Ain Shams University
Faculty of Computer and Information Sciences
BIOINFORMATICS PROGRAM

Skin Diseases Detection

By

Team Members Names

Mohamed Ahmed Abd El-Naby
Abdullah Afify Abdul-hameed
Ziad Ahmed Kamal
Alaa Abd El-Gawad Salah
Noha Nabih Abd El-khalek

Under Supervision of

Dina Al-Sayad
Professor in Information System Department,
Faculty of Computer and Information Sciences,
Ain Shams University

TA Mohammed Ramadan
Assistant Lecturer in Bioinformatics Department
Faculty of Computer and Information Sciences,
Ain Shams University

Acknowledgement

First of all, we want to thank God for his blessings in our steps, success in this project, and all achievements in life. The project was done with great effort and a lot of passion by all the team members, we did our best to get the best results from the project and to achieve our goal which is to make this project as good as it can be.

we are so grateful for all the guidance we got from our supervisors, who without their support this project won’t become possible.

We want to show our gratitude to Dr. Dina Al-Sayad for her supervision and guidance throughout the whole year it has been a great chance to work under her supervision.

We are thanking the teacher assistant Mohamed Ramadan for his efforts with us, helping and devotion in guiding us.

We also want to thank all the professors of Ain Shams University "faculty of computer and information science - Bioinformatics Department" for their efforts and contribution in guiding us in the field of 'Information Technology' and 'Bioinformatics' ending with the full care they provided us with, to reach this stage of our life.

Sincerely we want to thank our families, friends, and colleagues for supporting and helping us.

Abstract

Skin cancer is an alarming disease for mankind. Early diagnosis of skin lesion significantly increases the prevalence of recovery.

skin cancer has been a major problem that should be discovered in its early phases because of the rapid growth rate of skin cancer, its high treatment costs, and its death rate.

These cancer cells are detected manually, and it takes time to cure in most of the cases.

We proposed a simple solution for detecting the skin cancer type only by taking a photo of the inflammatory area and simple information about the patient by using image processing and Artificial intelligence “deep learning” techniques and produced it in form of a web application.

The features of the affected skin cells are extracted after the segmentation of the dermoscopic images using differential image preprocessing technique.

Our goal is to develop an automated framework that efficiently performs a reliable automatic lesion classification to Eight skin lesion types :
(Melanoma - Melanocytic nevi - Basal cell carcinoma - Actinic keratosis - Benign keratosis-like lesions - Dermatofibroma - Vascular - Squamous cell carcinoma).

A deep learning-based method convolutional neural network classifier is used for the stratification of the extracted features.

An accuracy of **89.5%** and the training accuracy of **93.7%** have been achieved after applying the publicly available data set.

Table of Contents

List of Figures

List of Abbreviations

Al	Artificial Intelligence
AK	Actinic keratosis
BCC	Basal cell carcinoma
BKL	Benign keratosis-like lesion
CNN	Convolutional Neural Networks
ConvNets	Convolutional Neural Networks
CV	Computer Vision
DL	Deep Learning
DF	Dermatofibroma
FC	Fully Connected Layer
ML	Machine Learning
MEL	Melanoma
NV	Melanocytic nevi
OCR	Optical character recognition
Relu	The rectified linear activation function
SCC	Squamous cell carcinoma
UV	Ultraviolet radiation
VASC	Vascular

Chapter 1
Introduction

1.1 Motivation

Skin cancer is an alarming issue, and it must be detected as early as possible. The diagnostic is a manual process that is time-consuming as well as expensive. But todays' world science has become advanced by using machine learning and it can be helpful in many ways. Hence, Deep learning can make easy for detecting cancerous cells and that is why Deep learning especially convolutional neural network is used to detect cancerous cell more quickly, and efficiently.

According to the World Health Organization statistics, the number of people will be affected by the skin cancer will rise up to almost 13.1 million by 2030. Skin cancer is a condition in which there is an abnormal growth of melanocytic cells in the skin. Malignant melanoma class of skin cancer is generally caused from the pigment-containing cells known as melanocytes. Melanoma is found among white males and females, and results in approximately 75% of deaths associated with skin cancer.

1.2 Problem Definition

According to the world cancer report, the primitive reason of melanoma is ultra-violate light exposure in those people who have low level of skin pigment. The UV ray can be from the sun or any other sources and approximately 25% of malignant can be from moles.

1.3 Objective

The cardinal objective of this project is to develop state of the art Convolutional Neural Network (CNN) model to perform the classification of skin lesion images into respective cancer types. The model is trained and tested on the dataset made available by International Skin Imaging Collaboration (ISIC).

Neural Network algorithm is utilized to detect the benign and malignant. This framework is based on learning the images that are captured with dermatoscopic device to find out whether it is benign or malignant.

Convolutional Neural Network (CNN) is a type of neural network which is used in signal and image processing. Convolutional Neural Network is also used in Recommender System. CNN is chosen because it gives high accuracy in image processing. CNN has four working standards. The primary layer fills in as input layer where dermatologists give every one of the information they obtained. The input layer at that point forms the information and sends it to the next layers which are then send to the pooling layer.

The pooling layer pools the information structure by performing max pool or min pool. The pooling layer sends that information for smoothing to straighten layer which changes over the information to one-dimensional vector. At that point, the information gets into the thick layer to get changed over to the class they want which is for the situation benign or malignant. This paper represents an automatic skin cancer detection approach based on convolutional neural network to classify the cancer images into either malignant or benign melanoma.

1.4 Dataset (Cancerous Skin Diseases)

With the aim of both supporting clinical training and further technical research, which will eventually lead to automated algorithmic analysis, the International Skin Imaging Collaboration (ISIC) developed an international repository of dermoscopic images known as the ISIC Archive https://www.isic-archive.com .

Every year the ISIC increases its archive and promote a challenge to leverage the automated skin cancer detection.

For ISIC 2019, 25,331 dermoscopy images are available for training across 8 different categories. The test dataset is composed of 8,239 images and contains an additional outlier class not represented in the training data, which the new systems must be able to identify. Beside the images, the dataset also contains metadata for most of the images. The meta-data is composed of the patient's age and sex, and the region where the individual with the skin lesion is located. All these data come from the BCN_20000, HAM10000, and from an Anonymous' resources.

1.4.1 Actinic keratosis: 867 images

Actinic keratoses (also called solar keratoses) are dry scaly patches of skin that have been damaged by the sun. The patches are not usually serious. But there's a small chance they could become skin cancer, so it's important to avoid further damage to your skin.

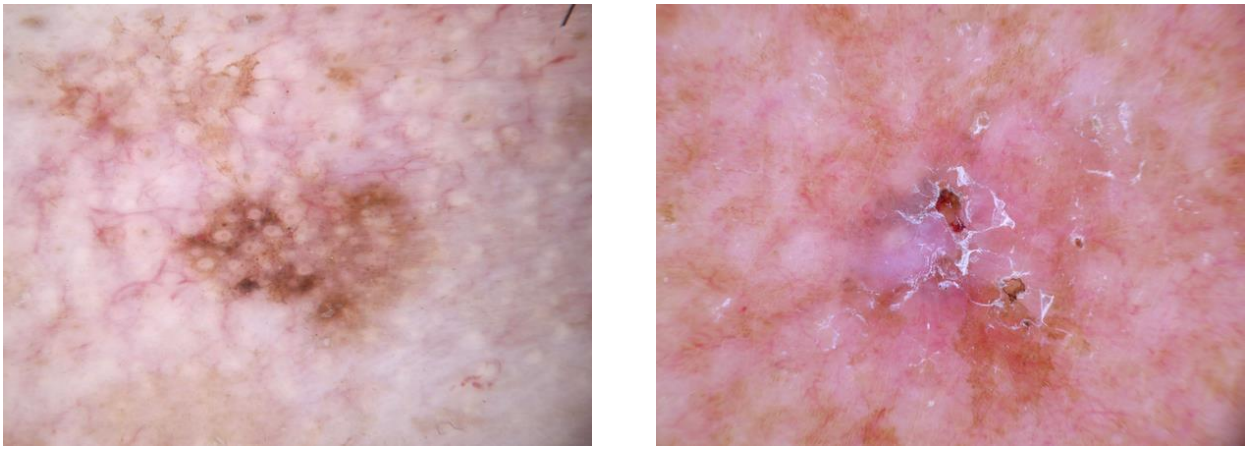


Fig 1: Actinic keratosis

1.4.2 Basal cell carcinoma: 3323 images

Basal cell carcinoma is a type of skin cancer that most often develops on areas of skin exposed to the sun, such as the face. On brown and Black skin, basal cell carcinoma often looks like a bump that's brown or glossy black and has a rolled border. Basal cell carcinoma is a type of skin cancer.
Subtypes of BCC Include:

- Nodular BCC
- Infiltrating BCC
- Superficial BCC

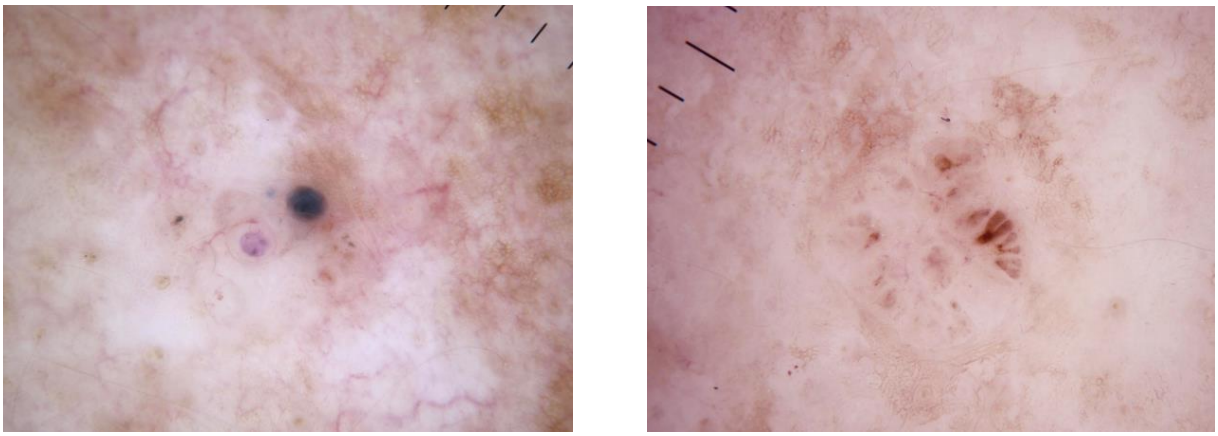


Fig 2: Basal cell carcinoma

1.4.3 Benign keratosis-like lesion : 2624 images

A seborrheic keratosis (seb-o-REE-ik ker-uh-TOE-sis) is a common noncancerous (benign) skin growth. People tend to get more of them as they get older. Seborrheic keratoses are usually brown, black or light tan. The growths (lesions) look waxy or scaly and slightly raised.

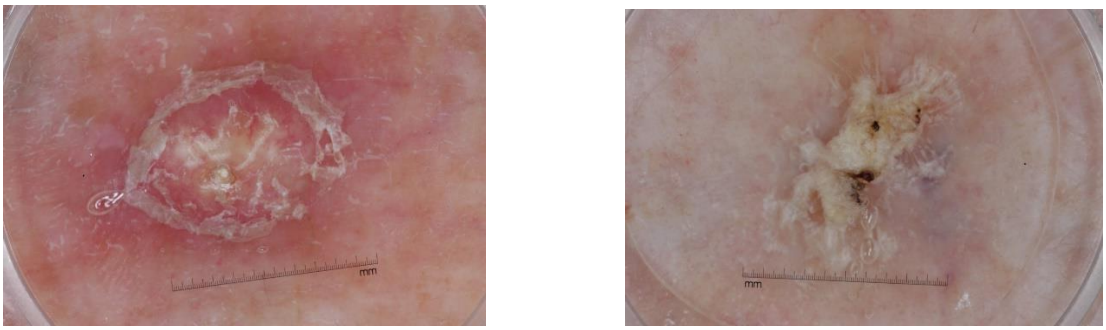


Fig 3: Benign keratosis-like lesion

1.4.4 Dermatofibroma: 239 images

Dermatofibroma (superficial benign fibrous histiocytoma) is a common cutaneous nodule of unknown etiology that occurs more often in women. Dermatofibroma frequently develops on the extremities (mostly the lower legs) and is usually asymptomatic, although pruritus and tenderness can be present.

