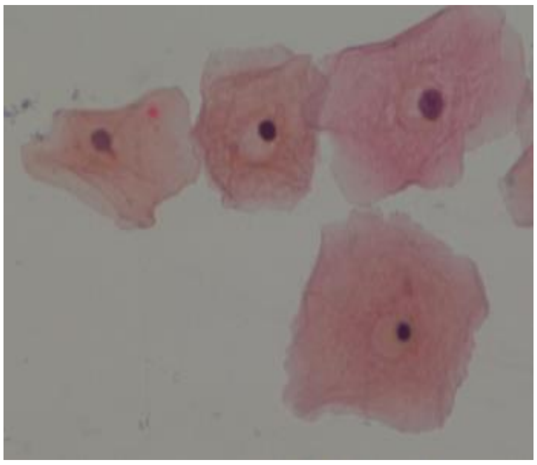


Figure 2 Superficial-Intermediate cells

ii. Parabasal cells: Parabasal cells are exceedingly small and undeveloped epithelial cells that appear in a normal smear of the vagina. This can be seen in Fig.3. The cytoplasm is usually cyanophilic and generally comprises of large vesicular nucleus. Parabasal cells are similar to cells identified as ‘metaplastic cells’ in terms of structural morphology and appearance, hence there is difficulty in differentiating them.

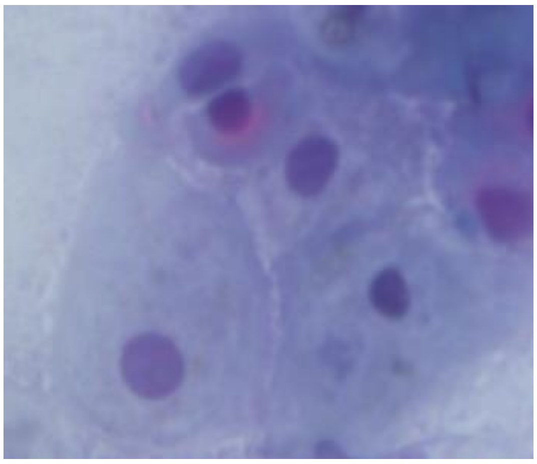


Figure 3 Parabasal cells

b) **Abnormal Cells** are illustrated by inflectional changes in their operational parts signifying the existence of pathological circumstances.

i. **Koilocytotic cells:** Koilocytotic cells are most resembling the mature squamous cells (intermediate and superficial) and can also relate to metaplastic type koilocytic cells time to time (Fig. 4). It is often seen having an affinity for blue or green dyes cyanophilous tissues distinguished by a larger perinuclear cavity which are lightly stained. The circumference of the cytoplasm is extremely 'narrow'. Koilocyte's nuclei are generally enlarged, overgrown, looks darker than normal when examined under the microscope and exhibit abnormal nuclear membrane contour. In many cases there are cells which contain two or higher number of cellular nuclei. Koilocytic cells are characteristic for a HPV infection and the nuclei of koilocytotics commonly shows several degrees of degeneration, depending at the various stages of affliction and also of the different virus form of infection.

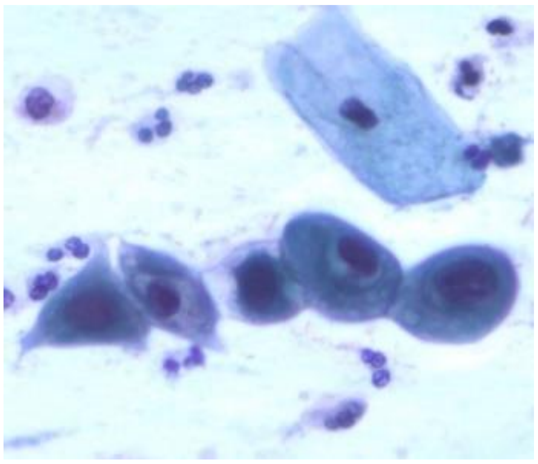


Figure 4 Koilocytotic cells

ii. Dyskeratotic cells*:* Dyskeratotic cells of squamous cells receive overhasty change into a form containing keratin within singular cells or much more commonly in 3-dimensional clusters (Fig. 5). They are visually graphic ‘orangeo-philic’ cytoplasm. They are distinguished through the presence of vesicular nuclei, which are resembles with koilocytotic cells. They combine to form an important feature of HPV infection, and from time to time even in the complete non-existence of koilocytes, can be evidence of pathognomonic. They are thick under standard situation and are in three-dimensional clusters which makes it challenging to separate either the boundaries of nucleus or cytoplasm

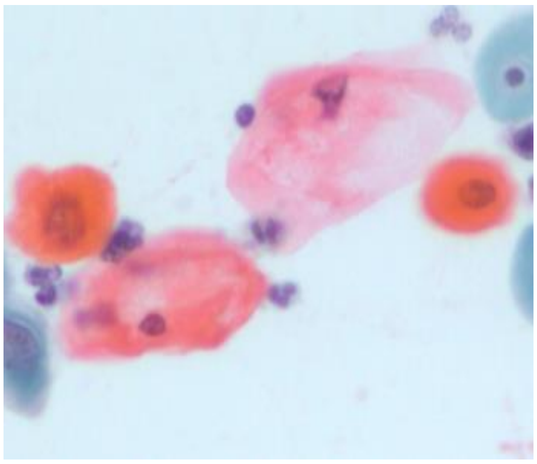


Figure 5 Dyskeratotic cells

iii. Metaplastic cells*:* Metaplastic cells are comparatively immensely large or tiny immature type of squamous cells with conspicuous cellular borders which many times unveil eccentric nuclei and often comprise of large vacuole located within the cells (Fig. 6) The blemish in the center of metaplastic cells under normal condition is observed to be light brown and it differs many a times from its marginal portion. It has been unearthed that there is mainly a darker-stained cytoplasm unveiling considerable conformity of shape and size when differentiated with parabasal cells because of its almost well-defined oval shape of cytoplasm. Their existence in Pap test relates to higher identification rates of pre-malignant lesions (HSIL).

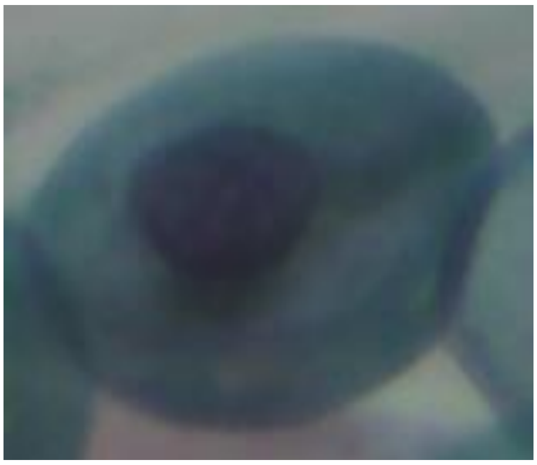


Figure 6 Metaplastic cells

B. SPILITING DATA

The overall figure of normal and abnormal cluster images of cells in the procured data set are 234 and 732, respectively. The whole data-set was divided in the ratio of 60:20:20 for training-set, validation-set and testing respectively. After the splitting there were 576 images in training ,194 images in validation and 196 images in the test subset.

C. DEEP LEARNING

Deep learning is an improvised version of machine learning which operates on the concept of artificially constructed neural networks deriving inspiration from neural networks and synapses of a human brain. These networks replicated the nerve impulse of a neuron using mathematical computations depending on where some conditions are met or not. A fundamental block of deep learning is a perceptron, an entity which takes multiple inputs and projects an output which relies on the activation and threshold put on any particular perceptron. Perceptrons are fundamentally binary classifiers (i.e., they could categorize random data into two classes). Such models have plentiful layers of perceptrons and hence they are evaluated as deep. There is high utilization of deep learning in image processing applications. In machine learning feature extraction/ acquired from images are required to be done manually, while deep learning purges this need. In earlier layers in a deep model simpler features like edges are extracted while the deeper layers could extract much more complex features such as identifiable objects.

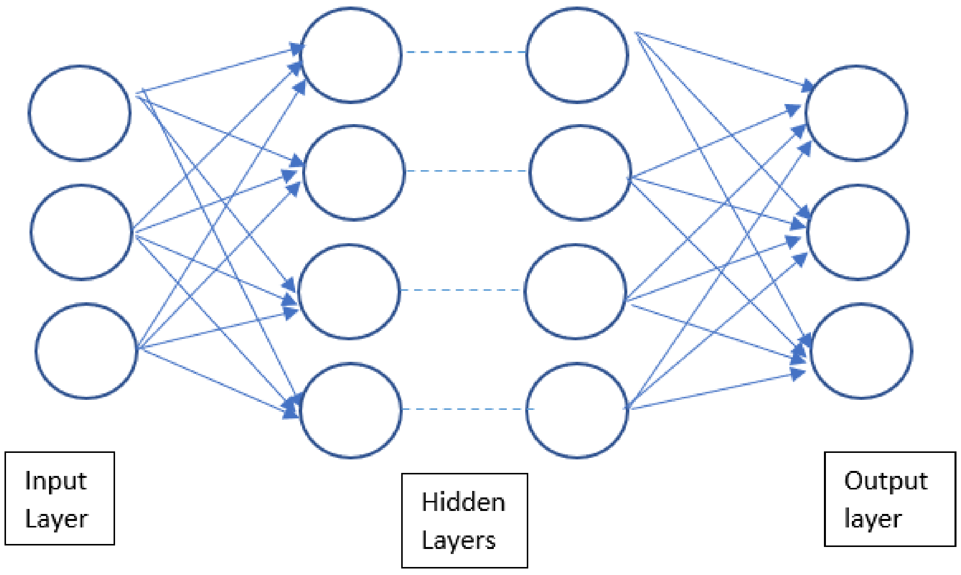


Figure 7 Deep Neural Network

Dense Layers

Dense layers in a neural network model are fully connected layers where a neuron in this layer amasses input incoming from all the perceptrons of the preceding dense layer. Essentially every neuron layer is fully interconnected and learns all the features from the complete set of features of the previous layer.

Convolution layers

Convolution layers are the essential components that make up any neural network architecture. Convolution is the means of directing a filter to an input image or simply a 2-dimensional array. When this method is reiterated by applying the filter on systematically changing parts of the same input, a feature map is obtained. A feature map indicates the position and intensity of the considered feature in the input image. In any convolution layers multiple such filters are applied, also parallelly stacked, on the input image. This in turn builds numerous feature maps that are stacked or piled over each other. The elements of the filter are multiplied with the image patch with dimensions same as those of the filter using dot product. The values obtained from one such product is accumulated and this resultant value constitutes to a single element of the resultant two-dimensional feature map.

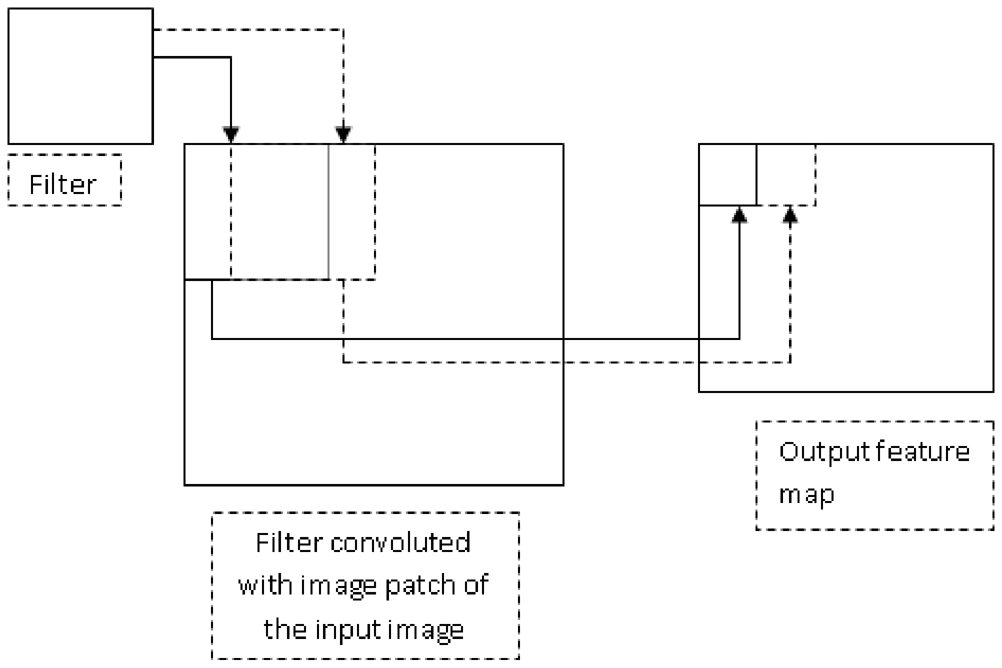


Figure 8 Convolution operation on image matrix

If multiple such features are convoluted with the image, then stacked two-dimensional output feature maps of the same size will be obtained. Each convolution multiplication is done by a distinct neuron of the layer. The input – output relationship in a convolution operation can be understood using the following mathematical formula:

$$\text{OutputDimension}=[(\backslash\{\text{InputImageDimension}-\text{FilterDimension}+2\times(\text{ZP})\backslash\}\div\text{Stride})+1] \qquad (1)$$

Where 'Z.P.' stands for zero-padding. The value of Z.P. could be either zero when there isn't any zero-padding done on the input image or the number of row/columns of zeros that have been padded on to the input image before the convolution.

Max Pooling Layer

The kernels of this layer find the maximum of all the pixel values of the image patch of the input image on which the filter is applied. Down sampling of the results is done, this way the features which are most present are highlighted. Pooling layers are not considered when layers of any architecture are counted. This block is used for dimensionality reduction of each feature map without losing important features. Thus, they are usually used after some convolution layers in shallow networks and even complex networks. For example, if a stack of 96 feature maps of size 55 ×55 each, is input to a max pooling layer of dimensions 3 by 3 pixels with a stride of 2 and no zero padding will result in a stack of feature maps of size 27 × 27. Therefore, this layer is essential for the removal of excessive connections and trainable parameters without losing image information.

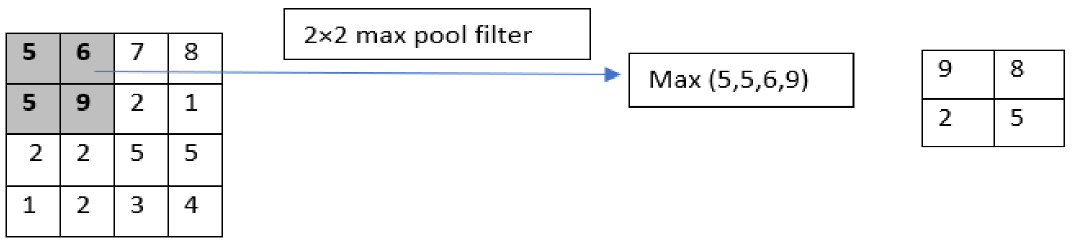


Figure 9 Max-Pool Layer

Average pooling Layers

Average pooling layers find the average of all the pixel values under computation. As compared to max pool layers, average pool layers retain information which may be deemed unimportant. Average pooling as its name keeps the average of that information. It is used for preserving smother features while max pool layers retain more prominent features like edges.