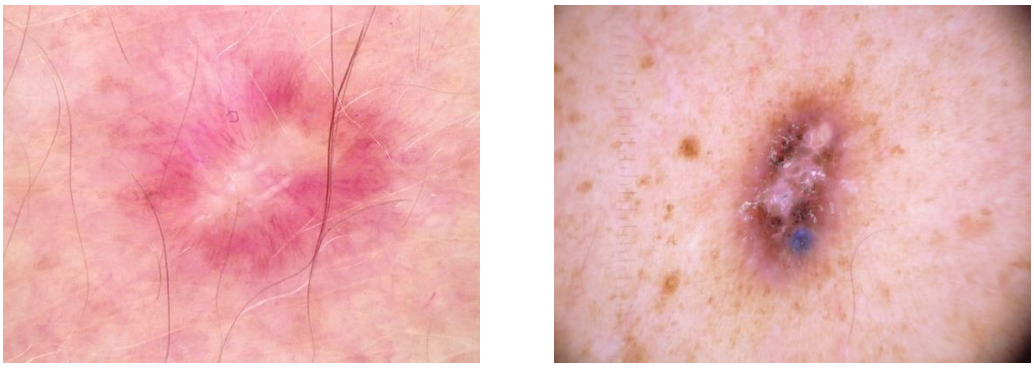


Fig 4: Dermatofibroma

1.4.5 Melanoma: 4522 images

Melanoma is a serious form of skin cancer that begins in cells known as melanocytes. While it is less common than basal cell carcinoma and squamous cell carcinoma, melanoma is more dangerous because of its ability to spread to other organs more rapidly if it is not treated at an early stage.

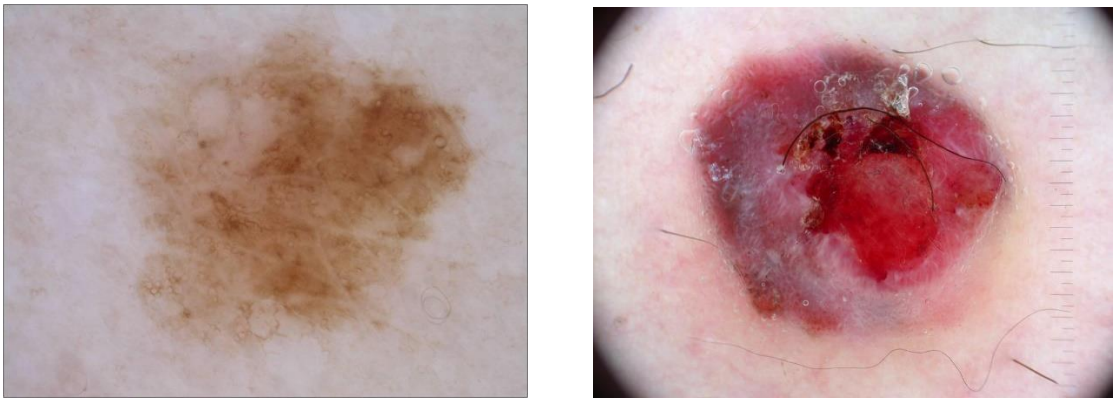


Fig 5: Melanoma

1.4.6 Melanocytic nevi: 12875 images

A melanocytic naevus (American spelling 'nevus'), or mole, is a common benign skin lesion due to a local proliferation of pigment cells (melanocytes). It is sometimes called a naevocytic naevus or just 'naevus' (but note that there are other types of naevi).

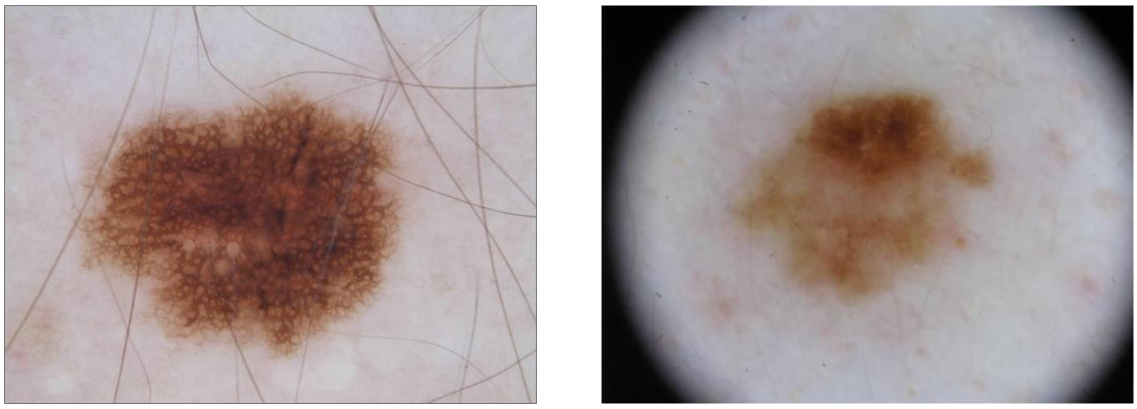


Fig 6: Melanocytic nevi

1.4.7 Vascular: 253 images

The vascular system, also called the circulatory system, is made up of the vessels that carry blood and lymph through the body. The arteries and veins carry blood throughout the body, delivering oxygen and nutrients to the body tissues and taking away tissue waste matter.

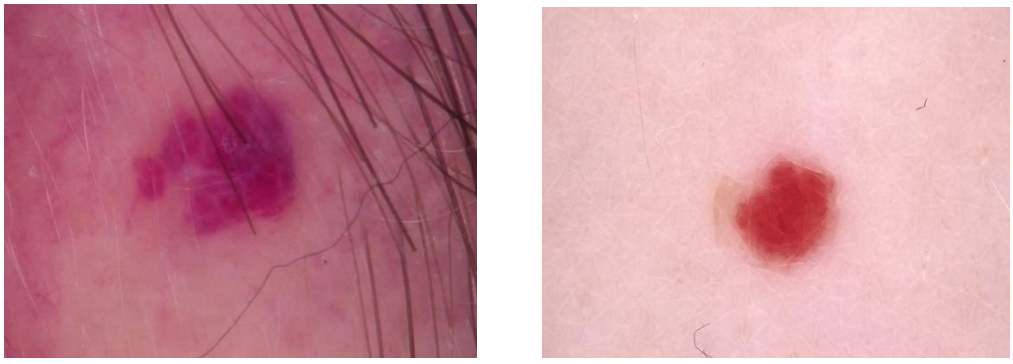


Fig 7: Vascular

1.4.8 Squamous cell carcinoma: 628 images

Squamous cell carcinoma (SCC) of the skin also known as cutaneous squamous cell carcinoma (cSCC) is the second most common form of skin cancer, characterized by abnormal, accelerated growth of squamous cells. When caught early, most SCCs are curable.

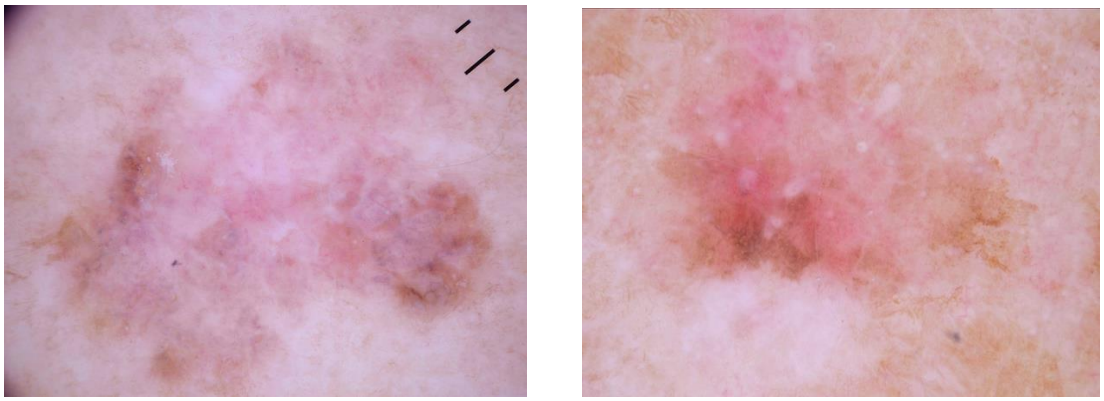


Fig 8: Squamous cell carcinoma

1.5 Document Organization

This document divided to six chapters describing all aspects of our project, and the description of how each chapter is written in the document comes as follows:

- Chapter 2 (Background):

  The project field, its scientific background and contain the objective of the system.
- Chapter 3 (analysis and design):

  Describes the Functional requirement, user of the system and all system analysis diagrams and the specification of each one.
- Chapter 4 (Implementation):

  Describes in detail the algorithms and techniques used to implement the project.
- Chapter 5 (user manual):

  A full instruction and devices that describe how to use the "Skin Cancer Detection" web application.
- Chapter 6 (conclusion and future work):

  Our conclusion after implanted project and what we have reached And our future vision of this project.

Chapter 2

Background

2.1 Skin Cancer

Skin cancer is the most common form of cancer in the United States. Skin cancer is generally classified as nonmelanoma skin cancer (NMSC) or melanoma. The exact incidence of skin cancer is difficult to determine due to the lack of diagnostic criteria and underreporting. However, several epidemiologic studies have shown an increasing incidence of both NMSC and melanoma over the past several decades. The diagnosis and treatment of these neoplasms represent a significant health problem from the standpoint of patient well-being and healthcare expenditures. Skin cancers are frequently located on the sun-exposed head and neck regions, which can result in significant morbidity during their diagnosis and treatment. Treatment options including surgical excision, cryotherapy, chemotherapy, immunotherapy, and radiation. Proper sun safety (i.e., sunscreen) is of the utmost importance to prevent skin cancer.

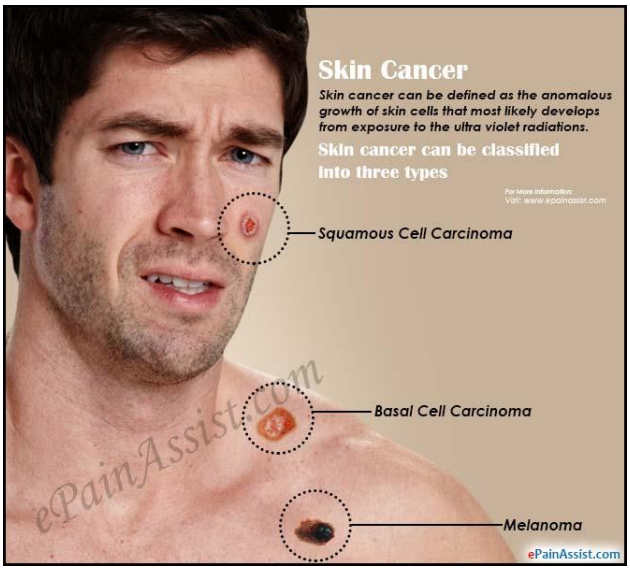


Fig 9: different Cancerous Cells

Etiology:

Ultraviolet (UV) solar radiation is the major etiologic factor in the development of cutaneous malignancies. An overwhelming majority of cases of NMSC and melanoma are related to UV exposure.

UV exposure induces carcinogenesis by a twofold mechanism; it generates DNA damage that leads to mutation formation and reduces the ability of the host immune system to recognize and remove malignant cells. Basal cell carcinoma

(BCC) and squamous cell carcinoma (SCC) are the most common forms of NMSC, and both are derived from mutated epidermal keratinocytes. The cumulative lifetime UV exposure is directly correlated with risk for developing BCC and SCC. Melanoma is the deadliest form of skin and is derived from mutated melanocytes. In contrast to BCC and SCC, the risk of developing melanoma is directly correlated with sun exposure during adolescence, specifically the number of sunburns from age 15-20. Other risk factors implicated in the development of cutaneous malignancies include family history, chemical exposure, tanning bed use, human papillomavirus (HPV), Fitzpatrick skin type, the presence of melanocytic nevi, and immunosuppressed status.

Melanomas can grow quickly, sometimes within a matter of weeks. However, many smartphone apps are available that claim to identify and track moles and lesions that could potentially become skin cancer.

According to the Academies of Dermatology, skin cancer is the most common type of cancer in the World. In fact, more than 10,000 people receive a diagnosis every day.

Early detection makes skin cancer more treatable and, for some people, completely reversible. Identifying and managing skin cancer before it spreads is the best way to achieve a good outcome.