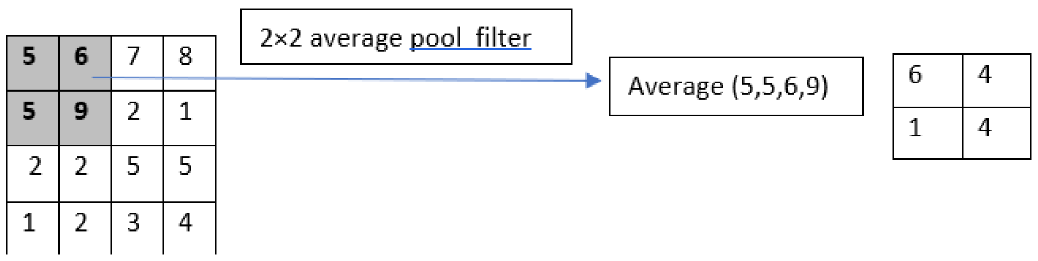


Figure 10 Average Pool Layer

Activation Functions

Activation functions are transfer functions that determine whether and what output is fired from a neuron. The output fired from the neuron is mapped between a certain range of values depending upon the application for which it is used, and which specific activation function type has been used. Activation functions can be subclassified into two classes: Linear and Non-linear functions. The linear activation function does not map the output in any restricted range.

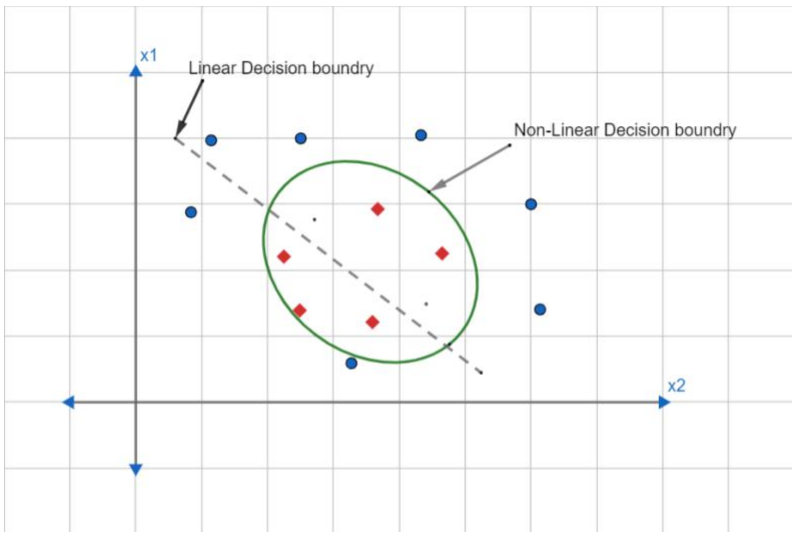


Figure 11 Linear and Non-Linear Activation

'Activation functions' shown in figure 11, that are 'Non-Linear' are much more popularly used and applied on a dataset in which the 'data-points' aren't linearly separable. Examples of Non-linear activation functions are: -

Sigmoid Function

Sigmoid function is mathematically represented as: -

$$g(x) = \frac{1}{1 + e^{-x}} (2)$$

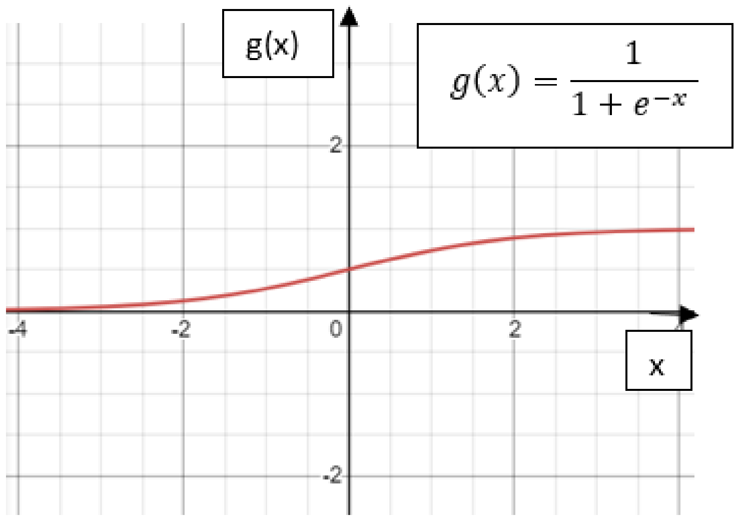


Figure 12 Sigmoid Activation Function

Sigmoid function is the most commonly used activation function. It maps the output between 0 and 1 and is therefore used whenever a mathematical probability is needed to be predicted. By nature, sigmoid activation function is differential and monotonic function whereas its derivative is not monotonic and lies between values 0 and 0.25. The disadvantage of using sigmoid is the increase in time required for training.

Tangent Hyperbolic Function/ Tanh function

Hyperbolic Tangent function is also monotonic and derivative in nature. Like sigmoid function its derivative is not monotonic and lies between values 0 to 1. Output from a Tanh function is mapped between -1 to 1. Mathematical expression of Tanh function is given by: -

$$g(x) = tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} (3)$$

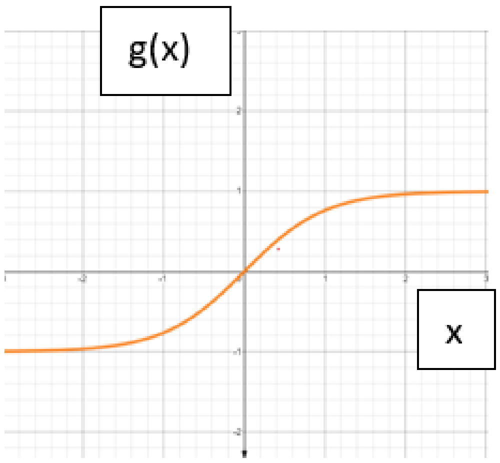


Figure 13 Tan Hyperbolic Activation Function

ReLU Activation Function

Rectified Linear Activation Function is a piece wise linear function. The input is directly passed to the output when the input is positive and when input is negative the output will be mapped to zero. ReLU is preferred in many application of neural network due to various reasons such as, it could provide good performance while being easier to train as well as requiring lesser time to do so. The drawback of sigmoid and tangent hyperbolic functions is that the mathematically saturate to either 0 or +-1. Also, their sensitivity for changes occurring around their median points is minute. Saturation at the output puts restrictions on adaptability of a model to changing weights and thereby limits the improvement of the performance. ReLU activation is mathematically represented as: -

$$g(x) = \begin{cases} xwhenx \geq 0 \\ 0whenx < 0 \end{cases}$$

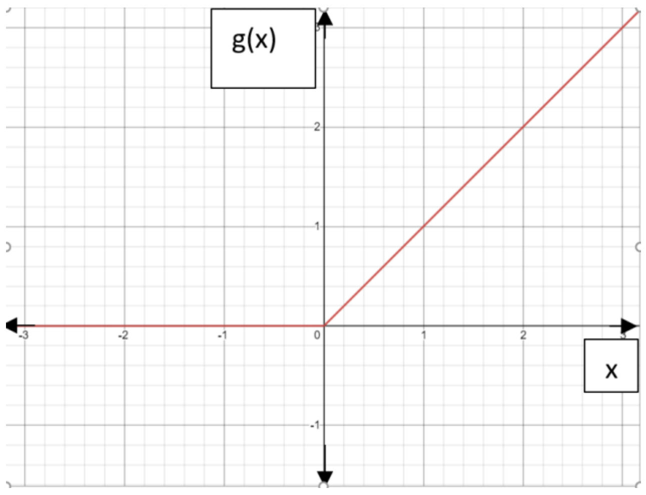


Figure 14 ReLU Activation Function

Soft-max function

While ‘Sigmoid’ and ‘Tangent hyperbolic’ functions work primarily for binary classification of data, Softmax operator which is usually put on the output/ final layer of the entire model converts an array of numbers to an array of probabilities which are relatively proportional to the scale of the array of numbers. When the categories are ‘mutually exclusive’ only then Soft-max function should be used. Soft-max function is mathematically represented as: -

$$\sigma(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

where x is the input vector x_i is the i^{th} element of the vector. The exponential function of any element of input vector is divided by the summation of exponential functions of all the elements of the input vector. This denominator is called the regularization factor. It ensures the restriction of the sum of all outputs to one such that a probabilistic classification is done i.e., any class whose probability value is maximum is the classified result.

Optimizers

Optimizers help us in handling computational complexity problems by reducing them. Although there can be some challenges like:

1. Choosing the right learning rate for your model.
2. Too small learning rate can lead to torturing slow convergence and too large can hinder convergence.

3. Avoid getting trapped in their numerous sub-optimal local minima.

So, choosing the right optimizer for your neural network is imperative as it can help us to minimalize or maximize the error function denoted by E(x). E(x) is a statistical function which relies upon leaning constraints of the model. These parameters are used to calculate the targeted objective values (Y) from a set of predictors (X) used in a model. Weight and bias are the first learning parameters of the neural network, which are utilized in the computation of the output, and are used to move the problem to the correct solution by minimizing losses through the network training process. It also plays a major role in model training. Some of the optimizers are listed below-

1. Batch gradient descent
2. Stochastic Gradient Descent (SGD)
3. Mini-batch gradient descent
4. Momentum Based optimizers (Adaptive Learning Rate)

Batch gradient descent

It is otherwise known as Vanilla (basic) gradient descent which calculates the gradient descent for whole training dataset. It computes the 'gradient' of the Loss/error function with respect to the weights plus bias parameters θ for the trainable data for a network where ∇_θJ(θ) is the objective function used in the neural network.

$$\theta = \theta - \eta \nabla_\theta J(\theta)(4)$$

It ensures the convergence to the 'global' minima for convex error surface and to local minimum for non-convex surfaces. But due to its redundant computation's performance on larger datasets, it cannot be used in real time 'Neural Network' applications. The following diagrammatic representation shows how the convergence in BGD occurs.

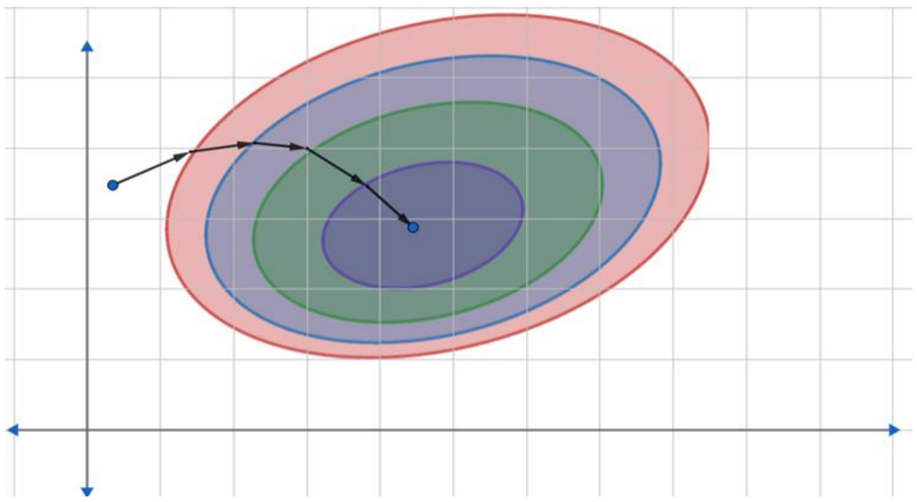


Figure 15 Convergence in BGD

Stochastic Gradient Descent (SGD)

The redundancy problem that we were having in ‘Batch Gradient Descent’ is solved by Stochastic Gradient Descent by carrying out one revision of weights at a time.
This helps to perform much faster and can be employed for online unsupervised . SGD similar convergence behavior as shown by BGD when learning rate is slowly decreased. The following figure represents the how there are rapid oscillations till convergence.

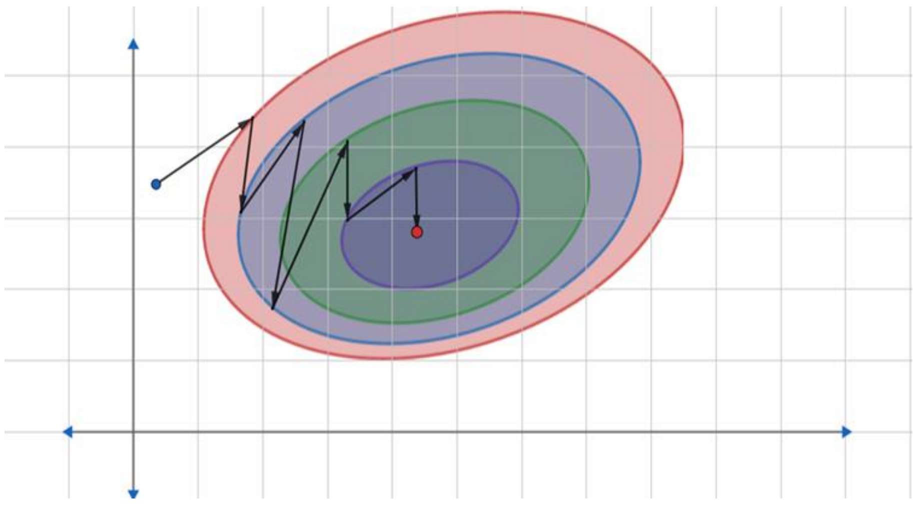


Figure 16 Convergence in SGD

Mini-batch gradient descent

Mini- batch gradient descent performs the best when compare with BGD and SGD as it executes a weight revision for each mini-batch of ‘k’ training examples. It decreases the disparity of the parameter updates, which leads to high stability convergence. Some of the reasonable minibatch sizes are usually of 32, 64, 128, 256, 512, 1024.

Momentum Based optimizers (Adaptive Learning Rate)

Momentum helps to accelerate SGD in the relevant direction and dampens oscillations. Summation of a fraction of the ‘update vector’ from the past time step to the current update vector makes it possible. One of the popular momentum-based optimizers is ‘Adam’ which is an abbreviation for Adaptive Moment Estimation. It is an algorithm that calculates variable rates of learning for every data sample and feature. Adam has better performance in optimizing neural network hyper-parameters because it the rate of convergence is high. Moreover, the learning speed is very high when compared to other optimizers, i.e., it overcomes most of the problems encountered in the previous discussed optimizers. Thus, Adam was most suited for hand movement classification, as it exhibited the highest accuracy in minimum number of epochs. It also had the lowest loss when compared to other optimizers.

Learning Rate

Learning rate is essentially the most important hyper-parameters that helps in adjustment of weights with respect to the loss gradient of neural network model. This what we call as tuning the models. In Deep learning, SGD optimizer is mostly used. Learning rate lets the optimizers know that how far to shift the weights in opposite direction of mini-batch gradient.

Low learning rate leads to lot of time for optimization because of small steps while large learning rate may cause the training not to converge to minima. The best way to setup learning rate is to start with high steps and then to decrease it during training the model to allow more epoch weight updates.

Although there are numerous ways to setup learning rate at the beginning. The absurd way is to try a few different values and see the which one offers us the best loss without much degradation in speed. A good approach could be to start with 0.1 and then lower the value of learning rate exponentially to 0.01 and 0.001 and even more less.

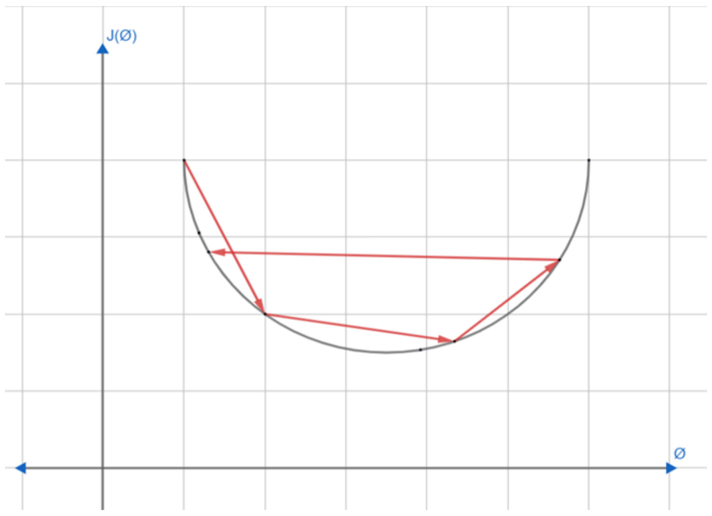


Figure 17 Large Learning Rate

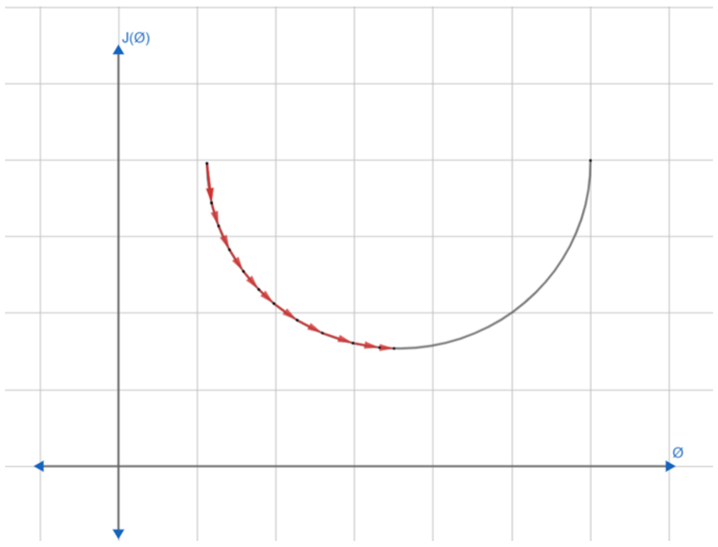


Figure 18 Small Learning Rate

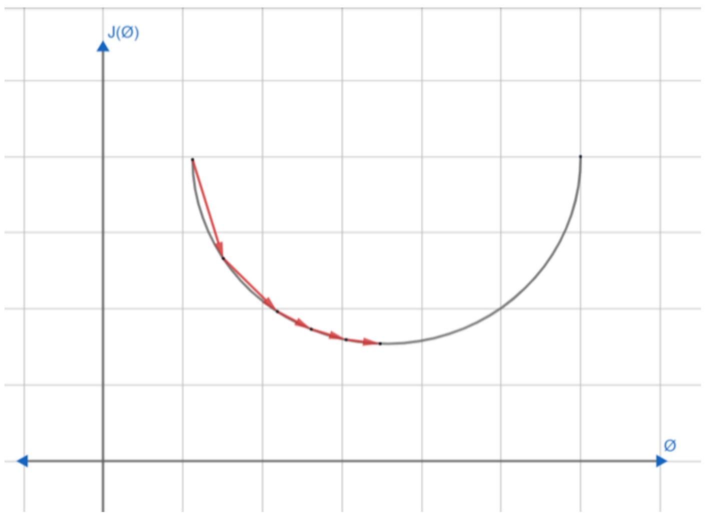


Figure 19 Adaptive Learning Rate

The above figures show the convergence with large, small, and adaptively optimized learning rate.

Back propagation and Loss Functions

Backward propagation is the process through which training neural networks are trained. In this method gradient of loss function is calculated with respect to all the 'weights' of the model. During the training period of the model, using arbitrary weights and biases the model computes a 'predicted' outcome, which may not be certainly equal to the actual output. If they are inequitable, a Cost Function is sought using the found and the desired output and this is then sent in the backward pathway in the model that alters the weights which would favour the prediction to resemble the actual output. One popular loss function is Mean Square Function given by:

$$L(\emptyset) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\left(\hat{y_{i,J}} - y_{i,j}\right) \ (5)$$

In the above equation $\emptyset$ represents weights or biases under consideration and L(Ø) is the loss function defined by the 'Mean-Square-Difference' of actual output $\hat{y_{i,j}}$ and the predicted output $y_{i,j}$ for all the inputs N and outputs K.

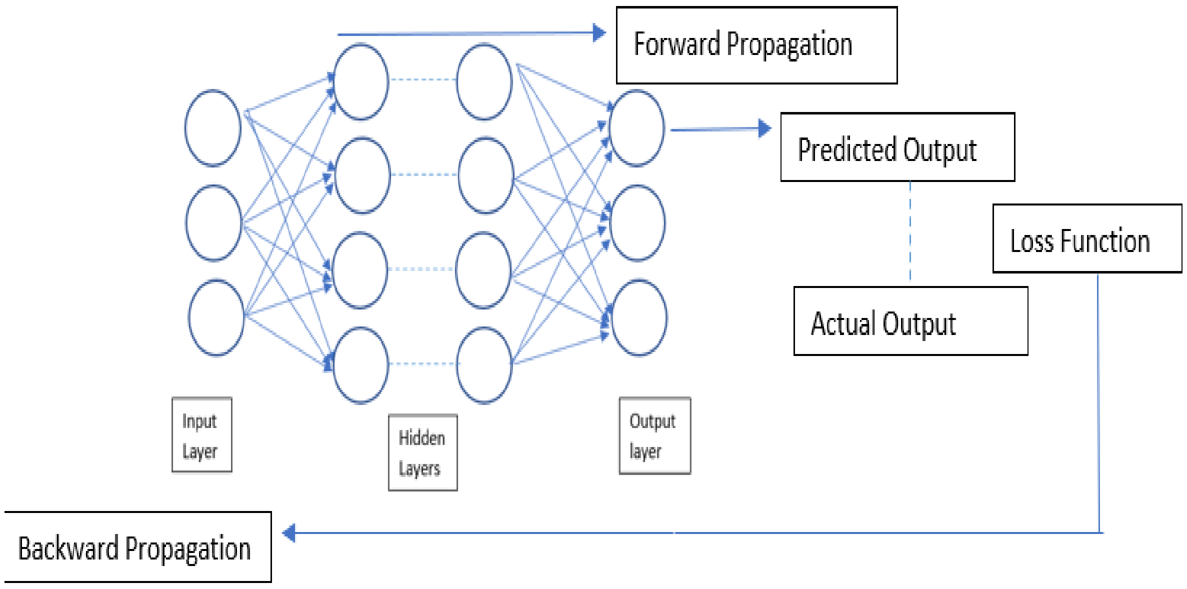


Figure 20 Backpropagation in Neural networks

During Back Propagation, the parameters are tweaked via the following formula: -

$$\emptyset_{New} = \emptyset_{old} - \mu \frac{\partial L(\emptyset)}{\partial \emptyset} (6)$$

Where, $\emptyset_{old}$ is any initial weight, $\emptyset_{New}$ is the corresponding updated weight, μ is the Learning Rate and $\frac{\partial L(\emptyset)}{\partial \emptyset}$ is the partial differential of the loss function following chain rule till the weight under consideration. It can be noticed that when Sigmoid or Hyperbolic Tangent functions are used for activation their derivatives are in ranges 0 to 0.25 for sigmoid and 0 to 1 for Tanh function. When chain rule is applied and multiple of values less than 1 are multiplied, the resultant value becomes diminishingly small, this is essentially what the Vanishing Gradient Problem is. But, with ReLU activation the derivative could either be 0 or 1 and not any intermediate value. Considering that for every case the derivative is 1 then new weights will always be different from the initial weights and convergence me be achieved. It is also imperative to notice that if at least one derivative in the 'chain-rule' partial derivations is zero then, the newly altered weight will be exactly same as the initial weight. In this case the neuron is unable to learn anything and is called a 'dead' neuron. To resolve this issue the concept of 'Leaky' ReLU was engineered. Under this method when x<0 then the output is not exactly mapped to zero but to a small-valued constant multiple of x.

$$g(x) = \begin{cases} x, when x \geq 0 \\ 0.01x, when x < 0 \end{cases} (7)$$

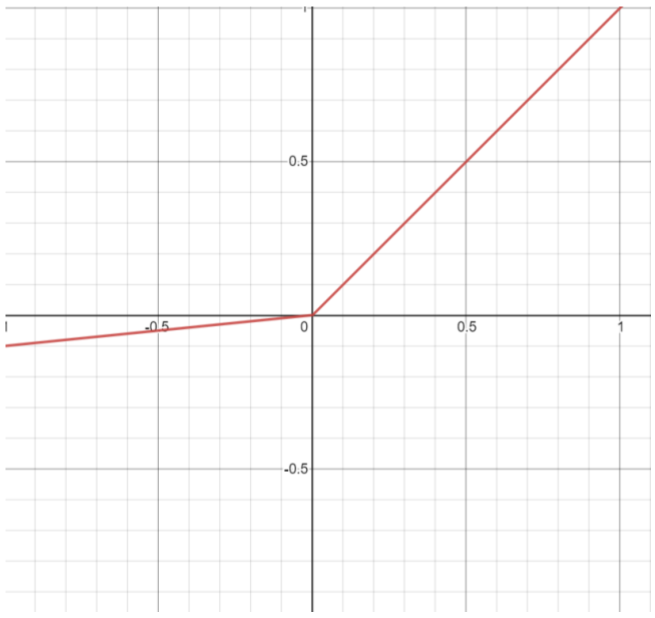


Figure 21 Leaky ReLU concept

Hence, when the derivative is found when x<0, it does not result in a dead neuron.

Shallow Network

Usually, neural network consists of many hidden layers for deeper extraction of features from the input and to minimize the loss function through back propagation but there is also a type of Neural-Net with less number of hidden layers. Less deep/Shallow neural nets comprise of 1 to 2 hidden layers only. Due to this few number of hidden layers, the extraction of information process become rapid. Although it does not guarantee to give very good results in all the situations due to its only few layers for features extraction. Fig.22 Shows a shallow neural network with one input layer along with a singular hidden layer and one output layer.

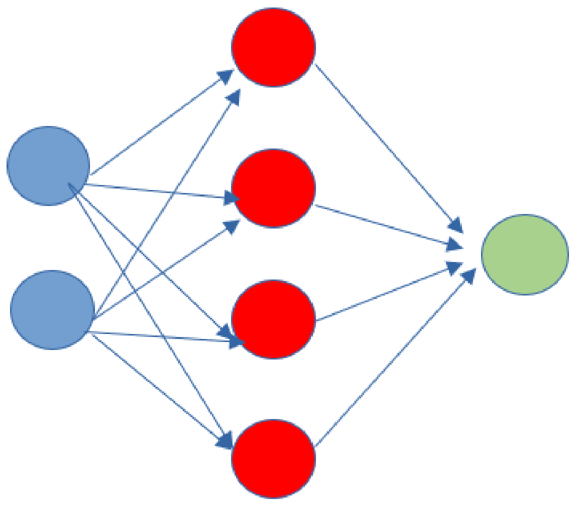


Figure 22 Shallow Neural Network with 1 Hidden layer

Fig. 23 shows the atomic unit of a neural network called as neuron. Given an input, it offers the result and passes that output as an input to the following/next layer. A neuron may be thought of as an aggregate of two elements. The primary element calculates the output ‘z’ which includes use of inputs and weights and bias, while the secondary element performs the activation on output of primary part to present out the very last output “a” of the neuron.

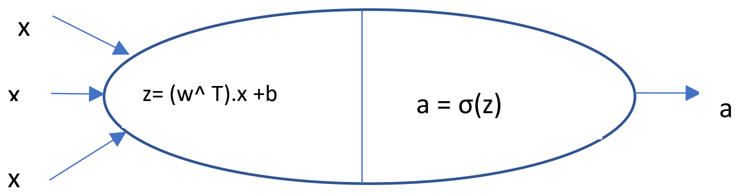


Figure 23 A Neuron

When a shallow network of architecture shown in **table-3** was run on dataset SIPaKMed the following results were found.