Some people use a skin cancer app to increase their chance of identifying skin cancer as early as possible.
From this point we decide to continue this way and develop application that detect some of skin cancer, to make easy for patients to get initial diagnosis with high accuracy.

2.2 Overview of Machine Learning

Machine learning (ML) is an area that aims to construct new algorithms to make predictions based on given data. ML generates general models using training data so that these models can detect the presence or the absence of a pattern in test (new) data. In the case of images like in this thesis, training data can be in the form of images, regions, or pixels which are labeled or not. Patterns can be a low-level

or high-level. For instance, a low-level pattern can be a label for pixels in segmentation while high-level pattern can be the presence or the absence of a disease in a medical image. In this case, the image classification becomes the addressed problem with a training set containing image-label pairs.

2.2.1. Categories of Machine Learning Algorithms

Machine learning algorithms can be classified into three key categories based on the different types of learning problems addressed. A list of these categories is:

- Supervised Learning: In supervised learning, the training dataset needs to be in a specific format. Each instance (data point) has an assigned label. Datasets are labeled as $(x, y) \in X \times Y$, where x and y denote a data point and the corresponding true prediction for x. If the output y is part of a discrete domain, the problem is a classification task. If the output belongs to a continuous domain, then it is a regression task.
- Unsupervised Learning: Unlike supervised learning, the datasets are not labeled in unsupervised learning. In order to develop a structure from unlabeled data, the ML algorithm should examine the similarities between object pairs.
- Semi-supervised Learning: This learning task is a class of supervised learning and uses a large amount of unlabeled data for training along with the small amount of labeled data

2.3 Neural Network

Biological neural network is an important part of the human brain. It is a highly complex system and has the ability to process different tasks simultaneously. Neural network (NN) is a classifier that simulates the human brain and neurons. Instead of neurons, “perceptron” is used as a basic unit of NN. NN architecture consists of the different layers: the input layer containing input feature vector(s), the output layer that comprises of the neural network response, and the layer containing neurons (perceptrons) between the input and output layers.

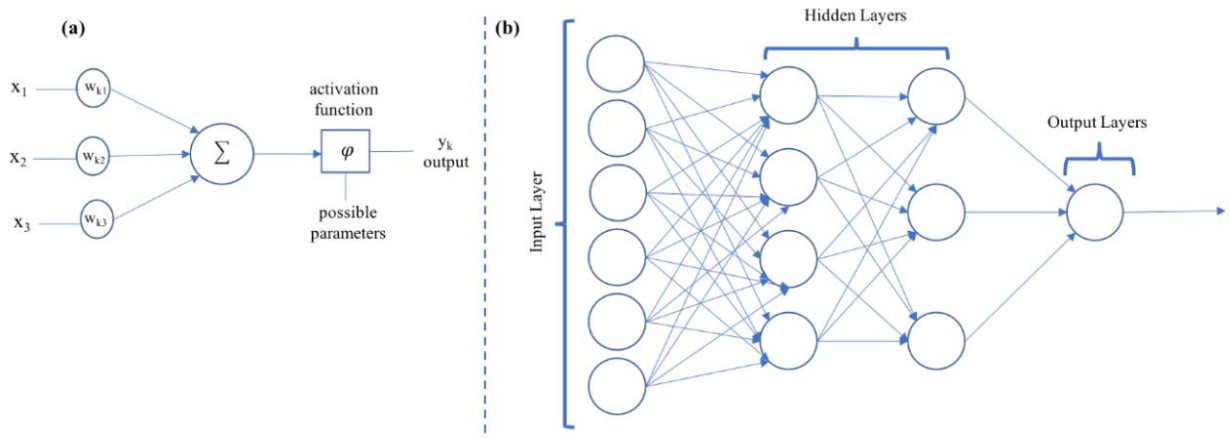


Figure 10: (a) is the perceptron layer and (b) is the image of Multi-layer Neural Network.

According to the McCulloch-Pitts model, the neuron k receives m input parameter x_j. The neuron also has m weight parameter w_{kj} .The sum of inputs and weights is combined and fed into an activation function j which produces the output y_k of the neuron as seen in Figure. The Equation 2 below gives the mathematical understanding of neural networks.
Equation:

$$Y_k = \Phi \sum_{J=0}^{m} w_{kj} x_j$$

A neural network can learn the estimated target outputs after training by selecting the weights of all neurons. However, it is challenging to analytically solve neuron weights of a multi-layer network. In order to solve the weights iteratively in a simple and effective way, the back-propagation algorithm is used. This algorithm calculates a gradient that is needed in the calculation of the weights. The back-propagation algorithm can be divided into two phases: propagation and weight update. In the first phase of this algorithm, an input vector is propagated forward through the neural network, and the output value is generated. After that, the cost (error term) is calculated. Then, the error values are propagated back to the network to calculate the cost of the hidden layer neurons. In the second phase of the algorithm, the neuron weights are updated by calculating the gradient of weights and subtracting the ratio of gradient of weights from the current weights. This ratio is called the learning rate. After the update of weights, the algorithm continues with different inputs until the weights are converged

2.4 Deep Learning

Deep learning is a subset of machine learning (ML), which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability allowing it to learn from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

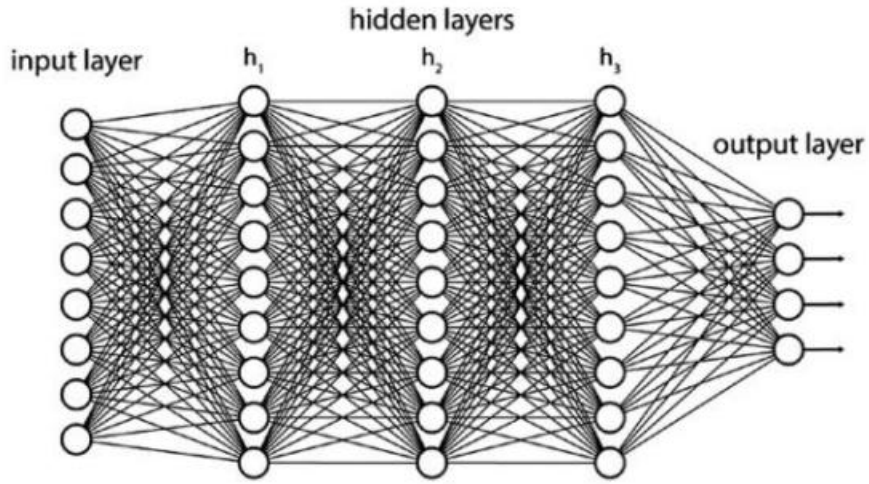


Fig 11: Neural Network

Deep learning models are performed by Multiple Layers which in the context of the Artificial Neural Networks have more than 2 hidden layers.

There are 2 different types of Deep Learning model, Deep model and Shallow model, Deeper models perform better than Shallow models.

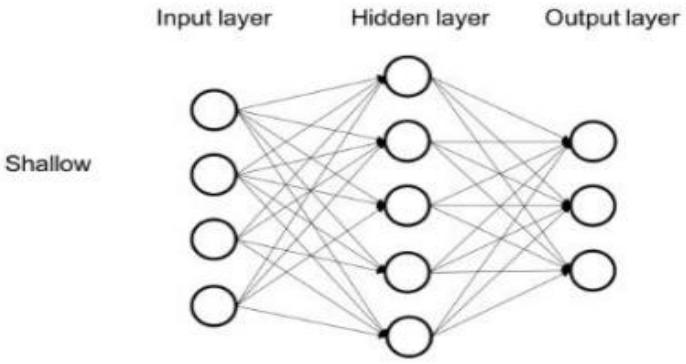


Fig 12: Shallow neural network

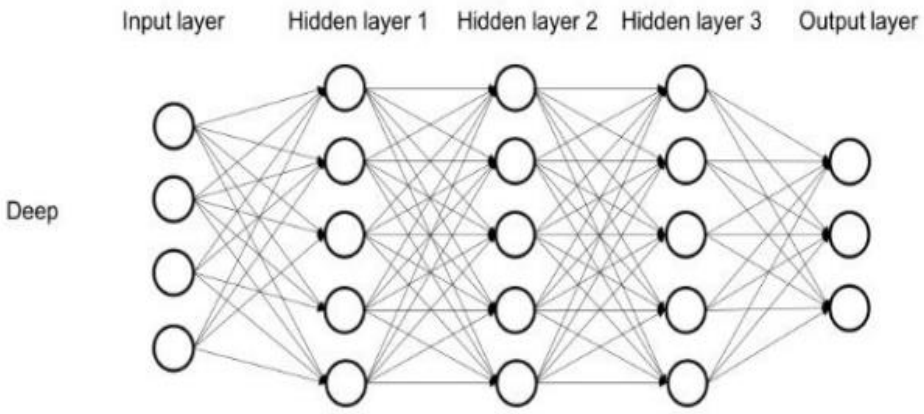


Fig 13: Deep neural network

Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called visible layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.

Another process called backpropagation uses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate.

The above describes the simplest type of deep neural network in the simplest terms. However, deep learning algorithms are incredibly complex, and there are different types of neural networks to address specific problems or datasets. For example,

- Convolutional neural networks (CNNs), used primarily in computer vision and image classification applications, can detect features and patterns within an image, enabling tasks, like object detection or recognition. In 2015, a CNN bested a human in an object recognition challenge for the first time.

- Recurrent neural network (RNNs) is typically used in natural language and speech recognition applications as it leverages sequential or times series data.

Benefits of Deep Learning over Machine Learning:

Because it performs better at extracting non-local and global relationships and patterns in data, compared to relatively shallow learning architectures.
Other useful characteristics of learnt abstract representations by deep learning include: it tries to explore huge datasets, continuous improvement as adding more training data, automatic data representation extraction.

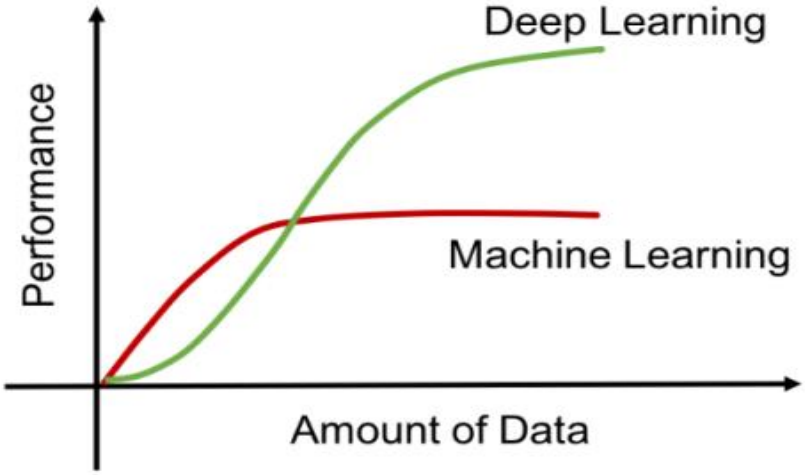


Fig 14: Deep learning vs Machine learning

2.5 Artificial Neural Networks

As we mentioned in above that the idea behind all of this was inspired by how human brain works in processing informations and how the distributed nodes communicating together, mainly they are made of connected units called artificial neurons which simulate the neurons biological human brain, these artificial neurons can send and receive signals from other neurons and process it.
The differences between ANNs are static and symbolic meanwhile human brains are analog and dynamic. The signal at a connection is represented as a real number and the output of each neuron is computed by some non-linear function of the sum of its inputs

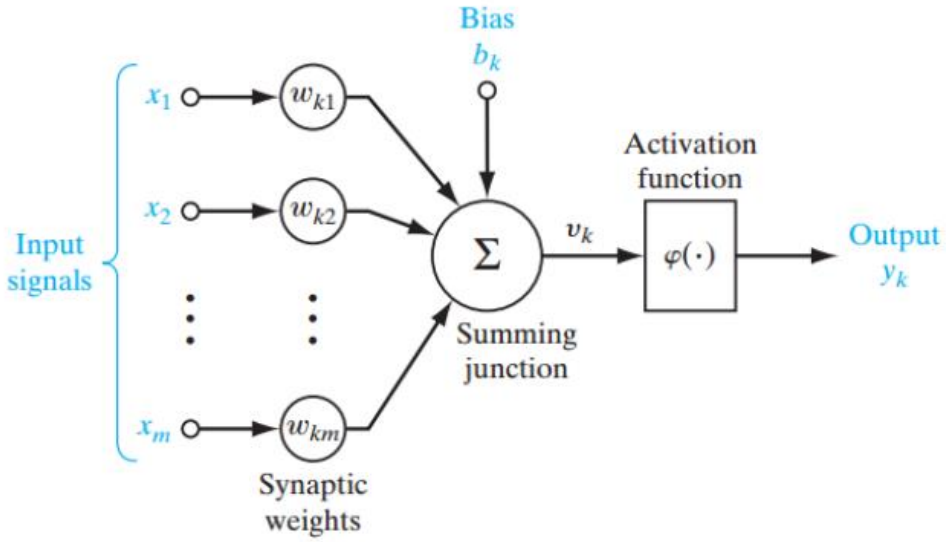


Fig 15: Artificial neural network

2.6 Convolutional Neural Networks

In the context of computer vision, the most commonly applied artificial neural network is a convolutional neural network (CNN). There are two main reasons why CNNs are used in computer vision problems. Firstly, with traditional NNs, solving the computer vision problem for even relatively small sized images is challenging. For example, a monochrome 750x563 image contains 422,250 pixels. If this image is polychrome, the number of pixels is typically multiplied by three which is the typical amount of color channels, and in this case, the image would have 1,266,750 pixels and the same number of weights. Consequently, the overall number of free parameters in NN quickly becomes extremely large which causes overfitting and reduces the performance. Additionally, CNNs require comparatively little image pre-processing compared to other image classification algorithms, which means CNNs can learn the filters by itself. The CNN consists of input and output layers as well as the multiple hidden layers. The hidden layers are usually made of convolutional layers, pooling layers, and fully connected layers.