Table 3 Architecture-1 of shallow network

Layer (type)	Output Shape	Parameters
conv2d (Conv2D)	(None,126,126,8)	224
max_pooling2d (MaxPooling2D)	(None,63,63,8)	0
conv2d_1 (Conv2D)	(None,61,61,16)	1168
max_pooling2d_1 (MaxPooling2	(None,30,30,16)	0
conv2d_2 (Conv2D)	(None,28,28,32)	4640
max_pooling2d_2 (MaxPooling2	(None,14,14,32)	0
Flatten (Flatten)	(None,6272),0,	0
Dense (Dense)	(None,1021)	6404733
dense_1 (Dense)	(None,512)	523264
dense_2 (Dense)	(None,5)	2565

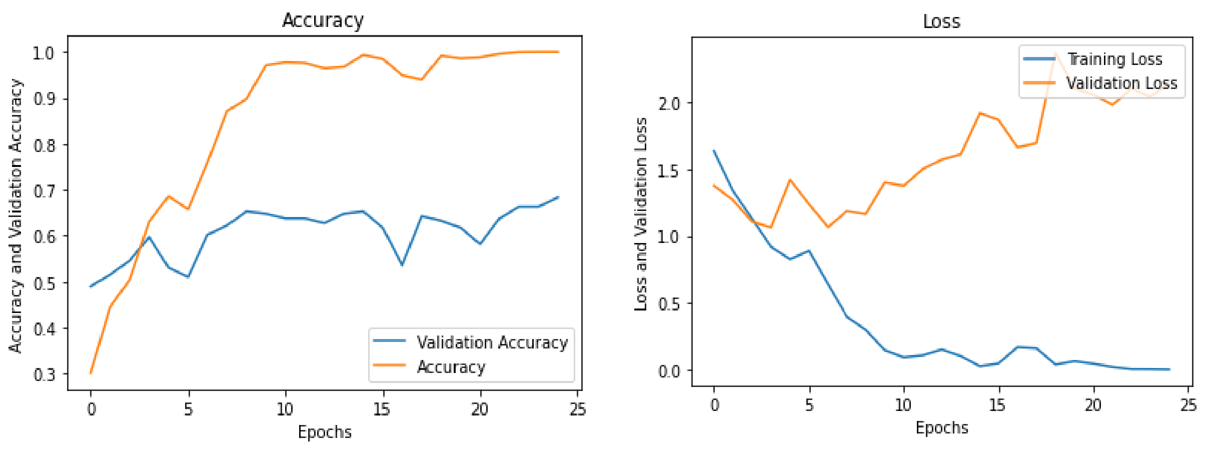


Figure 24 Training, Training loss, Validation loss and Accuracy of Shallow Architecture 1

The validation accuracy for the for the shallow network of Architecture- was found to be 68.37 as shown in Fig. 24. With the slight modification in architecture of shallow network shown in table 4, the model was executed again to get the better accuracy shown in fig. 26

Table-4: Architecture-1 of shallow network

Layer (type)	Output Shape	Parameters
conv2d (Conv2D)	(None,126,126,32)	896
max_pooling2d (MaxPooling2D)	(None,63,63,32)	0
conv2d_1 (Conv2D)	(None,61,61,64)	18496
max_pooling2d_1 (MaxPooling2	(None,30,30,64)	0
conv2d_2 (Conv2D)	(None,28,28,64)	36928
max_pooling2d_2 (MaxPooling2	(None,14,14,64)	0
Flatten (Flatten)	(None,12544)	0
Dense (Dense)	(None,512)	6423040
dense_1 (Dense)	(None,512)	262656
dense_2 (Dense)	(None,5)	2565

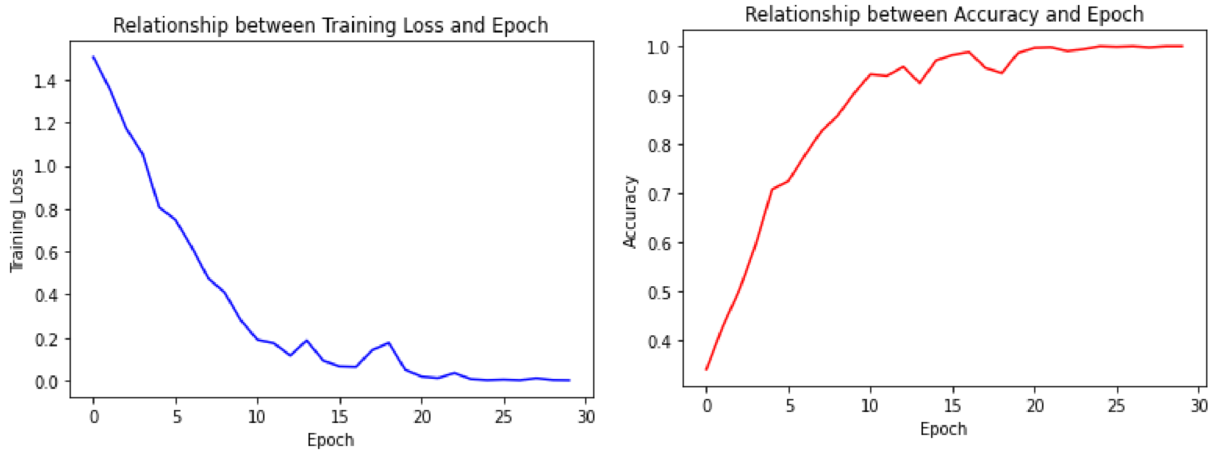


Figure 25 Training accuracy and Loss vs epochs

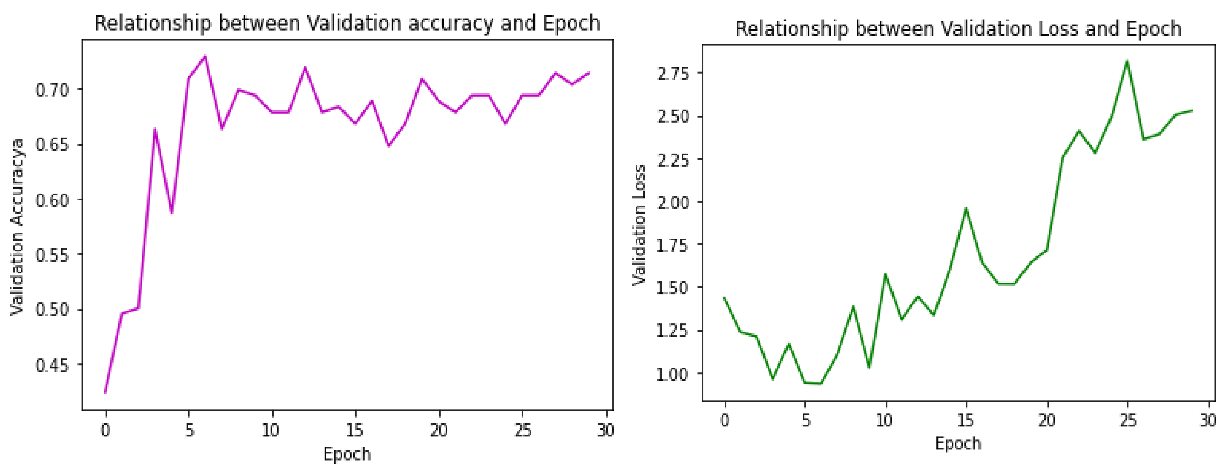


Figure 26 Validation accuracy and Loss vs epoch

If we sought the comparison of both shallow and deep neural network, they both can approximate any functions. Although for the same level of accuracy, deep neural networks may be a lot more prompt in terms of computation and quantity. They can extract deeper features at every layer and learn new and greater abstract representation of the input.

D. TRANSFER LEARNING

Machine-learning and deep-learning algorithms have been designed to work in isolation and are trained to solve certain types of tasks. The problem with the conventional deep learning models was that they need to be rebuilt from the scratch once the feature space distribution changes. Traditional learning is categorized and occurs only in terms of specific tasks, datasets and training models that differ from that. No knowledge is held which can be transferred from one to another. Transfer learning allows us to grasp information like features, weights etc. From previously trained model to train new models and also deal with problems such as having a small data for new tasks.

The deep transfer learning principles and concepts explain that a prebuilt model is not constrained to a particular task but, can be used to transfer the knowledge gained to previously unknown paradigms. This derives hefty inference from the 'human brain' itself and how the brain is capable of utilizing previous training, experiences and knowledge in situations it has not encountered before [14]. The pre-trained models are immensely beneficial when time and hardware requirements are considered. A shallow-net may not have as many trainable parameters, may not perform well with slightly scarce data and may be much more expensive in terms of hardware resources. These pre-trained models already have the knowledge gained from millions of images from large datasets; this knowledge could be inferred to handle multiple newer challenges.

E. FASTAI

For the implementation of the said models FASTAI has been used. This is a library primarily focused on deep learning tasks and claims to provide cutting edge technology with a seamless interface and allows architectures to be flexible and easily scalable without compromising the performance. Heavy use of data abstraction has been used to make this library easily understandable with concise commands [15]. A layered API architecture has been built which is immensely modifiable to suit the demands of the mission. This library is tremendously compatible with other libraries like PyTorch and TensorFlow. These capabilities are even more enhanced with Graphical Processing Units to build a highly optimized computer vision application.

F. MODELS USED FOR THE PURPOSE

Every model was tuned with the same configuration of hyper-parameters. The number of EPOCHS for fine tuning of weights of models was 50; BATCH_SIZE of 10; Input image resolution: 224×224. The concept of Cyclic Learning Rate was employed instead of static or decreasing learning rate {insert citation of Leslie Smith}. This method allows oscillations of the learning rate under a specific range. Training accuracy, validation loss, training loss and F-Beta scores as functions of number of epochs for all the models are as follows:

1. AlexNet

AlexNet is one of the most famous neural-net architectures used till date. It was proposed by Alex Krizhevsky et al., 2012 for the ImageNet 'Large-Scale Visual Recognition Challenge' (ILSVRC-2012) and is based on CNN. ILSVRC judge the algorithm for object detection and image classification. The dataset was trained on ImageNet consist of 1.2 million high resolution images having 1000 labeled classes. It consists of 650,000 neurons with 60 million parameters. Due to its huge parameter size, it took around five to six days for training on two GPUs of GTX 580 3GB. Because of all these factors, the architecture won ILSVRC competition and achieved a winning of top-5 'error-rate' which was 15.3%, contrasted to next best result achieved by second position was of 26.2%.

Exploring the architecture of AlexNet

The architecture of AlexNet comprises of 8 layers in total out of which 5 are convolutional layers and 3 are fully connected layers. Max-pooling layer is added just after convolutional layer to pull out maximum number of features from it for the 1st two convolutional layers, while the third, fourth and fifth CNN layers are connected to fully connected layers. All the outputs of the CNN and fully connected layers are connected to a non-linear activation function known as 'ReLU' (Rectified Linear unit) and the final output is connected to a SoftMax layer. The architecture of the AlexNet is shown at Fig 27.

Convolutional Layer 1: Convolutional layer with filter size of 11*11 and number of filters= 96 with stride= 4 was applied. This stride of 4 will help to decrease the dimensions of input image by the factor of 4.
Max pool 1: After the convolutional layer, max pooling layer of 3*3 filter and stride= 2 is used. This helps to extract the maximum number of features and also reduce the dimensions of the output given by convolutional layer 1.
Convolutional Layer 2: For second convolutional layer, filter size of 5*5 along with max pool-1 with stride= 0 is used.
Max pool-2: After the convolutional layer-2, we will apply max pooling with 3*3 filter. We will also use stride= 2 to reduce the dimensions.
Convolutional layer 3,4,5: For these three convolutional layers, we will use 3*3 filter along with the stride= 1 and padding=1.
Max pool 3: This will be the final 'max-pool' layer that is put in AlexNet. Stride=2 is used with the total dimensions of 6*6*256= 9216.
Fully Connected layer 6, 7, 8: Lastly, AlexNet consist of 3 fully-connected layers of which first two layers have 4096 nodes, and the ultimate fully-connected node has 1000 units. All the outputs of the CNN and sully connected layers are connected to a 'non-linear' activation function called as 'ReLU' (Rectified Linear unit) and the final output is connected to a SoftMax layer.

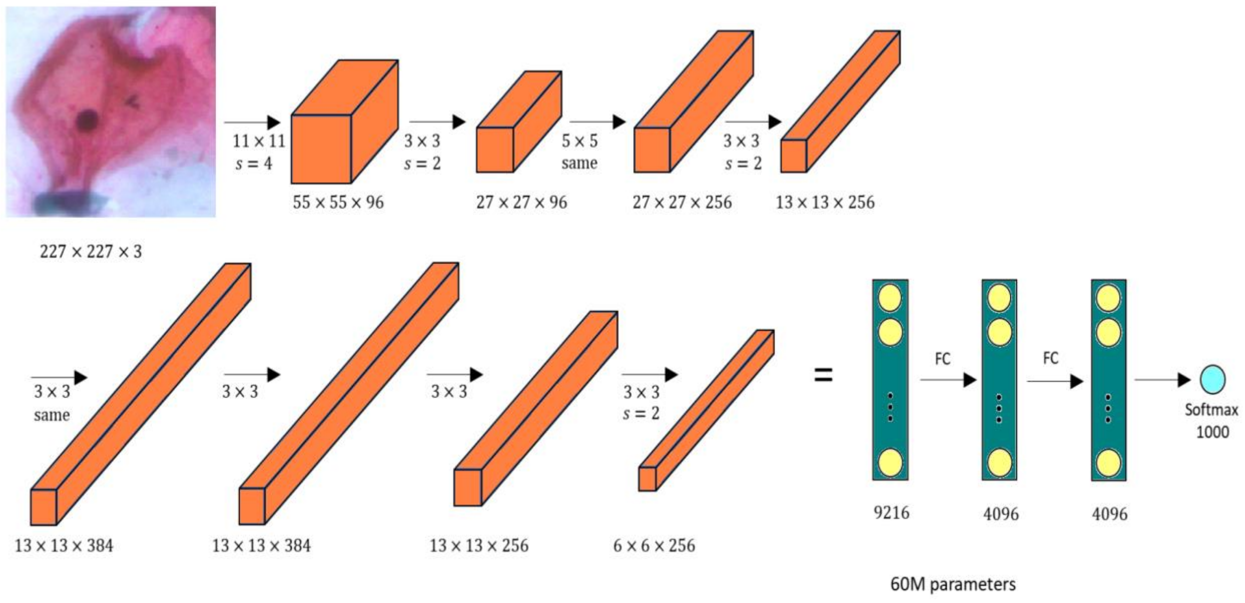


Figure 27 AlexNet Architecture

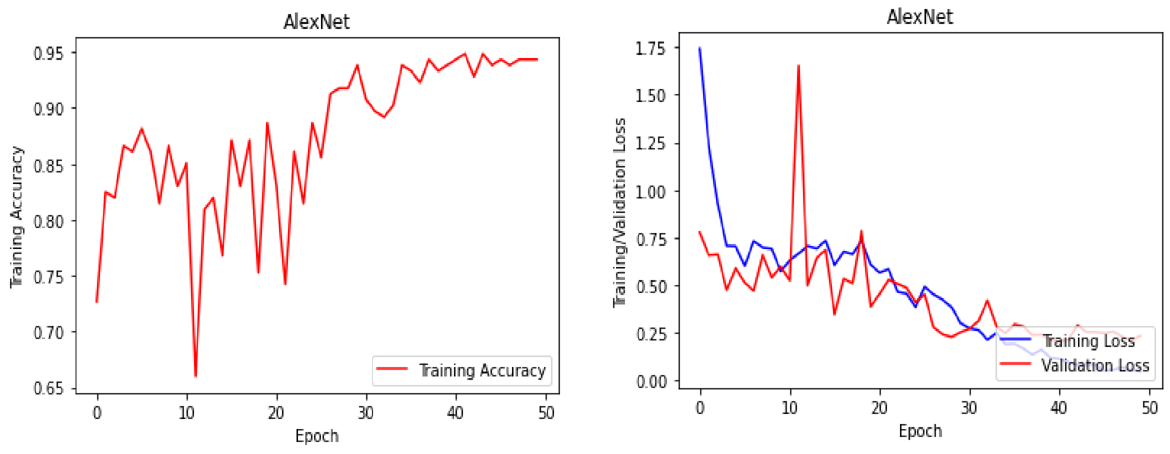


Figure 28 Training Accuracy, Training Loss, Validation Loss vs Epochs

2. VGG-16

Vgg16 is another convolution neural network architecture which was trained on ImageNet dataset. It was proposed by K. Simonyan et al., in their paper. ILSVRC judge the algorithm for object detection and image classification. The dataset was trained on ImageNet consist of 14 million high resolution images having 1000 labeled classes.

Highlight of this model was the replacement of larger convolution filters with multiple 3×3 filter receptive fields [18]. The purpose of doing so was to enable to model to learn deeper details and thereby more complex features could be extracted. This is a memory heavy architecture occupying about 500 megabytes and has over 130 million trainable parameters. It makes the step-up over AlexNet by swapping the kernel size filters to smaller one (3*3) for deeper extraction one after another. It took around weeks for training on NVIDIA Titan Black GPUs. Because of all these factors, the architecture won ILSVRC competition and achieved a winning of top-5 accuracy rate 92.7% in ImageNet.

The vgg16 architecture was trained for the 50 epochs and the best ‘training accuracy’ was obtained to be 95.87% as shown in Fig.30. The trends of training and validation losses for VGG-16 is shown in Fig.30

Exploring the architecture of VGG-16

First two layers of vgg-16 are convolutional layers with filter size of 3*3. Then we use pooling layer which lowers the dimensionality of a volume from 224*224*64 to 112*112*64. After that, we have few more couple of convolutional layers. We use 128 filters in all same convolutions, which means dimension will be 112*112*128. Then, a pooling layer is added which cuts the height*width*depth to 56*56*128. After that, 3 convolute-layers with 256 filter is used which gives the output of 14*14*512. Then the pooling layer is used along with few more convolutional layers with 512 filters and again a pooling-layer. Finally, at the end we have 7*7*512 fully connected layers (FC) with total of 4096 units. All the outputs of the CNN and fully connected layers are connected to a non-linear activation function called as ReLU and the final output is connected to the ultimate Soft-Max layer. The architecture of the Vgg-16 is shown at Fig. 29.

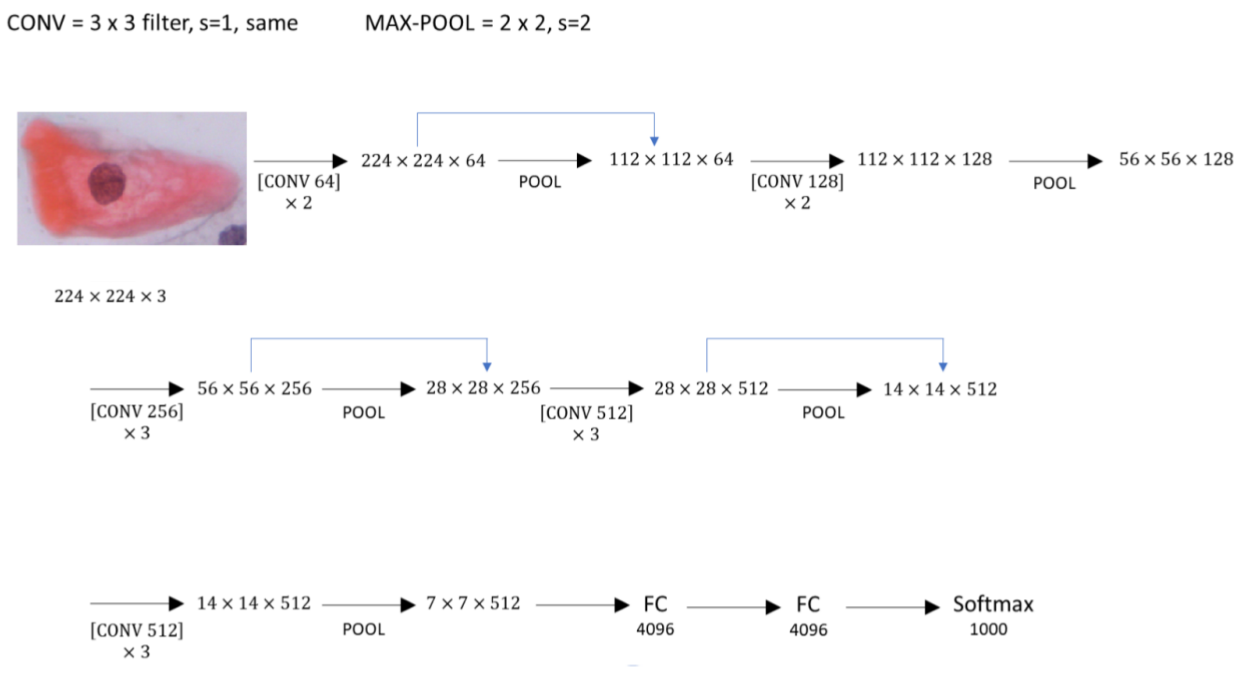


Figure 29 VGG16 Architecture

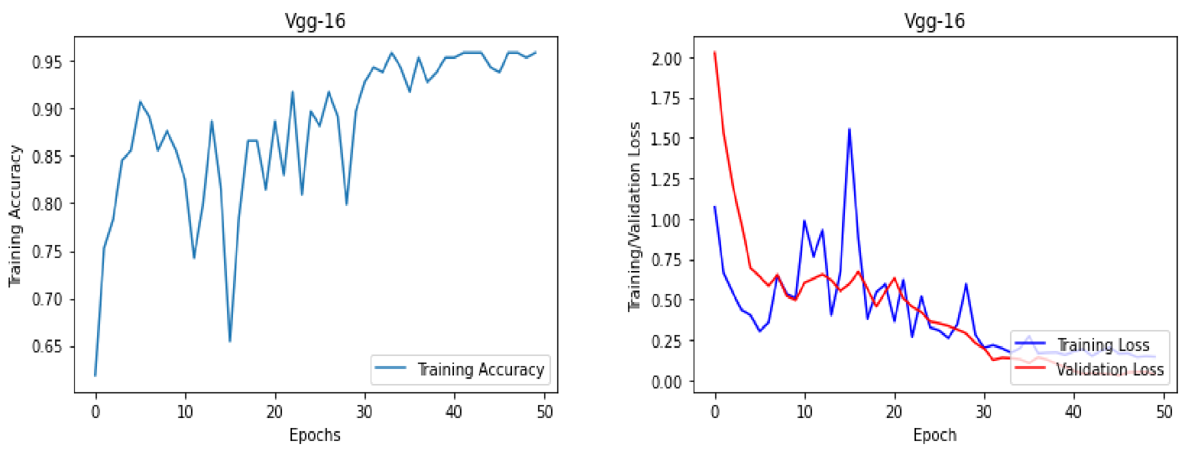


Figure 30 VGG16 Training accuracy, Training Loss, Validation Loss vs Epochs

3. VGG19

There is not anything fundamental difference between Vgg16 and Vgg19 except for the number of layers. While the developers were building the models the observed that there wasn't distinguishable improvement in accuracy when the number of layers were increased that was the reason this architecture was not developed any further [19]. Even though convergence was observed formerly but improvement in accuracy reduced. The model training was for the 50 epochs and the most admirable training accuracy was obtained to be 95.36% as shown in Fig.13. The trends of training and validation losses for ResNet-50 is shown in Fig.14.

Exploring the architecture of VGG-19

First two layers of vgg-19 are convolutional layers with filter size of 3*3 and stride=1 to enable them to cover whole part of image. Padding was also done to preserve the resolution of the image [20]. Then we use pooling layer along with stride=2 which helps to reduces the height and width of a volume. After that, we have few more couple of convolutional layers followed by max pooling layers. Then lastly, three fully-connected layers are used. All the outputs of the CNN and dense connection layers are connected to a non-linear activation function called as ReLU and the final output is connected to a Soft-Max layer. The architecture of the Vgg-19 is shown at Fig. 31

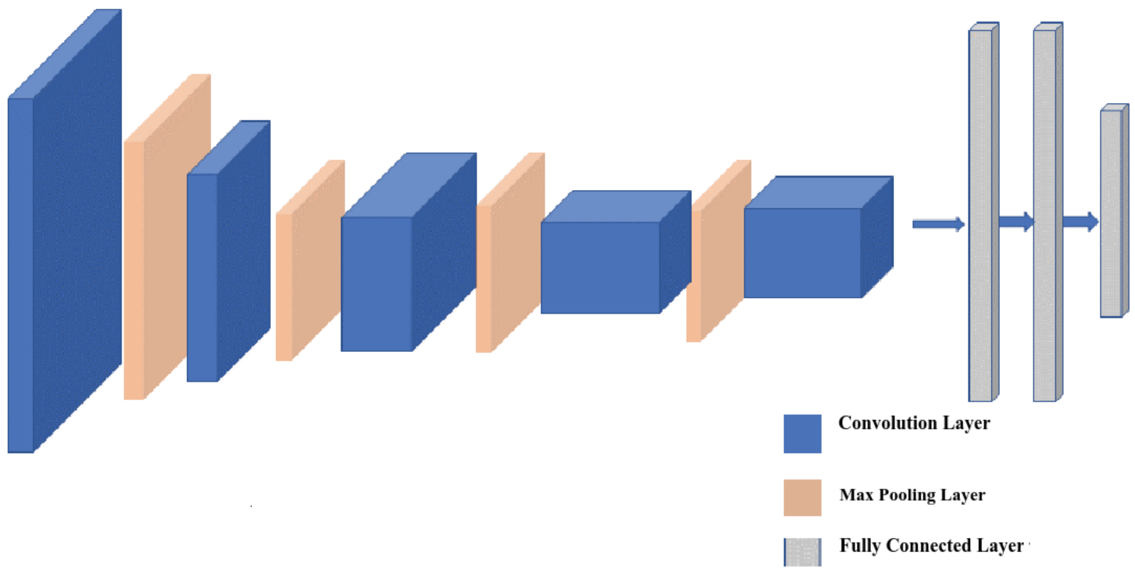
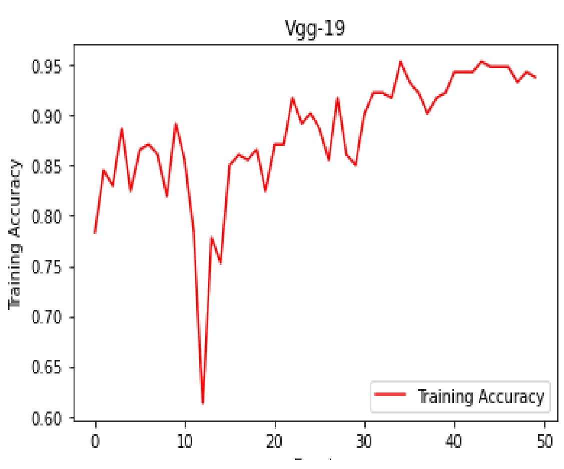


Figure 31 VGG19 Architecture



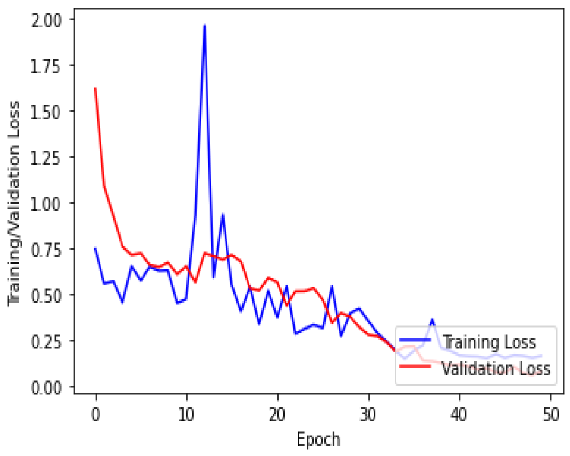


Figure 32 VGG19 Training Accuracy, Training Loss, Validation Loss vs Epochs

4. ResNet50

A fundamental question in deep learning was whether increasing the number of convolution layer will result in better accuracy. For some instances, increasing the number of layers did provide superior performance but it was not guaranteed for all use cases. There were certain obstacles that were risen due to addition of multiple convolution layers, which were, vanishing gradient problem and exploding gradient problem. Under the vanishing gradient problem, when certain activation functions were used on the layers the loss function gradient approached zero. This condition was unwanted because this indicated that the architecture is becoming harder to train. In Activation functions like sigmoid function when the input becomes smaller or larger, the derivative approaches zero. During back propagation derivative are found in reverse order, from the final/output layer to the first layer. Following the chain rule all the derivatives are multiplied. When these derivatives are of small value, their product will be an even smaller value. Thus, the gradient decreases exponentially thereby implying the weights and biases of the layers are not properly updated. On the other hand, the exploding gradient problem occurs when the large error gradients are multiplied to make an exponentially large product which results in greater valued updates in the parameters and makes the model unstable. Also, it has been observed that although it was expected that adding more layers would improve accuracy but, as the layers of a model are deepened the accuracy saturates a certain point and then further degrades. One solution for these concerns was the use of ReLU activation. The outcome value from ReLU activation was always high valued which required heavy batch normalization, or it will otherwise lead to exploding gradient again. RESNETs also provide another solution by adopting the concept of residual blocks. When the convolution layers took input as X and gave output as F(X), this output may decrease when additional layers are added. Here feed forward connections were used. These were alternatively called 'Skip Connections'. Using these the input was also transferred and added in the output as F(X)+X. the purpose of doing so considered two scenarios: first, if the extra layers degraded the performance, they will be hopped thereby giving the input a shortcut to the succeeding set of layers, bypassing the degrading layers. Second, if the layers are favouring the performance improvement, their effect will be more enhanced. RESNETs also utilized bottle neck connections, which are 1×1 convolutions which decreased the number of trainable parameters.