CNN has:

- Convolutional Layers: These layers pass the results of the input to the next layer. It simulates the response of a neuron to visual stimuli.
- Pooling Layers: These layers combine the outputs of neuron clusters at one layer into a single neuron in the next layer. The purpose of this layer is to reduce the parameters and computation in network.
- RELU Layers.
- Fully Connected Layers: These layers connect each and every neuron in one layer to every neuron in another layer.

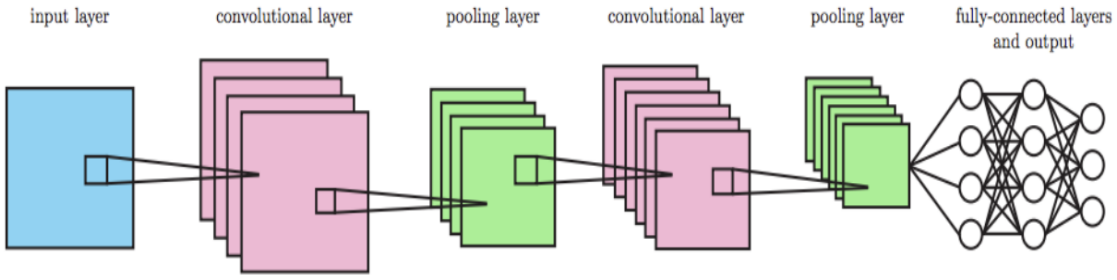


Fig 16: An Example of a convolutional neural network

2.6.1 The convolution layer (Kernel):

is the core building block of the CNN. It carries the main portion of the network's computational load.

This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.

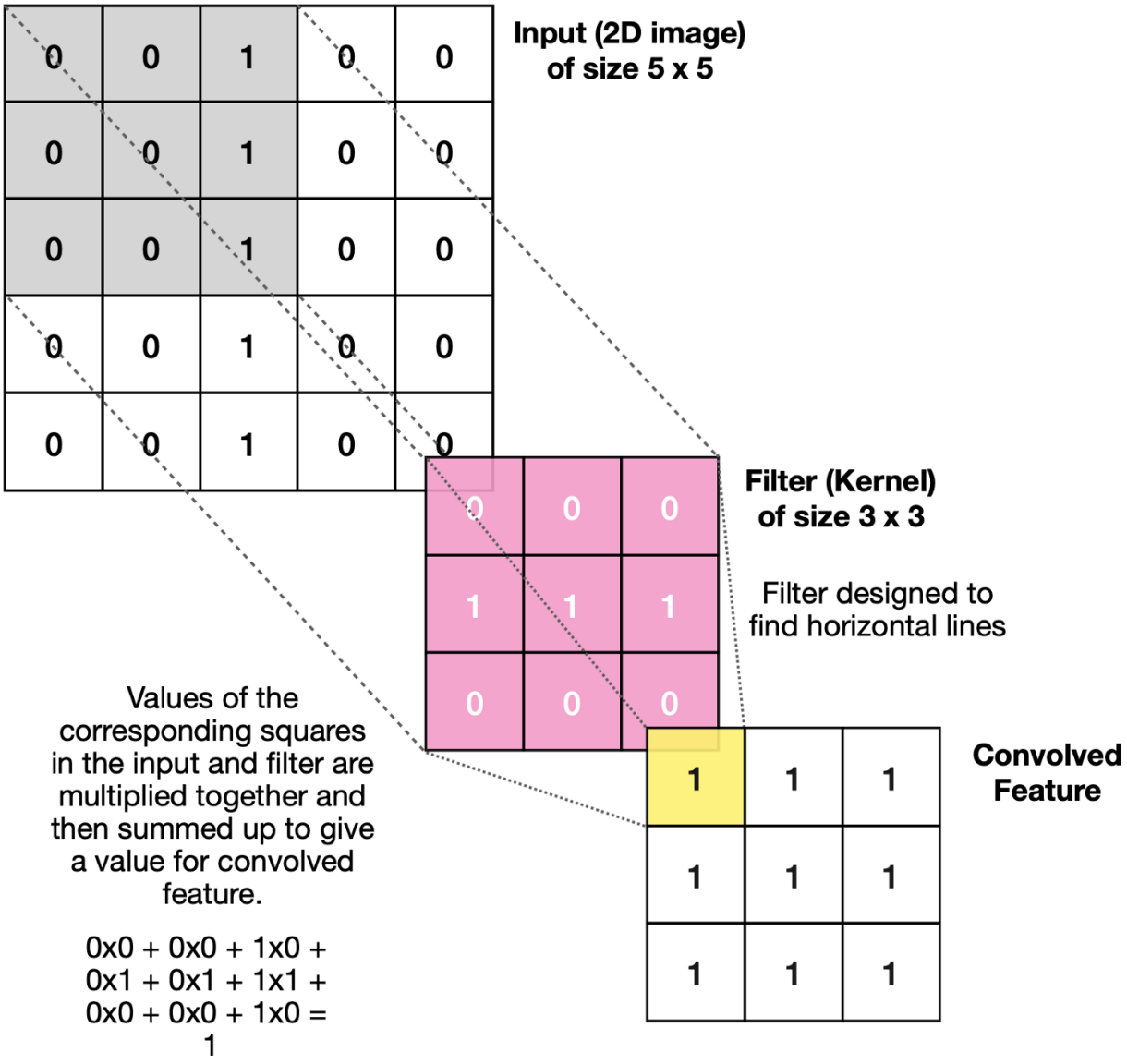


Fig 17: Illustration of Convolution Operation

During the forward pass, the kernel slides across the height and width of the image-producing the image representation of that receptive region. This produces a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image. The sliding size of the kernel is called a stride. If we have an input of size W x W x D and D_{out} number of kernels with a spatial size of F with stride S and amount of padding P, then the size of output volume can be determined by the following formula:

$$W_{out} = \frac{(W-F+2P)}{S} + 1$$

Formula for Convolution Layer

2.6.2 Motivation behind Convolution:

Convolution leverages three important ideas that motivated computer vision researchers: sparse interaction, parameter sharing, and equivariant representation.

Trivial neural network layers use matrix multiplication by a matrix of parameters describing the interaction between the input and output unit. This means that every output unit interacts with every input unit. However, convolution neural networks have sparse interaction. This is achieved by making kernel smaller than the input e.g., an image can have millions or thousands of pixels, but while processing it using kernel, we can detect meaningful information that is of tens or hundreds of pixels. This means that we need to store fewer parameters that not only reduces the memory requirement of the model but also improves the statistical efficiency of the model.

If computing one feature at a spatial point (x1, y1) is useful then it should also be useful at some other spatial point say (x2, y2). It means that for a single two-dimensional slice i.e., for creating one activation map, neurons are constrained to use the same set of weights. In a traditional neural network, each element of the weight matrix is used once and then never revisited, while convolution network has shared parameters i.e., for getting output, weights applied to one input are the same as the weight applied elsewhere.

Due to parameter sharing, the layers of convolution neural network will have a property of equivariance to translation. It says that if we changed the input in a way, the output would also get changed in the same way.

2.6.3 Pooling Layer

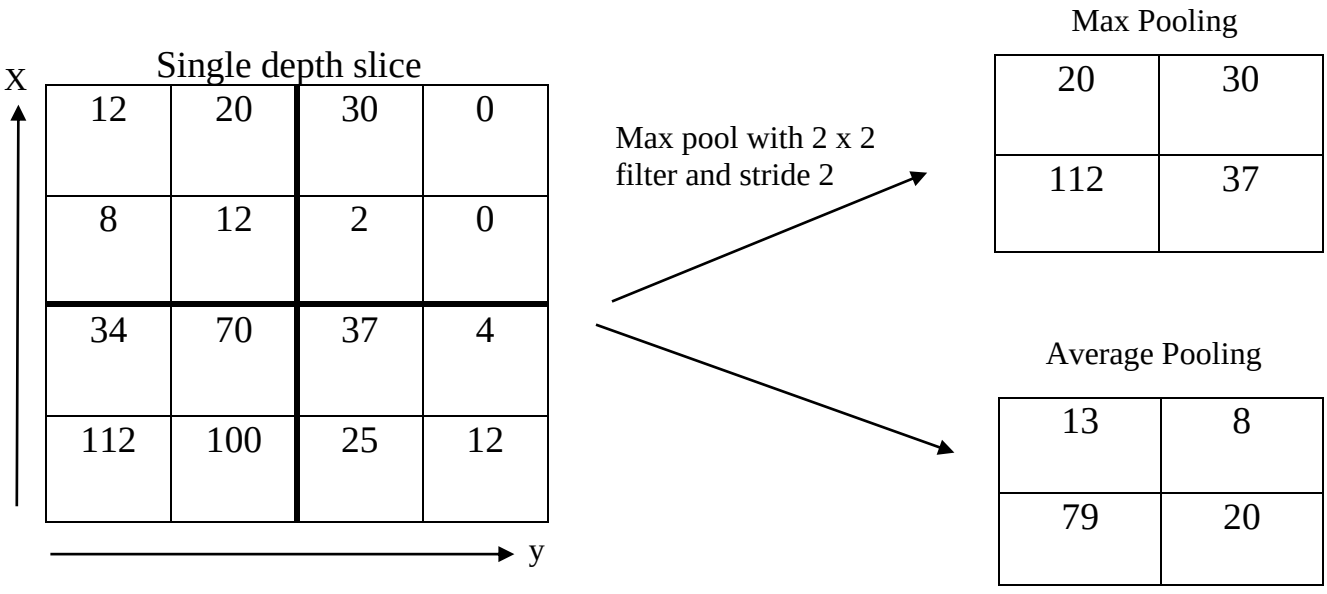
The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.
There are several pooling functions such as the average of the rectangular neighborhood, L2 norm of the rectangular neighborhood, and a weighted average based on the distance from the central pixel. However, the most popular process is max pooling, which reports the maximum output from the neighborhood.

Types of Pooling:

There are mainly two different types of Pooling which are as follows:

1. Max Pooling:
   The Max Pooling basically provides the maximum value within the covered image by the Kernel.

2. Average Pooling:
   The Average Pooling provides and returns the average value within the covered image by the Kernel.



The other functionality of Max Pooling is also noise-suppressing, as it works on discarding those activations which contain noisy activation. And on the other side, the Average Pooling simply works on the mechanism of noise-suppressing by dimensionality reduction. So, in short words, we can conclude that Max Pooling works more efficiently than Average Pooling.
The Convolutional Layer, altogether with the Pooling layer, makes the "i-th layer" of the Convolutional Neural Network. Entirely reliant on the image intricacies, the layer counts might be rise-up for the objective of capturing the details of the detailed level, but also needs to have more computational power. After analyzing the above-described information about the process, we can easily execute the model for understanding the features. Moreover, here we are about to get the

output and then provide it as an input for the regular Neural Network for further classification reasons.