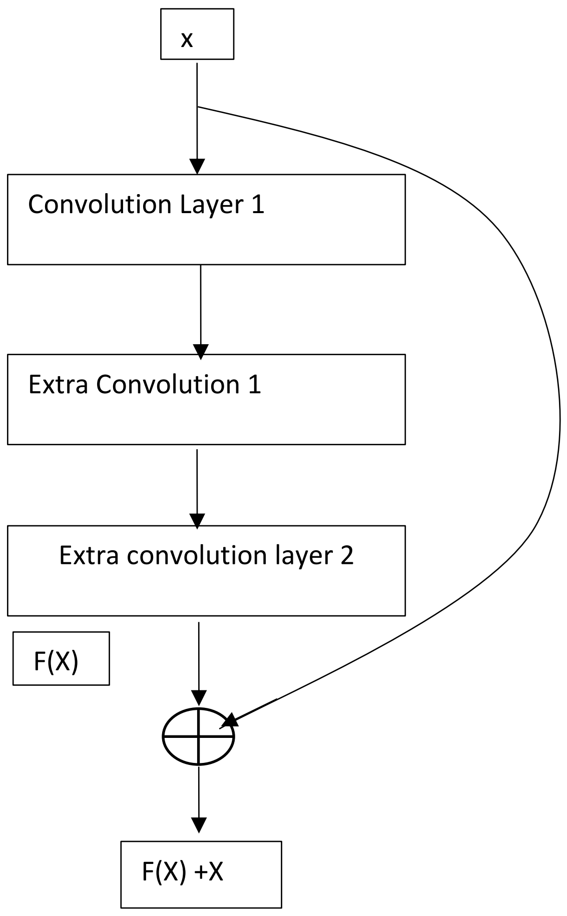


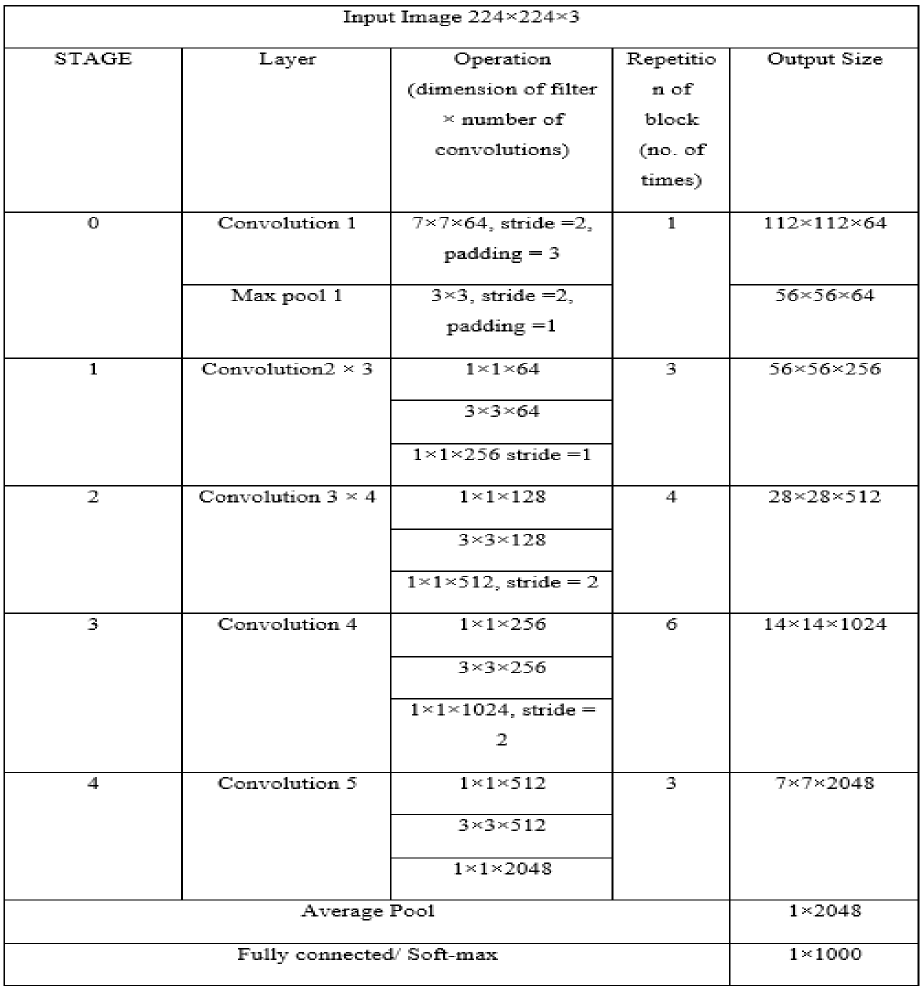
Figure 33 Residual Networks

RESNET 50 Architecture

ResNet50 is a moderately deep architecture with 50 convolution layers, an average pool layer after these convolution layers and finally a fully connected layer with 1000 output neurons. The activation function which is applied on the output layer is Soft-max. As the output layer has 1000 neurons as ImageNet had 1000 classes, whenever a multiclass classification with n number of classes needs to be performed, after the training of architecture on training subset, another layer is to be added with n neurons dense connected with the last layer. Now the activation of the penultimate layer needs to be altered from Soft-max to ReLU. The function on the new last layer must be put to Soft-max.

ResNet50 architecture is concisely explained in the following table: -

Table 4 ResNet50 Architecture



Input Image 224×224×3				
STAGE	Layer	Operation (dimension of filter × number of convolutions)	Repetition of block (no. of times)	Output Size
0	Convolution 1	7×7×64, stride =2, padding = 3	1	112×112×64
	Max pool 1	3×3, stride =2, padding =1		56×56×64
1	Convolution2 × 3	1×1×64	3	56×56×256
		3×3×64		
		1×1×256 stride =1		
2	Convolution 3 × 4	1×1×128	4	28×28×512
		3×3×128		
		1×1×512, stride = 2		
3	Convolution 4	1×1×256	6	14×14×1024
		3×3×256		
		1×1×1024, stride = 2		
4	Convolution 5	1×1×512	3	7×7×2048
		3×3×512		
		1×1×2048		
Average Pool				1×2048
Fully connected/ Soft-max				1×1000

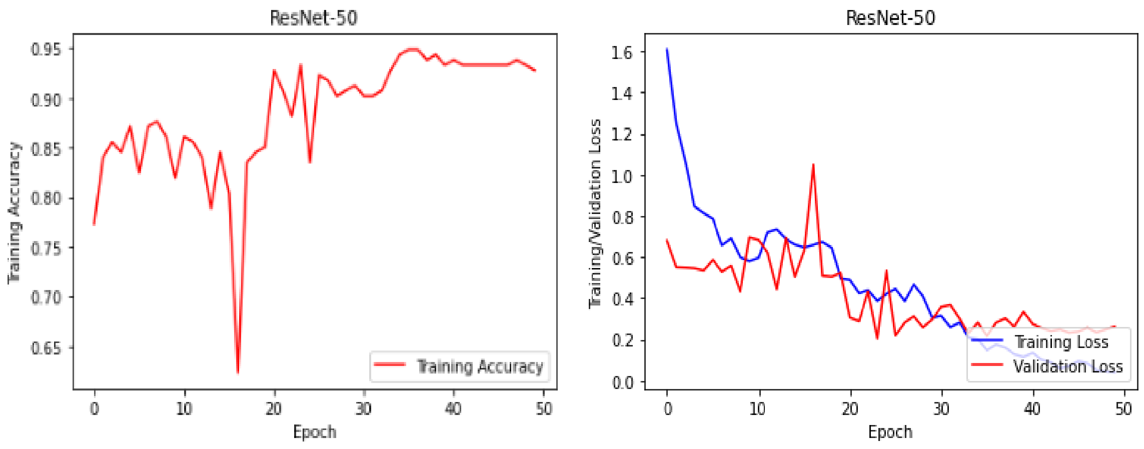
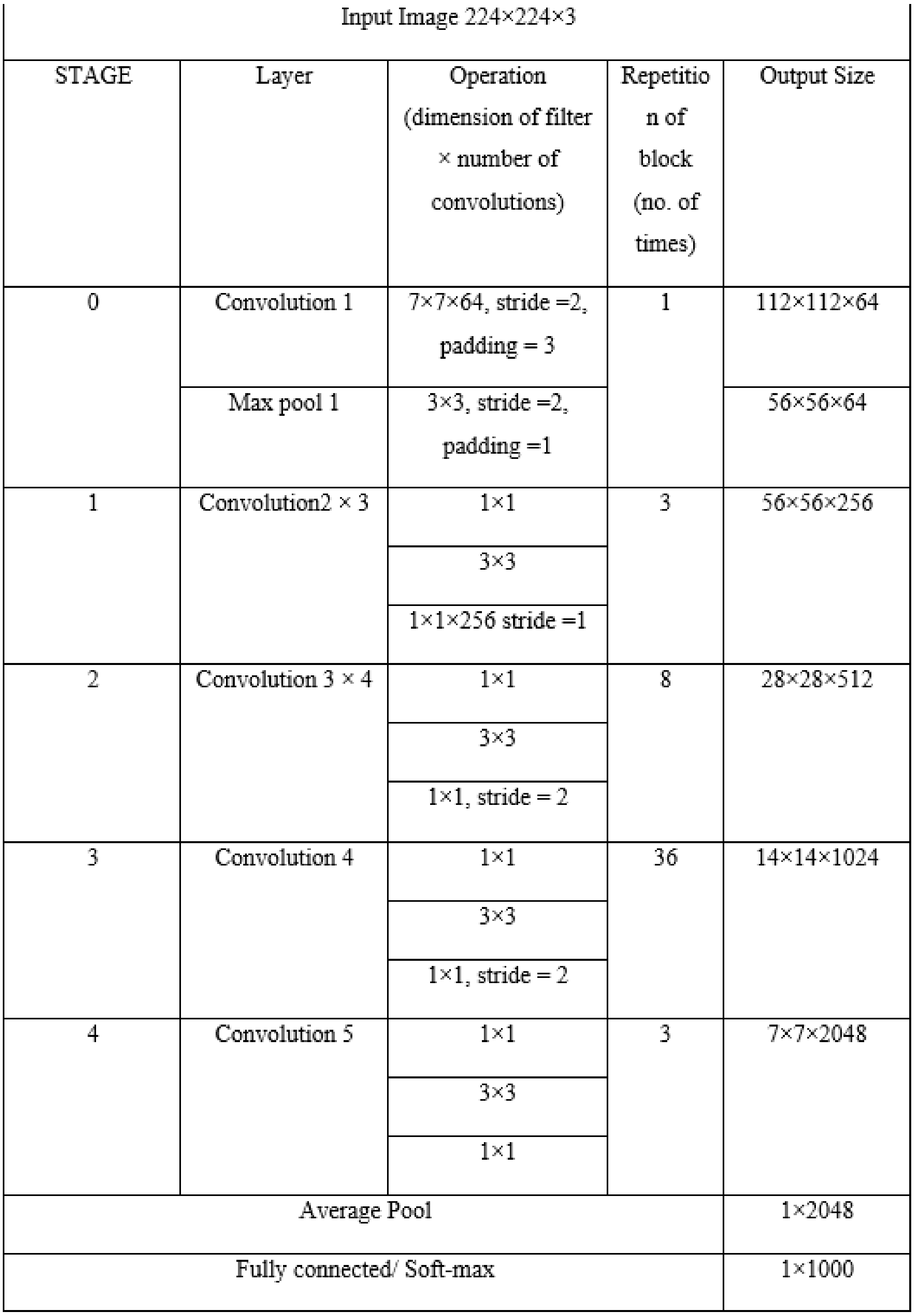


Figure 34 Training Accuracy, Training Loss, Validation Loss vs Epochs

5. ResNet152

ResNet152 is the deeper variant of ResNet50 with 152 total convolution layers. The fundamental concept remains same. Where ResNet50 has 23 million trainable parameters ResNet152 has 60.3 million parameters. It is also trained on one million images of the ImageNet dataset. It was evidently stated that even though it being such a deep network, it has complexity even lesser than that of VGG16 which was only a 16-layer deep architecture [21].

Table 5 ResNet152 Training Accuracy, Training Loss, Validation Loss vs Epochs



Input Image 224×224×3				
STAGE	Layer	Operation (dimension of filter × number of convolutions)	Repetition of block (no. of times)	Output Size
0	Convolution 1	7×7×64, stride =2, padding = 3	1	112×112×64
	Max pool 1	3×3, stride =2, padding =1		56×56×64
1	Convolution2 × 3	1×1	3	56×56×256
		3×3		
		1×1×256 stride =1		
2	Convolution 3 × 4	1×1	8	28×28×512
		3×3		
		1×1, stride = 2		
3	Convolution 4	1×1	36	14×14×1024
		3×3		
		1×1, stride = 2		
4	Convolution 5	1×1	3	7×7×2048
		3×3		
		1×1		
Average Pool				1×2048
Fully connected/ Soft-max				1×1000

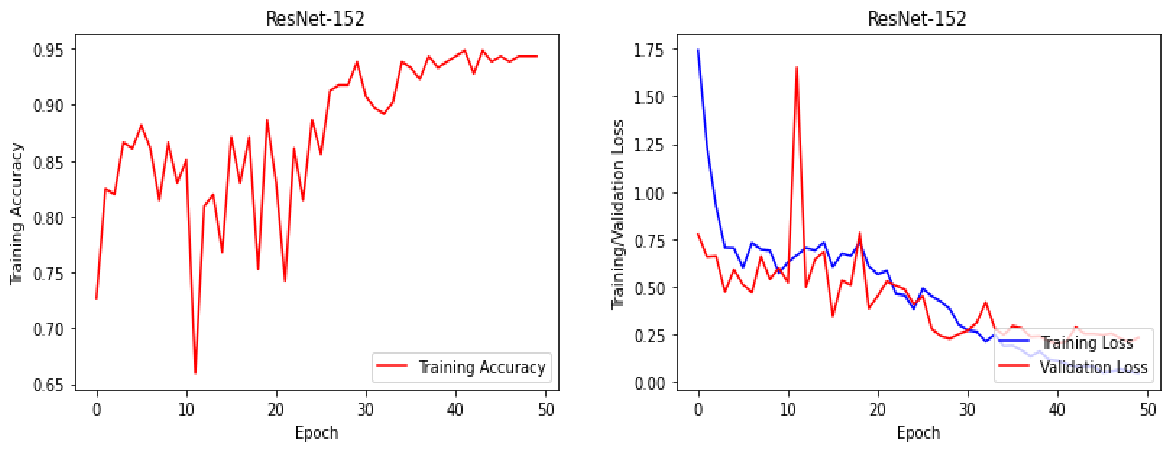


Figure 35 ResNet152 Training Accuracy, Training Loss, Validation Loss vs Epochs

6. DenseNet169

DenseNets or Densely Connected CNNs are a modification of the conventional CNN models. In traditional CNN models any layer takes the feature maps of the previous layers as input and outputs a stack of feature maps to the next layer as inputs. In DenseNets, any intermediate layer receives input from not just the previous layer but every preceding layer as well. So, if there are N number of layers, there will be N(N+1)/2 direct connections. These models concatenate the feature maps of preceding layers which is input to the next layer. But the concatenation can only occur when the dimensions of the all the previous feature maps is same but in Convolution nets the 'feature map' dimension change after every single convolution operation. To resolve this the model was broken down into multiple dense blocks, inside which the feature map remains same but it dimensions may vary from block to block. On the outside of these blocks, the pooling and convolution operations are done. These layers between dense blocks are called transition layers.

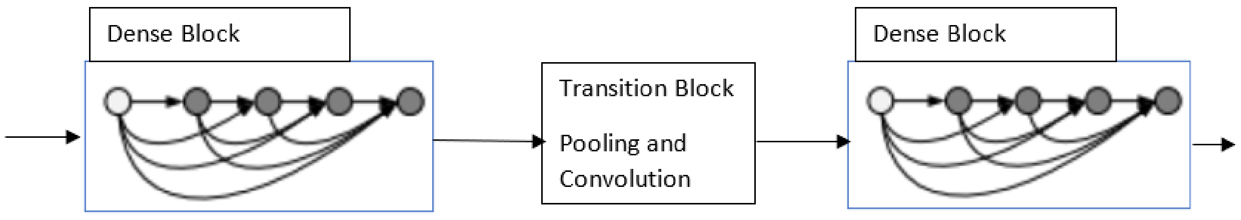


Figure 36 Dense blocks and Transition layers of DenseNet Architecture

Table 6 DenseNet169 Architecture

Input Image 224×224×3			
Layer	Operation	Repetition of block (no. of times)	Output Size
Convolution 1	7×7 convolution, stride =2	1	112×112
Max pool 1	3×3 convolution, stride =2		56×56
Dense Block 1	1×1 convolution	6	56×56
	3×3 convolution		
Transition layer 1	1×1 convolution		56×56
	2×2 average pooling, stride = 2		28×28
Dense Block 2	1×1 convolution	12	28 ×28
	3×3 convolution		
Transition layer 2	1×1 convolution		28×28
	2×2 average pooling, stride = 2		14×14
Dense Block 3	1×1 convolution	32	
	3×3 convolution		14×14
Transition layer 3	1×1 convolution		14×14
	2×2 average pooling, stride = 2		7×7
Dense Block 4	1×1 convolution	32	7×7
	3×3 convolution		
Classifier Layer	7×7 global average pooling		1×1

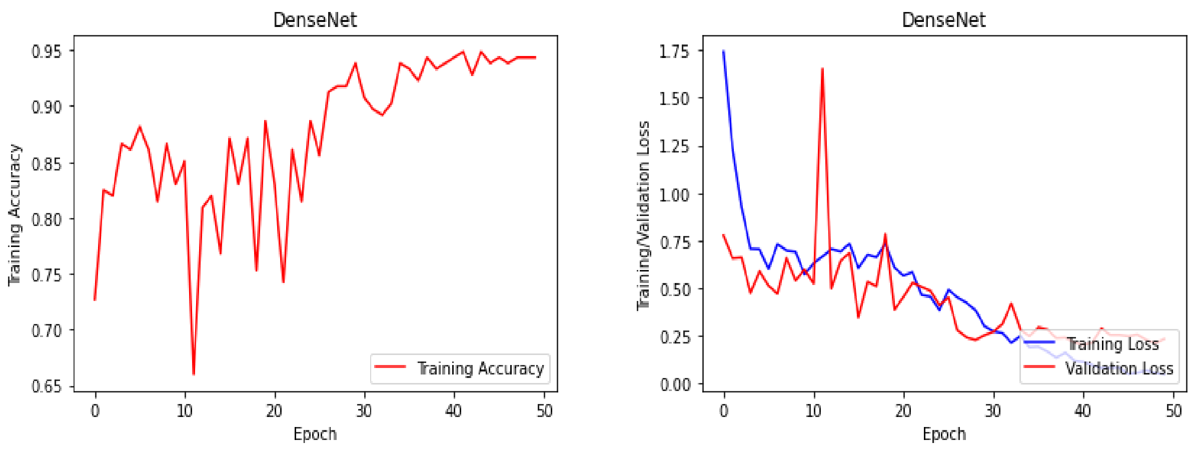


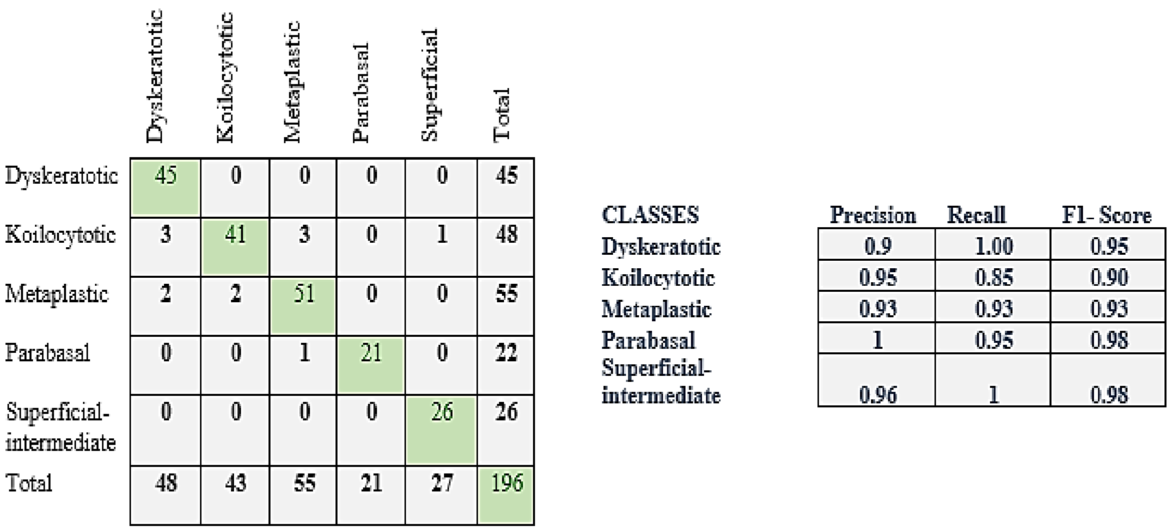
Figure 37 DenseNet169 Training Accuracy, Training Loss, Validation Loss vs Epochs

CHAPTER-4: RESULTS

To obtain the best accuracies the learning rate was cyclically varied which in itself is a hefty task, therefore a graphic card of 6 GB dedicated memory with 1920 CUDA enabled cores and 240 tensor cores was used. Testing of every one of the transfer learning models was done with 20% of the dataset, i.e., 196 images. This is a multi-class classification therefore precision, recall and F1-score will be calculated for every class.

1. RESNET-50

Confusion matrices which explain the correctness of classifications are built for all the models. Fig. 15 illustrates the 'confusion matrix' for ResidualNet50. This model has worked very well for Dyskeratotic and Superficial Intermediated classes with no misclassifications for either of them. Whereas the most terrible case for this model was seen in Koilocytotic cells with a net of 5 images misclassified. Testing accuracy of 93.87% was obtained for this model.



	Dyskeratotic	Koilocytotic	Metaplastic	Parabasal	Superficial	Total
Dyskeratotic	45	0	0	0	0	45
Koilocytotic	3	41	3	0	1	48
Metaplastic	2	2	51	0	0	55
Parabasal	0	0	1	21	0	22
Superficial-intermediate	0	0	0	0	26	26
Total	48	43	55	21	27	196

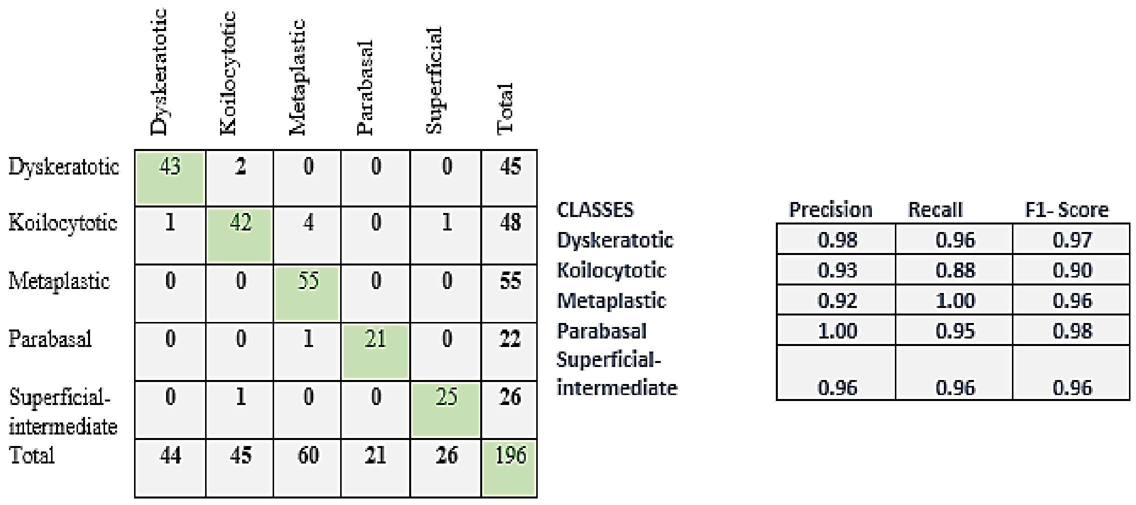
CLASSES	Precision	Recall	F1- Score
Dyskeratotic	0.9	1.00	0.95
Koilocytotic	0.95	0.85	0.90
Metaplastic	0.93	0.93	0.93
Parabasal	1	0.95	0.98
Superficial-intermediate	0.96	1	0.98

Figure 38 Confusion Matrix and performance matrices for ResNet50

Performance matrices for all 5 classes are given in Fig. 16. A 100% recall was obtained in Dyskeratotic and Superficial-intermediate classes which was in accordance with the classification performance seen in the confusion matrix.

2. RESNET-152

The model which performed best out of all four architectures was ResNet-152. The confusion matrix for the model is depicted in Fig. 17. This model worked efficiently for Metaplastic class with all 55 images correctly classified. Considerable deviations were still seen in case of koilocytotic cells due to their visual similarity with metaplastic cells. An accuracy of 94.89% was obtained for this model (Fig. 39).



	Dyskeratotic	Koilocytotic	Metaplastic	Parabasal	Superficial	Total
Dyskeratotic	43	2	0	0	0	45
Koilocytotic	1	42	4	0	1	48
Metaplastic	0	0	55	0	0	55
Parabasal	0	0	1	21	0	22
Superficial-intermediate	0	1	0	0	25	26
Total	44	45	60	21	26	196

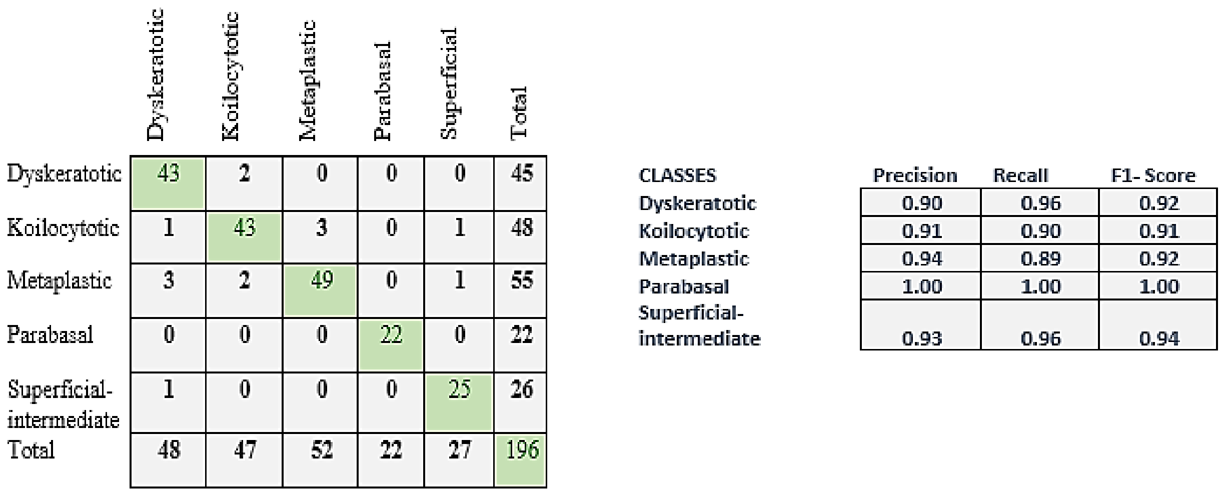
CLASSES	Precision	Recall	F1- Score
Dyskeratotic	0.98	0.96	0.97
Koilocytotic	0.93	0.88	0.90
Metaplastic	0.92	1.00	0.96
Parabasal	1.00	0.95	0.98
Superficial-intermediate	0.96	0.96	0.96

Figure 38 Confusion Matrix and performance matrices for ResNet152

Although a perfect recall was only obtained for metaplastic class, the model also worked well with Dyskeratotic cells. While ‘recall’ was observed for koilocytes was the worst (Fig. 39).

3. VGG-16

Vgg16 performed best with Parabasal cells and an accuracy of 92.85% was obtained which was the smallest amongst all four models (Fig.19).



	Dyskeratotic	Koilocytotic	Metaplastic	Parabasal	Superficial	Total
Dyskeratotic	43	2	0	0	0	45
Koilocytotic	1	43	3	0	1	48
Metaplastic	3	2	49	0	1	55
Parabasal	0	0	0	22	0	22
Superficial-intermediate	1	0	0	0	25	26
Total	48	47	52	22	27	196

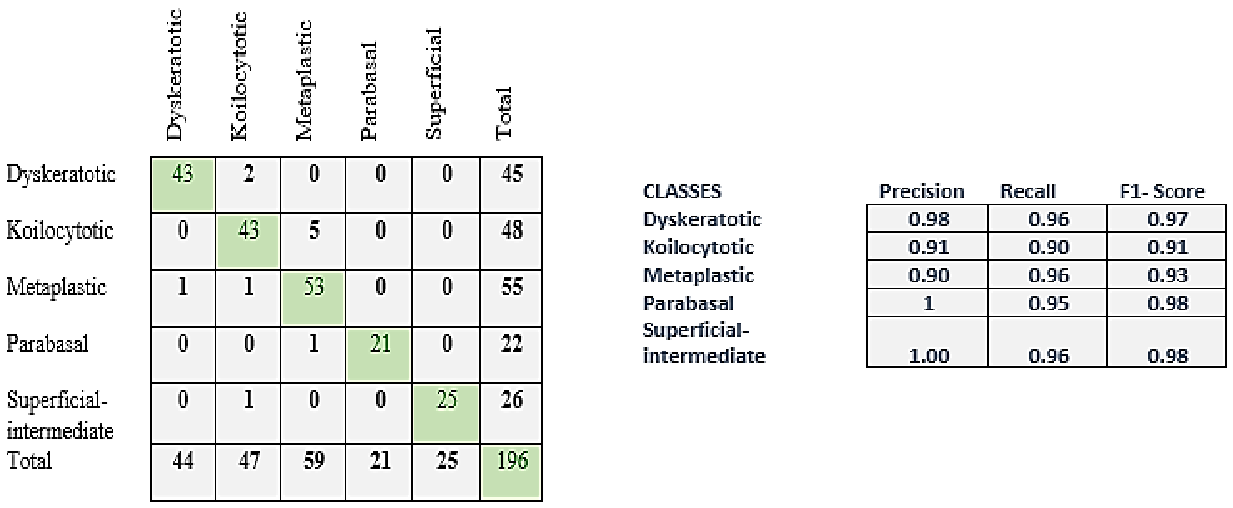
CLASSES	Precision	Recall	F1- Score
Dyskeratotic	0.90	0.96	0.92
Koilocytotic	0.91	0.90	0.91
Metaplastic	0.94	0.89	0.92
Parabasal	1.00	1.00	1.00
Superficial-intermediate	0.93	0.96	0.94

Figure 39 Confusion Matrix and performance matrices for Vgg16

Evidently only parabasal class had a perfect recall while metaplastic suffered the most in this matric.