If we have an activation map of size W x W x D, a pooling kernel of spatial size F, and stride S, then the size of output volume can be determined by the following formula:

$$\mathbf{W_{out}} = \frac{\boldsymbol{W-F}}{\boldsymbol{S}} + \mathbf{1}$$

Formula for Padding Layer

This will yield an output volume of size W_{out} x W_{out} x D.

In all cases, pooling provides some translation invariance which means that an object would be recognizable regardless of where it appears on the frame.

2.6.4 Flatten Layers

It involves taking the pooled feature map that is generated in the pooling step and transforming it into a one-dimensional vector. Here is a visual representation of what this process looks like:

1	1	0
4	2	1
0	2	1

Flattening →

1
1
0
4
2
1
0
2
1

2.6.5 Fully Connected Layer

The FC layer helps to map the representation between the input and the output.

The addition of the FC layer is mostly the easiest way for the learning purpose of the non-linear combinations of the abstract level structures, as it is also revealed by the output of the convolutional layer. The FC layer provides the space for learning non-linear functions. As now we have achieved our task to convert our image output into a specific form of Multi-layer Perceptron, now we must flatten the output image into a form of a column vector. Over the different eras of epochs, the model is basically succeeded for the distinguishing function between the dominating and low-level features.

2.6.6 Non-Linearity Layers

Since convolution is a linear operation and images are far from linear, non-linearity layers are often placed directly after the convolutional layer to introduce non-linearity to the activation map.
There are several types of non-linear operations, the popular ones being:

1. **Sigmoid**

   The sigmoid non-linearity has the mathematical form $\sigma(\kappa) = 1/(1+e' \kappa)$. It takes a real-valued number and squashes it into a range between 0 and 1. However, a very undesirable property of sigmoid is that when the activation is at either tail, the gradient becomes almost zero. If the local gradient becomes very small, then in backpropagation it will effectively “kill” the gradient. Also, if the data coming into the neuron is always positive, then the output of sigmoid will be either all positives or all negatives, resulting in a zig-zag dynamic of gradient updates for weight.

2. **Tanh**

   Tanh squashes a real-valued number to the range [-1, 1]. Like sigmoid, the activation saturates, but — unlike the sigmoid neurons — its output is zero centered.

3. **ReLU**

   The Rectified Linear Unit (ReLU) has become very popular in the last few years. It computes the function $f(\kappa)=\max(0,\kappa)$. In other words, the activation is simply threshold at zero.

   In comparison to sigmoid and tanh, ReLU is more reliable and accelerates the convergence by six times.

Unfortunately, a con is that ReLU can be fragile during training. A large gradient flowing through it can update it in such a way that the neuron will never get further updated. However, we can work with this by setting a proper learning rate.

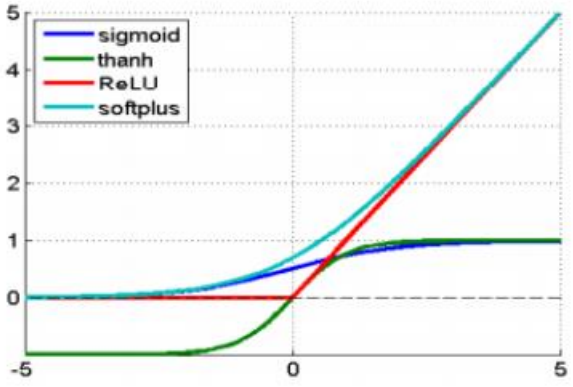


Fig 18: Common types of non-linearity

Our convolutional neural network architecture would look something like:

[INPUT] → [Convolution 1] → [BATCH NORMALIZATION] → [ReLU] → [POOL 1] → [Convolution 2] → [BATCH NORM] → [ReLU] → [POOL 2] → [Fully Connected LAYER] → [RESULT]

A CNN Convolves learned features with input data uses 2D convolutional layer. This means that this type of network is ideal for processing 2D images. Compared to other image classification algorithms, CNNs actually use very little preprocessing. This means that they can learn the filters that have to be hand-made in other algorithms. CNNs can be used in tons of applications from image and video recognition, image classification, and recommender systems to natural language processing and medical image analysis.

CNNs have an inputs layer, output layer and hidden layers. The hidden layers usually consist of convolutional layer, RELU layers, pooling layers, fully connected layers.

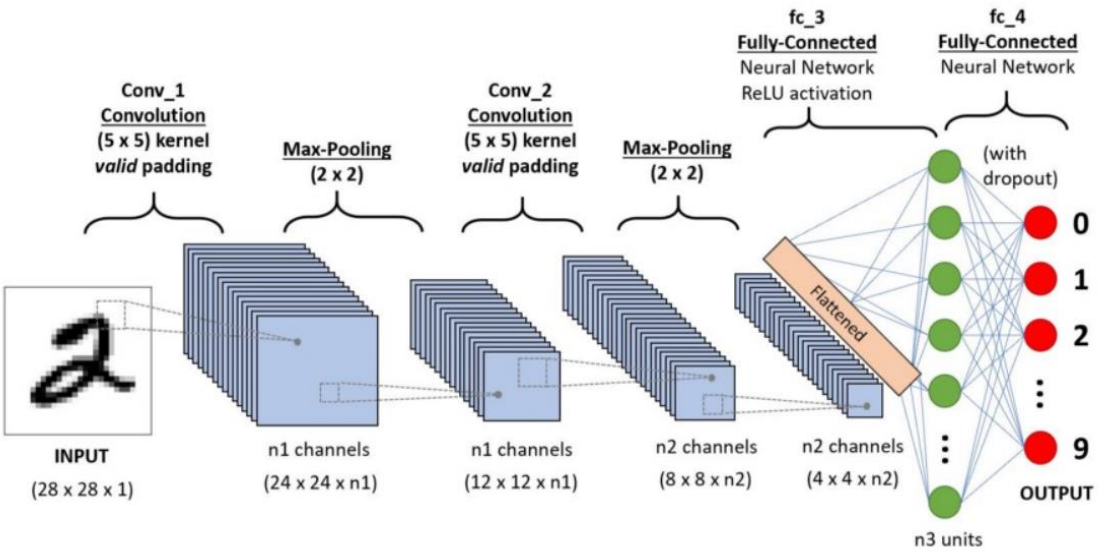


Fig 19: Feed-Forward Neural Network

- Convolutional layers apply a convolution operation to the input. This passes information on to the next layer.
- Pooling combines the outputs of clusters of neurons into a single neuron in the next layer.
- Fully connected layer connect very neuron in one layer to every neuron in the next layer.

In a convolutional layer, neuron only receive input from a subarea of the previous layer. In a fully connected layer, each neuron receives input from every element of the previous layer.

A CNN works by extracting features from image. This eliminates the need for manual feature extraction. The feature is not trained they are learned while the network trains on a set of images. This makes deep learning models extremely accurate for computer vision tasks. Each layer increases the complexity of the learned features.

CNN:

- Start with an input image
- Applies many different filters to it to create a feature map
- Applies a RELU function to increase non-linearity
- Applies pooling layer to each feature map
- Flatten to pooled image into one long vector

- Input the vector into a fully connected artificial neural network
- Processes the feature through the network. The final fully connected layer provides the "Voting" of the classes that we are after.

Here are some impressive examples of CNN architectures:

- AlexNet
- GoogLeNet
- ZFNet
- LeNet
- ResNet

2.7 Transfer learning

Transfer learning is a machine learning method where we reuse a pre-trained model as the starting point for a model on a new task.

To put it simply—a model trained on one task is repurposed on a second, related task as an optimization that allows rapid progress when modeling the second task.
By applying transfer learning to a new task, one can achieve significantly higher performance than training with only a small amount of data.

Transfer learning is so common that it is rare to train a model for an image or natural language processing-related tasks from scratch.

Instead, researchers and data scientists prefer to start from a pre-trained model that already knows how to classify objects and has learned general features like edges, shapes in images.

ImageNet, AlexNet, and Inception are typical examples of models that have the basis of Transfer learning.

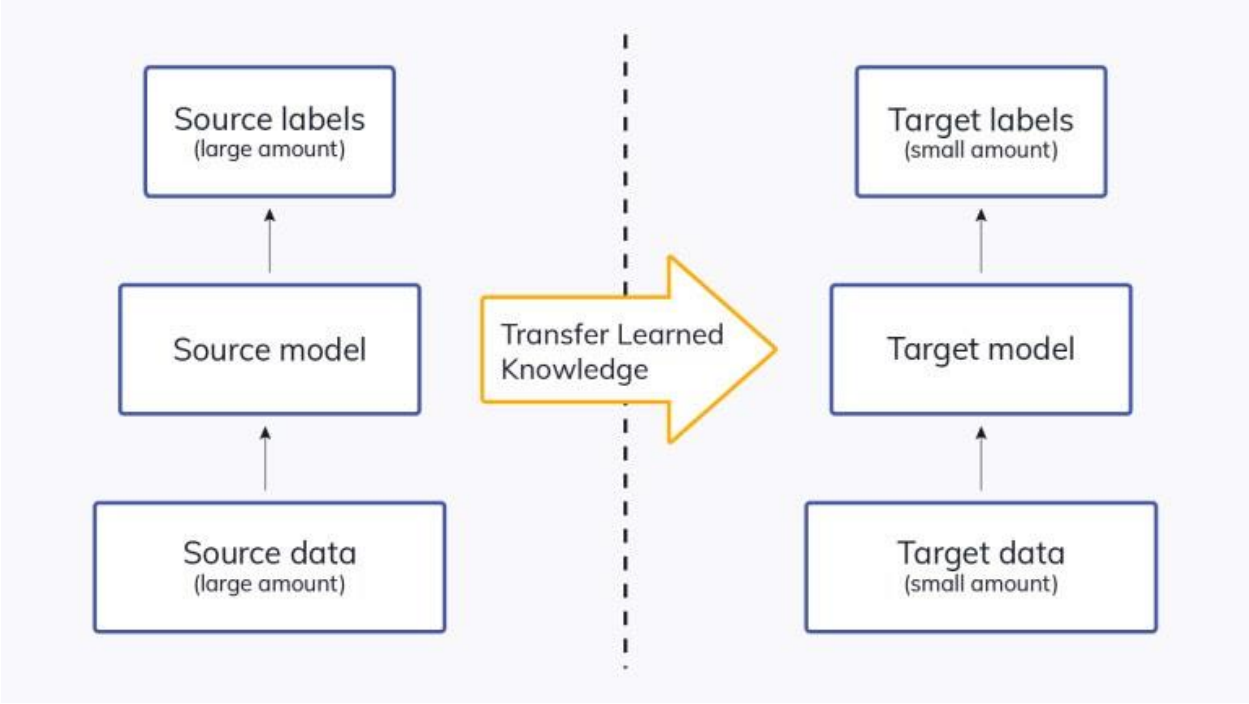


Fig 20: Transfer Learning

Traditional Machine Learning vs. Transfer Learning:

Deep learning experts introduced transfer learning to overcome the limitations of traditional machine learning models. Let's have a look at the differences between the two types of learning.

1. Traditional machine learning models require training from scratch, which is computationally expensive and requires a large amount of data to achieve high performance. On the other hand, transfer learning is computationally efficient and helps achieve better results using a small data set.
2. Traditional ML has an isolated training approach where each model is independently trained for a specific purpose, without any dependency on past knowledge. Contrary to that, transfer learning uses knowledge acquired from the pre-trained model to proceed with the task. To paint a better picture of it:

   One can not use the pre-trained model of ImageNet with biomedical images because ImageNet does not contain images belonging to the biomedical field.

3. Transfer learning models achieve optimal performance faster than the traditional ML models. It is because the models that leverage knowledge (features, weights, etc.) from previously trained models already understand the features. It makes it faster than training neural networks from scratch.

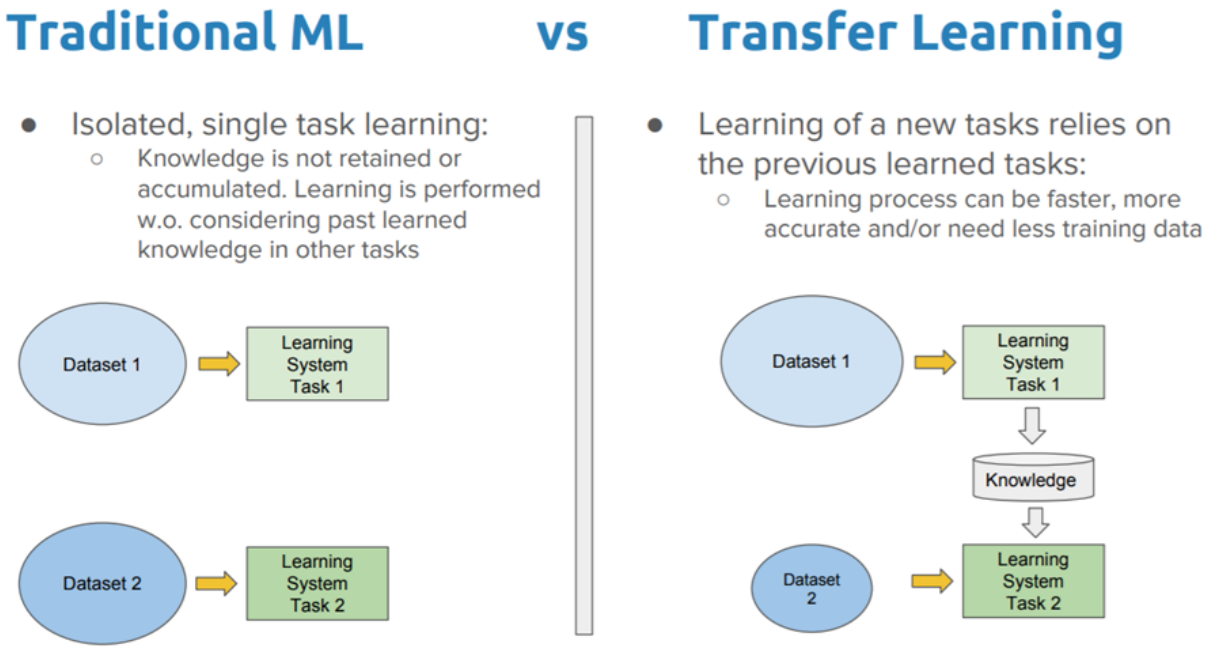


Fig 21: Traditional Machine Learning vs. Transfer Learning

Transfer Learning in 6 steps:

1. **Obtain pre-trained model:**

   The first step is to choose the pre-trained model we would like to keep as the base of our training, depending on the task. Transfer learning requires a strong correlation between the knowledge of the pre-trained source model and the target task domain for them to be compatible.
   Some of the pre-trained models:

   - For computer vision:
     - VGG-16
     - VGG-19
     - Inception V3
     - XCeption
     - ResNet-50

- For NLP tasks:
    - Word2Vec
    - GloVe
    - FastText

2. **Create a base model:**

   The base model is one of the architectures such as ResNet or Xception which we have selected in the first step to be in close relation to our task. We can either download the network weights which saves the time of additional training of the model. Else, we will have to use the network architecture to train our model from scratch.

   There can be a case where the base model will have more neurons in the final output layer than we require in our use case. In such scenarios, we need to remove the final output layer and change it accordingly.

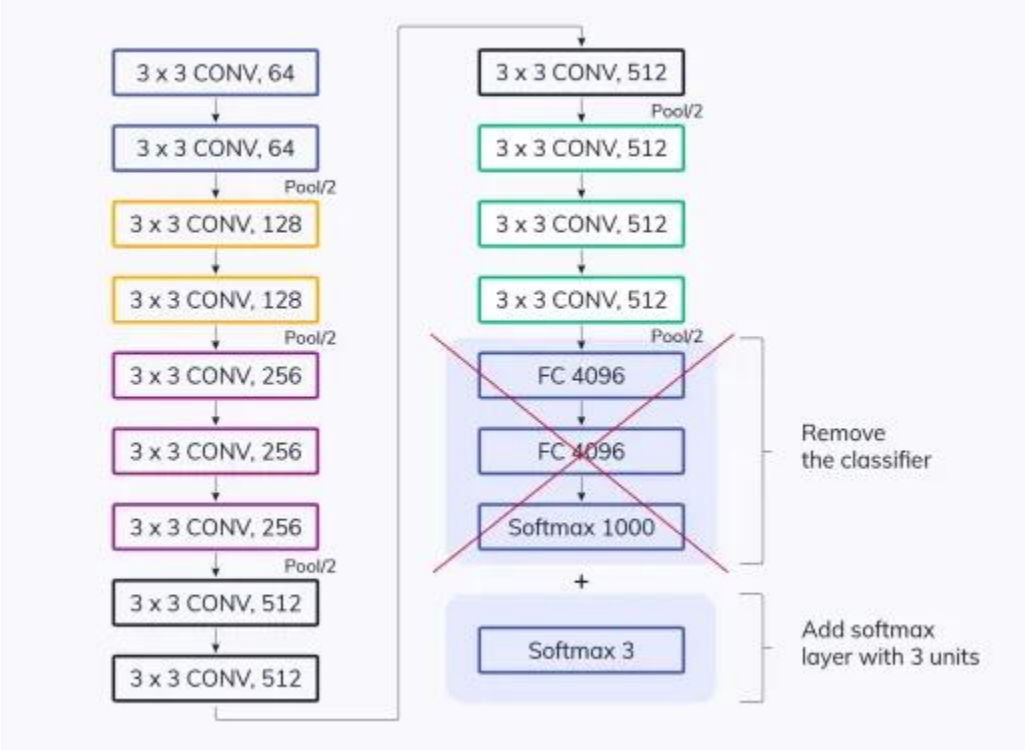


Fig 22: transfer in deep learning

3. **Freeze layers**

Freezing the starting layers from the pre-trained model is essential to avoid the additional work of making the model learn the basic features.

If we do not freeze the initial layers, we will lose all the learning that has already taken place. This will be no different from training the model from scratch and will be a loss of time, resources, etc.

4. **Add new trainable layers**
   The only knowledge we are reusing from the base model is the feature extraction layers. We need to add additional layers on top of them to predict the specialized tasks of the model. These are generally the final output layers.

5. **Train the new layers**
   The pre-trained model's final output will most likely differ from the output we want for our model. For example, pre-trained models trained on the ImageNet dataset will output 1000 classes.

   However, we need our model to work for two classes. In this case, we have to train the model with a new output layer in place.

6. **Fine-tune your model**
   One method of improving the performance is fine-tuning.

   Fine-tuning involves unfreezing some part of the base model and training the entire model again on the whole dataset at a very low learning rate. The low learning rate will increase the performance of the model on the new dataset while preventing overfitting.

Types of Deep Transfer Learning

- **Domain Adaptation**
  Domain adaptation is a transfer learning scenario where the source and target domains have different feature spaces and distributions. Domain adaptation is the process of adapting one or more source domains to transfer information to improve the performance of a target learner. This process

attempts to alter a source domain to bring the distribution of the source closer to that of the target.

- **Domain Confusion**
  In a neural network, different layers identify different complexity of features. In a perfect scenario, we'd develop an algorithm that makes this feature domain invariant and improves its transferability across domains.

  The feature representations between the source and target domains should be as similar as possible in such a context. The goal is to add an objective to the model at the source to encourage similarity with the target by confusing the source domain itself.

  Specifically, domain confusion loss is used to confuse the high-level classification layers of a neural network by matching the distributions of the target and source domains.

  In the end, we want to make sure samples come across as mutually indistinguishable to the classifier. To achieve this, one has to minimize the classification loss for the source samples, and one has to also minimize the domain confusion loss for all samples.

- **Multi-task Learning**

  In the case of multitask learning, several tasks from the same domain are learned simultaneously without distinction between the source and targets.

  We have a set of learning tasks, t1 , t2 , …, t(n) and we co-learn all tasks simultaneously.

  This helps to transfer knowledge from each scenario and develop a rich combined feature vector from all the varied scenarios of the same domain. The learner optimizes the learning/performance across all of the n tasks through some shared knowledge.

- **One-shot Learning**

One-shot learning is a classification task where we have one or a few examples to learn from and classify many new examples in the future.

This is the case of face recognition, where people's faces must be classified correctly with different facial expressions, lighting conditions, accessories, and hairstyles, and the model has one or a few template photos as input.

For one-shot learning, we need to fully rely on the knowledge transfer from the base model trained on a few examples we have for a class.

- **Zero-shot Learning**

If transfer learning is applied excessively using zero instances of a class and does not depend on labeled data samples, then the corresponding strategy is called Zero-shot learning.

Zero-shot learning needs additional data during the training phase to understand the unseen data. Zero-shot learning focuses on the traditional input variable, x, the traditional output variable, y, and the task-specific random variable.

Zero-shot learning comes in handy in scenarios such as machine translation, where we may not have labels in the target language.

Deep Transfer Learning applications:

- **NLP:**
  is one of the most attractive applications of transfer learning. Transfer learning uses the knowledge of pre-trained AI models that can understand linguistic structures to solve cross-domain tasks. Everyday NLP tasks like next word prediction, question-answering, machine translation use deep learning models like BERT, XLNet, Albert, TF Universal Model, etc.

- **Computer Vision:**
  Transfer learning is also applied in Image Processing.

Deep Neural Networks are used to solve image-related tasks as they can work well identifying complex features of the image. The dense layers contain the logic for detecting the image; thus, tuning the higher layers will not affect the base logic. Image Recognition, Object Detection, noise removal from images, etc., are typical application areas of Transfer learning because all image-related tasks require basic knowledge and pattern detection of familiar images.

- **Audio/Speech**
  Transfer learning algorithms are used to solve Audio/Speech related tasks like speech recognition or speech-to-text translation.

  When we say "Siri" or "Hey Google!", the primary AI model developed for English speech recognition is busy processing our commands at the backend.

  Interestingly, a pre-trained AI model developed for English speech recognition forms the basis for a French speech recognition model.

In most problems in medical field of computer vision such as skin cancer detection, the size of the data is not big enough (e.g., there are only thousands of images; however, CNN require much more than that), and a lot of time is required to train a CNN from the scratch.

Therefore, it is common to use a network that is pretrained on a very large dataset (i.e., ImageNet in 1.2 million images) and then use this knowledge as an initialization for the task of interest.

There are two most common ways to apply transfer learning as follows:

- **Fixed Feature Extractor**
  We can use the pre-trained model as a feature extraction mechanism. The way it works is by removing the output layer or the last fully connected layer and using the rest of the network as a fixed feature extractor for the dataset of our interest.
- **Fine-tuning**
  Fine-tuning is making some fine adjustments to increase performance further. For example, if we have one dataset, we can randomly separate it to the training and testing (validation) dataset with the ratio of our choice. Afterwards, we can train the model file with the training dataset and then train the same model with the testing dataset.

2.8 TensorFlow

TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind. However, it proved to be very useful for deep learning development as well, and therefore Google open-sourced it.
TensorFlow accepts data in the form of multi-dimensional arrays of higher dimensions called tensors. Multi-dimensional arrays are very handy in handling large amounts of data.
TensorFlow works on the basis of data flow graphs that have nodes and edges. As the execution mechanism is in the form of graphs, it is much easier to execute TensorFlow code in a distributed manner across a cluster of computers while using GPUs.

Benefits of TensorFlow:

- **TensorFlow Offers Python API's:**
  Before the development of libraries, the coding mechanism for machine learning and deep learning was much more complicated. This library provides a high-level API, and complex coding isn't needed to prepare a neural network, configure a neuron, or program a neuron. The library completes all of these tasks.

- **TensorFlow Supports Both CPUs and GPUs Computing Devices:**

  Deep learning applications are very complicated, with the training process requiring a lot of computation. It takes a long time because of the large data size, and it involves several iterative processes, mathematical calculations, matrix multiplications, and so on. If you perform these activities on a normal Central Processing Unit (CPU), typically it would take much longer.

Graphical Processing Units (GPUs) are popular in the context of games, where you need the screen and image to be of high resolution. GPUs were originally designed for this purpose. However, they are being used for developing deep learning applications as well.

One of the major advantages of TensorFlow is that it supports GPUs, as well as CPUs. It also has a faster compilation time than other deep learning libraries, like Keras and Torch.