4. VGG-19

Similar to Vgg16, Vgg19 also performed well with parabasal cells. Additional to this Superficial-intermediate also had only 1 misclassification, thus justifying a slightly higher testing accuracy than Vgg16, which was of 94.38% (Fig.21).



	Dyskeratotic	Koilocytotic	Metaplastic	Parabasal	Superficial	Total
Dyskeratotic	43	2	0	0	0	45
Koilocytotic	0	43	5	0	0	48
Metaplastic	1	1	53	0	0	55
Parabasal	0	0	1	21	0	22
Superficial-intermediate	0	1	0	0	25	26
Total	44	47	59	21	25	196

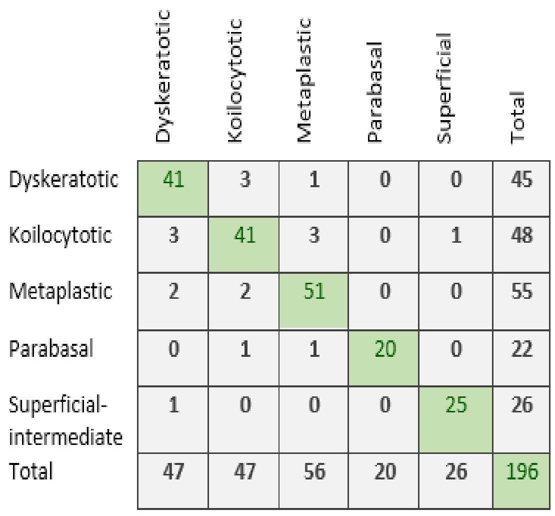
CLASSES	Precision	Recall	F1- Score
Dyskeratotic	0.98	0.96	0.97
Koilocytotic	0.91	0.90	0.91
Metaplastic	0.90	0.96	0.93
Parabasal	1	0.95	0.98
Superficial-intermediate	1.00	0.96	0.98

Figure 40 Confusion Matrix and performance matrices for Vgg19

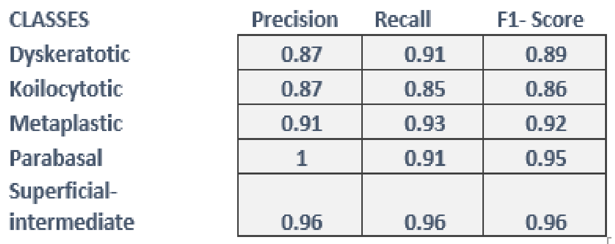
A perfect recall wasn’t seen in either of the classes, although precision of 100% was observed in parabasal and Intermediated classes (Fig.22).

5. AlexNet

Fig.42 presents the ‘confusion matrix’ for AlexNet. This model has worked very well for Parabasal class with no misclassification. While in case of Dyskeratotic, Koilocytotic and Metaplastic cell images, 5 images in total and in class of Superficial images, 1 miss classification was found. Testing accuracy of 90.81% was obtained for this model.



	Dyskeratotic	Koilocytotic	Metaplastic	Parabasal	Superficial	Total
Dyskeratotic	41	3	1	0	0	45
Koilocytotic	3	41	3	0	1	48
Metaplastic	2	2	51	0	0	55
Parabasal	0	1	1	20	0	22
Superficial-intermediate	1	0	0	0	25	26
Total	47	47	56	20	26	196

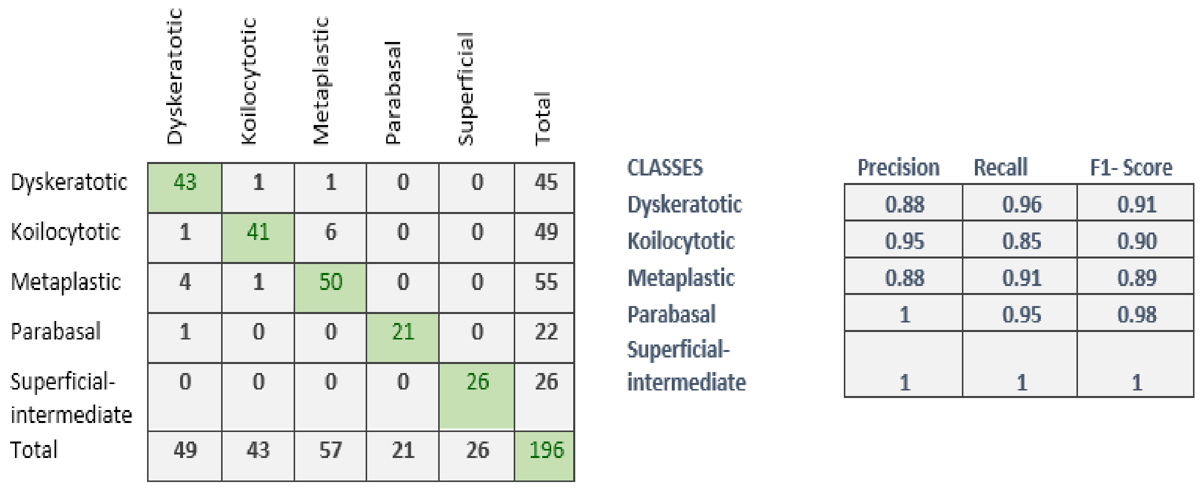


CLASSES	Precision	Recall	F1- Score
Dyskeratotic	0.87	0.91	0.89
Koilocytotic	0.87	0.85	0.86
Metaplastic	0.91	0.93	0.92
Parabasal	1	0.91	0.95
Superficial-intermediate	0.96	0.96	0.96

Figure 41 Confusion Matrix and performance matrices for AlexNet

6. DenseNet-169

Fig.43 presents the confusion matrix for DenseNet-169.This model has worked very well for Parabasal and Superficial class with no misclassification. Testing accuracy of 92.34% was obtained for this model.



	Dyskeratotic	Koilocytotic	Metaplastic	Parabasal	Superficial	Total
Dyskeratotic	43	1	1	0	0	45
Koilocytotic	1	41	6	0	0	49
Metaplastic	4	1	50	0	0	55
Parabasal	1	0	0	21	0	22
Superficial-intermediate	0	0	0	0	26	26
Total	49	43	57	21	26	196

CLASSES	Precision	Recall	F1- Score
Dyskeratotic	0.88	0.96	0.91
Koilocytotic	0.95	0.85	0.90
Metaplastic	0.88	0.91	0.89
Parabasal	1	0.95	0.98
Superficial-intermediate	1	1	1

Figure 42 Confusion Matrix and performance matrices doe DenseNet169

CONCLUSION AND FUTURE SCOPE

Cervical Cancer is among the universally diagnosed diseases affecting females. The pap-test (Papanicolaou-test) has proven to be the cheapest and least time-consuming method of diagnosis. Even though it requires a simple process, still after extensive studies it was proven to be very comprehensive and has enough useful information Multiclass classifiers been implemented using deep transfer learning. The SIPAKMED dataset which is newer as compared to many studies which were previously published. The classification was done between five classes namely, Dyskeratotic, Koilocytotic, Metaplastic, Parabasal and Superficial-intermediate[1]. Out of the classifiers, it was observed that ResNet-152 had the best performance (Fig.23). All these findings are restricted to the visual properties of a single dataset. This project can be extrapolated to be more robust by incorporating multiple datasets of similar pap smear modality. Benefit of doing so is to bypass the limiting environmental characteristics.

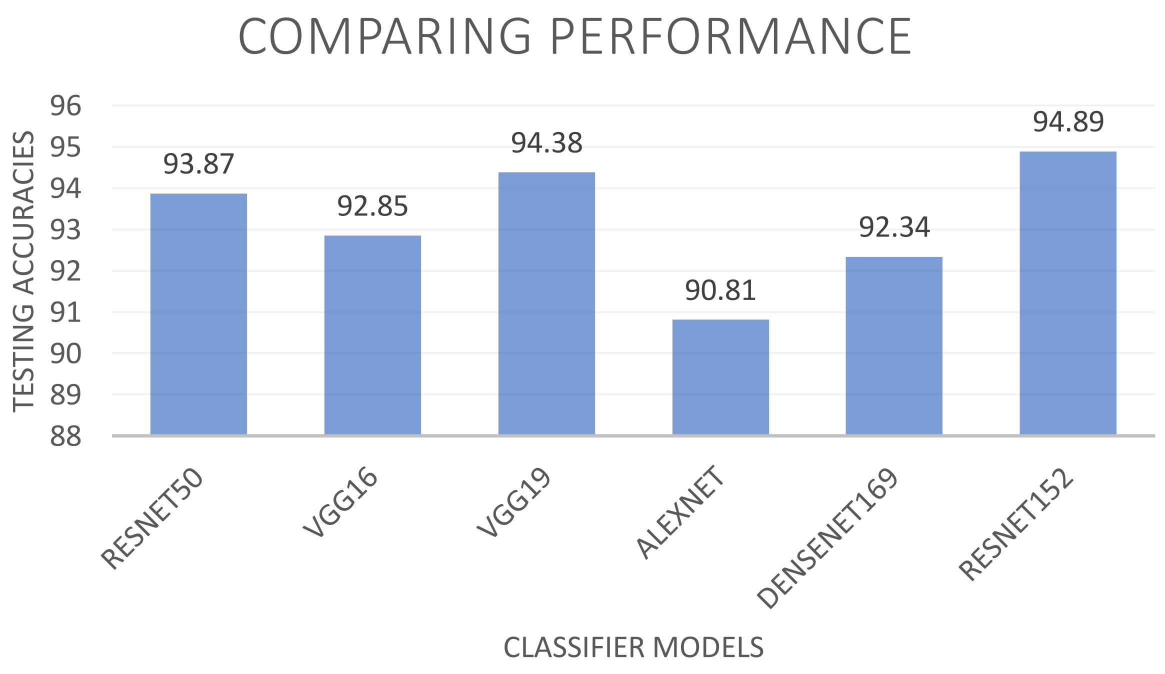


Figure 43: Comparison of Testing Accuracies for different model

REFERENCE

[1] S. D. R. N. N. A. D. Mrinal Kanti Bhowmik, "Nucleus Region Segmentation Towards Cervical Cancer Screening Using AGMC-TU Pap-Smear Dataset," in *International Conference on Pattern Recognition and Artificial Intelligence*, 2018.

[2] E. M. L. F. C. M. M. L. Z. A. Ferlay J, "Estimating the global cancer incidence and mortality," *International Journal of Cancer 144,* 2018.

[3] S. D. R. N. N. A. D. Mrinal Kanti Bhowmik, "Nucleus Region Segmentation Towards Cervical Cancer Screening Using AGMC-TU Pap-Smear Dataset," in *the International Conference*, 2018.

[4] N. M. G. S. Mithlesh Arya, "Cervical Cancer Detection Using Segmentation on Pap smear Images," in *the International Conference*, 2016.

[5] S. S. P. S. Dinshaw K.A., "Cancer control programme in India: Challenges for the new millennium," 2005.

[6] K. G. A. B. A. K. S. Srishti Gautam, "Unsupervised Segmentation of Cervical Cell Nuclei via Adaptive Clustering," in *Medical Image Understanding and Analysis*, 2017.

[7] D. S. H. W. L. M. K. Debbie Saslow, "American Cancer Society, American Society for Colposcopy and Cervical Pathology, and American Society for Clinical Pathology screening guidelines for the prevention and early detection of cervical cancer," *CA Cancer Journal for Clinicians,* vol. 62, pp. 47-172, 2012.

[8] P. A. V. M. M. S. S. Kumar Singh, " "Classification of clinical dataset of cervical cancer using KNN"," *Indian Journal of Science and Technology,* vol. 9, pp. 1-5, 2016.

[9] S. A. T. C. Nipon Theera Umpon, ""Automatic cervical cell segmentation and classification in Pap smears"," *Computer Methods and Programs in Biomedicine,* vol. 113, 2014.

[10] S. S. R. K. Rajeev Srivastava, "Detection and Classification of Cancer from Microscopic Biopsy Images Using Clinically Significant and Biologically Interpretable Features," *Journal of Medical Engineering,* 2015.

[11] A. C. O. K. a. D. I. F. M.E. Plissiti, "Automated segmentation of cell nuclei in PAP smear images," in *In proc. IEEE International Special Topic Conference on Information Technology in Biomedicine,* 2006.

[12] N. C. a. A. C. M. E. Plissiti, "Automated detection of cell nuclei in pap smear images using morphological reconstruction and clustering," in *IEEE Transactions on Information Technology in Biomedicine.*

[13] P. D. G. S. C. N. O. K. a. A. M. E. Plissiti, "Sipakmed: A New Dataset for Feature and Image Based Classification of Normal and Pathological Cervical Cells in Pap Smear Images," in *2018 25th IEEE International Conference on Image Processing (ICIP),* 2018.

[14] K. D. D. X. Fuzhen Zhuang Zhiyuan Qi, "A Comprehensive Survey on Transfer Learning," in *Proceedings of the IEEE,* 2021.

[15] S. G. Jeremy Howard, "fastai: A Layered API for Deep Learning," *Information 2020,* 2020.

[16] A. S. B. R. a. D. S. Juliet, "Transfer Learning with ResNet-50 for Malaria Cell-Image Classification," in *2019 International Conference on Communication and Signal Processing (ICCSP),* 2019.

[17] X. Z. S. R. a. J. S. K. He, "Deep Residual Learning for Image Recognition," in *016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Las Vegas, 2016.

[18] A. V. a. D. F. H. Qassim, "Compressed residual-VGG16 CNN model for big data places image recognition," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC),* Las Vegas, 2018.

[19] M. S. a. M. Pawar, "Transfer Learning for Image Classification," in *018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA),* Coimbatore, 2018.

[20] A. Tripathi, A. Arora and A. Bhan, "Classification of cervical cancer using Deep Learning Algorithm," 2021 5th International Conference on Intelligent Computing and Control

Systems (ICICCS), 2021, pp. 1210-1218, doi: 10.1109/ICICCS51141.2021.9432382.

[21] A. Arora, A. Tripathi and A. Bhan, "Classification of Cervical Cancer Detection using Machine Learning Algorithms," 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021, pp. 827-835, doi: 10.1109/ICICT50816.2021.9358570.