- **TensorFlow competes with a slew of other machine learning** frameworks. PyTorch, CNTK, and MXNet are three major frameworks that address many of the same needs. Let's close with a quick look at where they stand out and come up short against TensorFlow:
  - **PyTorch** is built with Python and has many other similarities to TensorFlow: hardware-accelerated components under the hood, a highly interactive development model that allows for design-as-you-go work, and many useful components already included. PyTorch is generally a better choice for fast development of projects that need to be up and running in a short time, but TensorFlow wins out for larger projects and more complex workflows.
  - **CNTK**, the Microsoft Cognitive Toolkit, is like TensorFlow in using a graph structure to describe dataflow, but it focuses mostly on creating deep learning neural networks. CNTK handles many neural network jobs faster, and has a broader set of APIs (Python, C++, C#, Java). But it isn’t currently as easy to learn or deploy as TensorFlow. It's also only available under the GNU GPL 3.0 license, whereas TensorFlow is available under the more liberal Apache license. And CNTK isn't as aggressively developed.
  - **Apache MXNet**, adopted by Amazon as the premier deep learning framework on AWS, can scale almost linearly across multiple GPUs and multiple machines. MXNet also supports a broad range of language APIs—Python, C++, Scala, R, JavaScript, Julia, Perl, Go—although its native APIs aren’t as pleasant to work with as TensorFlow’s. It also has a far smaller community of users and developers.

TensorFlow allows you to create dataflow graphs that describe how data moves through a graph. The graph consists of nodes that represent a

mathematical operation. A connection or edge between nodes is a multidimensional data array. It takes inputs as a multi-dimensional array where you can construct a flowchart of operations that can be performed on these inputs.

Tensorflow architecture works in three significant steps:

- Data pre-processing - structure the data and brings it under one limiting value
- Building the model - build the model for the data
- Training and estimating the model - use the data to train the model and test it with unknown data

TensorFlow requirements can be classified into the development phase (training the model) and run phase (running the model on different platforms). The model can be trained and used on GPUs as well as CPUs. Once the model has been trained,

you can run it on:

- Desktop (Linux, Windows, macOS)
- Mobile devices (iOS and Android)
- Cloud as a web service

Components of TensorFlow:

- Tensor: forms the core framework of TensorFlow. All the computations in TensorFlow involve tensors. It is a matrix of n-dimensions that represents multiple types of data. A tensor can be the result of a computation, or it can originate from the input data.
- Graphs: describe all the operations that take place during the training. Each operation is called an op node and is connected to the other. The graph shows the op nodes and the connections between the nodes, but it does not display values

list of algorithms currently supported by TensorFlow:

- Classification - tf.estimator.LinearClassifier
- Linear regression - tf.estimator.LinearRegressor
- Boosted tree classification - tf.estimator.BoostedTreesClassifier
- Booster tree regression - tf.estimator.BoostedTreesRegressor
- Deep learning classification - tf.estimator.DNNClassifier
- Deep learning wipe and deep - tf.estimator.DNNLinearCombinedClassifier

TensorFlow programs work on two basic concepts:

- Building a computational graph
- Executing a computational graph

2.9 Image processing

- **Dermoscopy Image Preprocessing:**

The optical lenses of digital cameras reduce the quality of the digital images of skin lesions. This causes some difficulties in the diagnosis of malignancy by visual assessment due to the complexity of digital images. Therefore, there is a need for efficient image processing techniques to help physicians diagnose skin lesions accurately. Image pre-processing makes images suitable for this application by improving the quality of an image and for manipulating datasets by removing the noise and irregularities present in an image. In this study, the training set contained more than 13,000 skin lesion images of different resolutions. Because the resolution of all lesion images is greater than 299 x 299, it was necessary to extract the region of interest (skin lesion) and get rid of unnecessary/redundant regions from each image. Thus, we automatically cropped and processed these images before using in the image classification algorithm. This preprocessing step is necessary for; first, reducing the computation time by removing/reducing number of pixels to be processed; second, increasing performance of the classifier. Image pre-processing steps used in this study are segmentation, auto-cropping, and image resampling

- **Image Segmentation:**

Image segmentation is a process of dividing an image into multiple segments that are considerably/perceptually homogeneous in terms of preferred characteristics such as color, texture, etc. Image segmentation is typically used to identify objects, estimate the boundaries of an image,

remove unwanted regions on the image, compress and edit images or manipulate and visualize the data with a goal of providing a description or classification of the image. This process is widely used especially in medical image processing.

Segmentation process by first finding the superpixels. Superpixels are one of the most popular images over-segmentation algorithms. Among many superpixel algorithms, the choice of superpixel algorithm in this thesis is Simple Linear Iterative Clustering (SLIC). SLIC is categorized as a gradient ascent method, and it is often used as a baseline. SLIC implements a local K-means clustering to generate a superpixel segmentation with K-superpixels. More specifically, it groups similar pixel values and improves superpixel centers using K-means clustering algorithm. In our case, we use SLIC to automatically detect the region of interest (skin lesion) and use that to automatically crop the image without losing a part of a skin lesion. The original skin lesion images were divided to 400 superpixel areas (dynamically determining each super pixel size of 16 x 16), which are separated by blue lines. We empirically found that starting with 400 superpixel centers is optimal.

Example of superpixels:

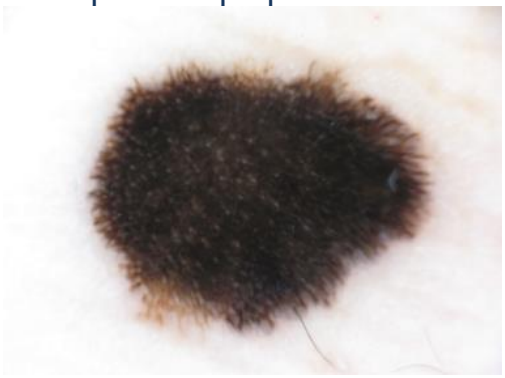
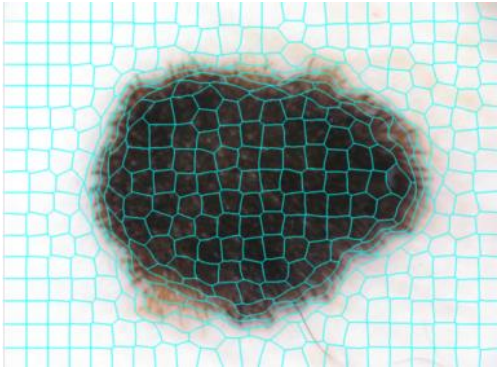


Fig 23: original image



subdivided to 400 superpixel areas

The next step is roughly extracting the entire the skin lesion. To do that, needed to find a way to merge superpixels that include some part of the lesion. For merging superpixels, we considered using some segmentation techniques including thresholding technique, edge detection technique, region extraction technique, and fuzzy-based image segmentation. To merge superpixels that fall partially or entirely in skin lesion region, we used

thresholding technique and merging superpixels. The next step is automatically cropping the image.

- **Cropping and Image Resampling:**

Image resampling is a technique used to manipulate the size of an image. Increasing the size of the image is called upsampling while decreasing the size is called downsampling. These two techniques are essential for applications like image display, compression, and progressive transmission. During downsampling or upsampling processes, a two-dimensional (2D) representation is kept the same while the spatial resolution is reduced or increased, respectively. On the other hand, cropping is a technique used to find the region of interest in an image by framing around and clipping the area.

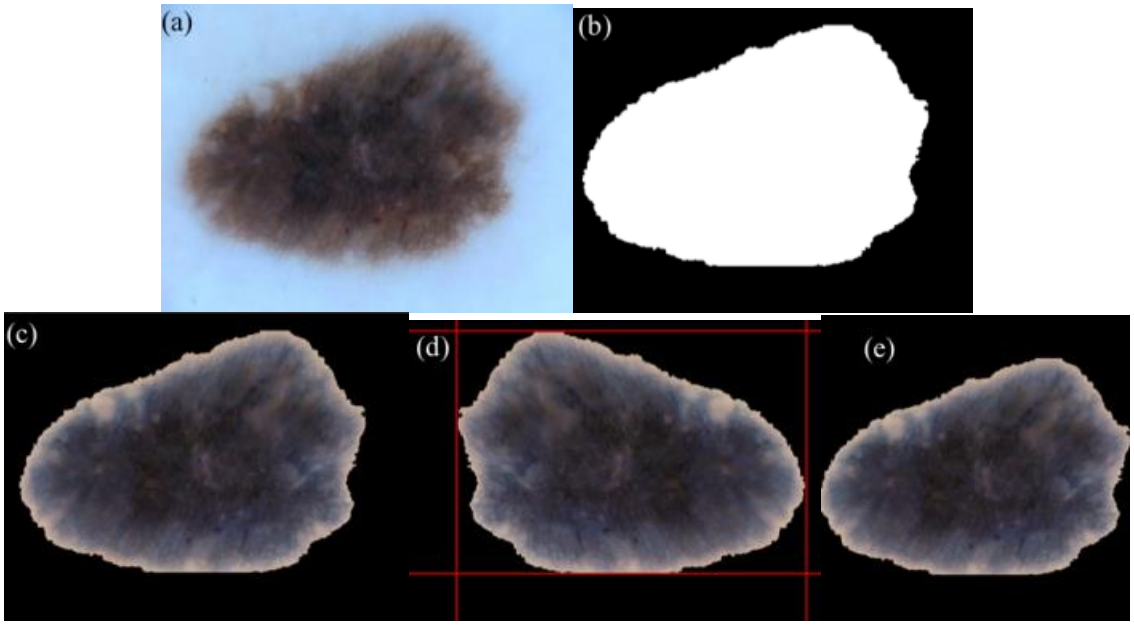


Fig 24: Images after each preprocessing steps: (a) is the original image, (b) is the segmentation mask of the image, (c) is the overlap of (a) and (b), (d) is framing the Region of Interest, and (e) is the cropped and resized to n x n image.

- **Image Resizing with Adding Zero-Padding**

The data obtained from the ISIC archive is not always available to directly feed the algorithm which requires structures, clean and meaningful data. In

order to overcome this problem, we resized all images from the archive to 256x256 without losing any feature. Pseudo-code for this process is as follows:

1. Identify which side of the image is short.
2. Find the difference between two sides.
3. Take half of the difference.
4. Do padding by putting number of zeros to short sides by adding half of the difference.
5. Resize the image to 256x256.

After the skin lesion is roughly extracted from the dermoscopy images, the next step is classifying these lesions.

- **Grayscale conversion:**

Grayscale image contains only brightness information. Each pixel value in grayscale image corresponds to an amount or quantity of light. The brightness graduation can be differentiated in grayscale image. Grayscale image measures only light intensity. 8-bit image will have brightness variation from 0 to 255 where '0' represents black and '255' represents white. In grayscale conversion color image is converted into grayscale image. Grayscale images are easier and more faster to process than colored images. All image processing technique are applied on grayscale image.
In our proposed system colored or RBG image is converted into grayscale image by using weighted sum method by using following equations:

$$\text{Grayscale intensity} = 0.299\ R + 0.587\ G + 0.114\ B$$

- **Color space conversion:**

The conversion operation from the input image (RGB) into YIQ color space

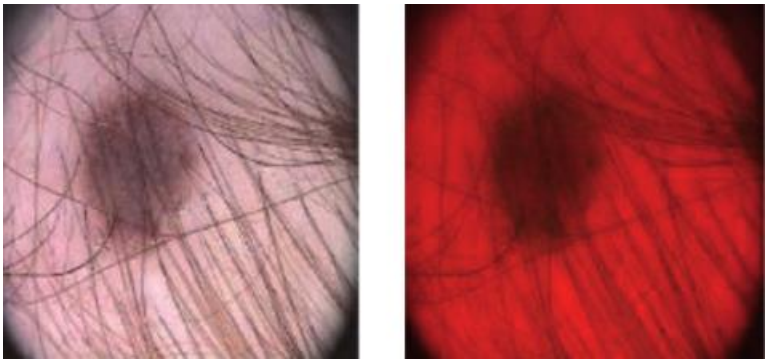


Fig 25: RGB and YIQ color space

- **Binary Image Conversion:**

  Binarization is the action of binarizing (make binary with 2 elements) data. From a practical point of view, an image with 2 colors (coded on 1 bit) is quick to store, each pixel is either 0 or 1.

- **Inpainting operation:**

  Divide the repaired Y-channel and the binarized image into 256 non-overlapped blocks. During experimental studies, several block sizes are tested such as 4×4, 8×8, 16×16, etc. We concluded that the implementation of block size 16×16 introduced better results for inpainting stage compared with other block sizes

- **Repaired RGB image:**

  The resulted image from the inpainting process as discussed earlier in the previous subsection is subsequently used as an input image to the conversion operation to the RGB color space as depicted in.

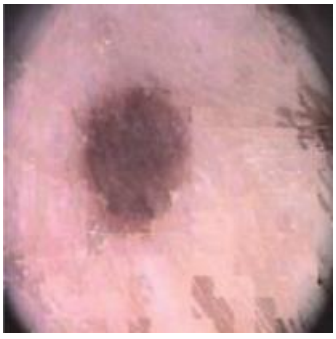


Fig 26: Repaired RGB image

- **Noise Removal:**

  The objective of noise removal is to detect and removed unwanted noise from digital image. The difficulty is in deciding which features of an image are real and which are caused by noise. Noise is random variations in pixel values. In our proposed system we are using median filter to remove unwanted noise. Median filter is nonlinear filter, it leaves edges invariant. Median filter is implemented by sliding window of odd length . Each sample value is sorted by magnitude, the centremost value is median of sample within the window, is a filter output.

- **Contrast Enhancement techniques:**

  Contrast refers to the amount of differentiation that is there between the various image features. Images having a higher contrast level generally display a greater degree of color or gray-scale variation than those of lower contrast.

  Contrast Enhancement refers to the sharpening of image features to remove the noisy feature such as edges and contrast boundaries. Contrast Enhancement Algorithms aim to improve the perception of the image by human eye.

  - Histogram Equalization: is an image processing technique that adjusts image intensities to improve contrast. Histogram Equalization is one of the simplest and commonly used method in low level image enhancement using the histogram. The logic behind Histogram Equalization is that the image with the best visual appearance, is the one whose histogram looks like the regular distribution.
    A Cumulative Distribution Function (CDF) of a histogram is the fraction of pixels in with a pixel value is less than or equal to the specified pixel value.

    Histogram Equalization is particularly useful in cases where both backgrounds and foregrounds are both bright and dark.
    Histogram Equalization can lead to better views of bone structure in x-ray images, and to better detail in over-exposed or under-exposed

photos. A quintessential advantage of Histogram Equalization method is that it is a fairly straight forward image processing technique.

OpenCV loads color images in BGR (Blue Green Red) color space. In BGR, it is not possible to perform histogram equalization without affecting the color information as all channels contain color information, therefore we have to convert the BGR image into YCrCb. In YCrCb (Luminance; Chroma: Blue; Chroma: Red) color space, the Y channel of the image only contains intensity information where as Cr and Cb channels contain all the color information of the image. The operation must be performed only on the Y channel to get the Histogram Equalized output without changing any color related details.

- Adaptive Histogram Equalization: Adaptive Histogram Equalization computes many histograms for each of the separate part of the image and uses them to redistribute the lightness values of the image, hence it differs from Histogram Equalization. Hence it is suitable for bettering the local contrast in images.

- CLAHE: Contrast Limited Adaptive Histogram Equalization (CLAHE) is a variant of Adaptive Histogram Equalization. CLAHE has one additional step over Adaptive Histogram Equalization and that is clipping of the histogram. The 5 steps in CLAHE:
    - Divide the image into tiny regions.
    - Decide the mapping functions of local histogram.
    - Choose the clipping point of histogram.
    - Apply the function to every region.
    - Reduce the noise by the background subtraction method.

- Contrast Stretching: In Contrast Stretching the contrast in an image is stretched from the range of intensity values it contains to span a desired range of values. It is also called Normalization.
  Min-Max Contrast Stretching:
  In Min-Max Contrast Stretching for each pixel:
  pixel = ((pixel – min) / (max – min)) * 255
  Where min and max are the maximum and minimum pixel values in the image.

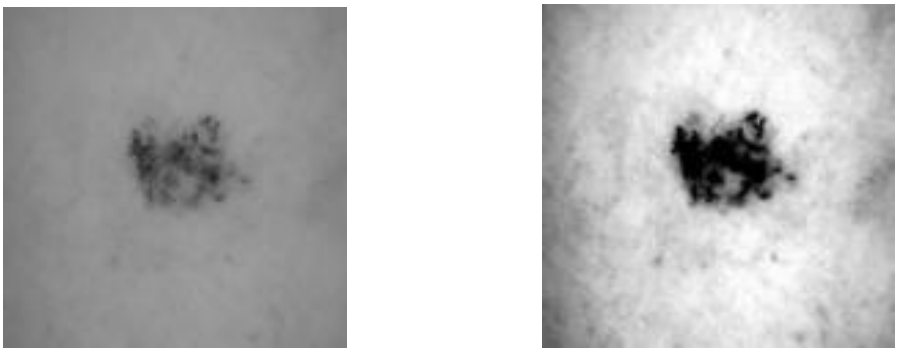


Fig 27: Contrast Enhancement (before and after)

- **Vignette Removal**

The vignetting effect refers to a position-dependent loss of brightness in the output of an optical system. This effect is seen as the darkening of the outer portions of the images (especially the corners) in some dermoscopy images. The vignetting is detected in the red plane (because it is least affected by the lesion) and normalized for each of the other two planes. Three steps are used to address vignetting: first, define a certain set of concentric circular regions with the width of the circle radius based on a tunable parameter; second, start the procedure with the image center; and third, adjust the brightness of the next circular region so that the average intensity is the same as the center and continue this for every region.

In stage of image acquisition, the vignetting can be reduced to a certain extent by removing additional filters, setting longer focal length or smaller aperture. Of course, these actions do not correct all types of vignetting and are not always possible to do. Therefore, a computational method for vignetting correction is used during preprocessing of acquired image. Most of these methods of vignetting correction require to estimate a mathematical model of vignetting. We can divide modelling methods into two groups: physically based models and approximation of vignetting functions.

Pseudocode for vignette removal algorithm

For up to 10 iterations
    Create a circular mask with

radius = half the height of the input image – 20 pixels + iteration*5 pixels
Calculate the mean of the outer circle
If the mean pixel value of the outer ring < 6.0 then
Fill the outer ring with the mean pixel value of the center region Return the corrected image

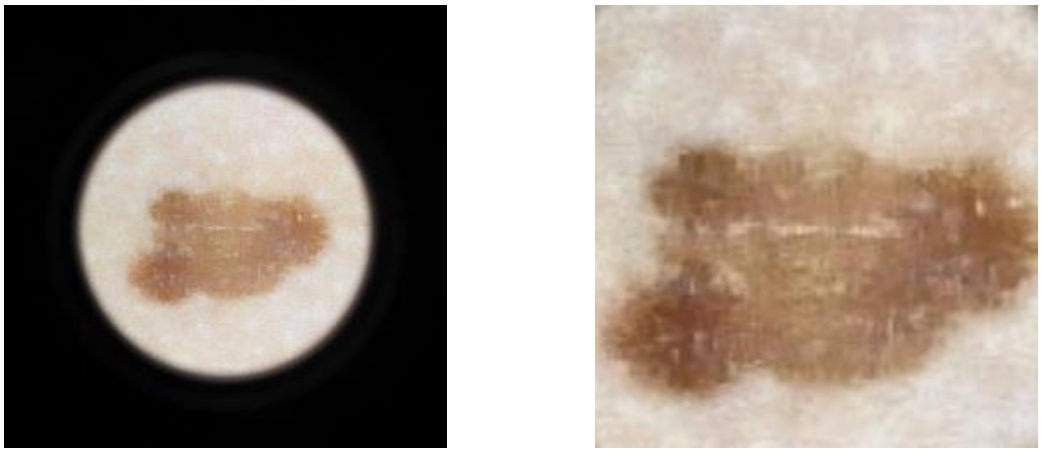


Fig 28: vignette correction (before and after)

- **Hair Removal:**

Body hair is one of the factors that can affect lesion segmentation. For the detection and removal of hairs, a pre-processing technique called DullRazor is used, which consists of applying a series of morphological operations to the image in order to generate a mask that contains the hairs. The steps to apply the DullRazor algorithm are:

1. Convert the original image to grayscale.
2. Closing to the grayscale image, using a linear or cross-shaped kernel.
3. Calculate the difference between the resulting image and the original.
4. Apply binary thresholding to obtain a mask with the hairs in the image.
5. Replace the pixels in common between the mask and the original image, with pixels from the latter.

For steps 2 and 3 of the DullRazor algorithm, the advanced morphological operation blackhat from the OpenCV library was used. On the other hand, the cv2.inpaint command from the same library was used in the last step of the algorithm.

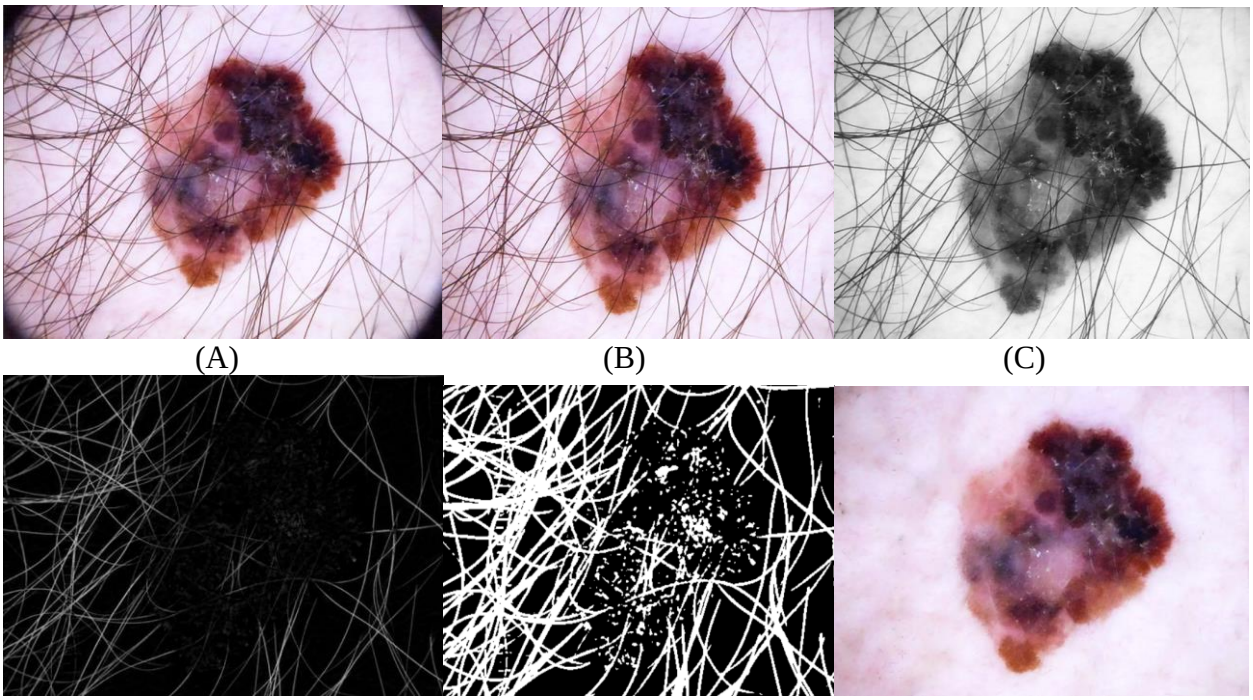


Fig 29: (A) Input image, (B) Cropped image, (C) Gray scale image, (D) Black hat filter, (E) Binary Thresholding (Mask), (F) Replace pixels of the mask (Output image)

Advantages of our technique of hair removal:

1. It identifies the dark hair locations by a generalized grayscale morphological closing operation
2. It verifies the shape of the hair pixels as thin and long structure and replaces the verified pixels by a bilinear interpolation
3. It smoothes the replaced hair pixels with an adaptive median filter.

2.10 Classification

Classification can be defined as making groups of things based on shared features, characteristics or other predefined properties. During the training phase, feature values of images are extracted from dermoscopy images of skin lesions for estimating the classes of images in test sets. There is one label for each image. There are several most popular algorithms to solve

classification problems such as decision trees, neural networks (NN), and support vector machines (SVM). However, these algorithms require manual feature extraction and preprocessing, and they can only calculate numeric values. In order to pass these cumbersome steps and make the algorithm do the feature extraction itself, we used transfer learning algorithm Inception V3 that is a specialized version of convolutional neural network. By transferring previously gained knowledge, we drastically reduced overall computation time without sacrificing efficiency

There are other several convolutional neural network architectures such as AlexNet, VGGNet, and ResNet. Because training these architectures requires big data and is a computationally intensive process, excessive computation power (GPU power) is needed. Also, since we use a supervised technique, training data sets need to have verified labels. In order to label these large datasets, crowd sourcing or community driven labelling methods should be used.

2.11 Web Application

- **Definition**
  A web application "web app" is an application software that runs on a web browser. Web apps are delivered on the world wide web to users with an active network connection.

- **Advantages**
  Web applications are better than mobile software as you can use the same responsive web design on any platform such as (windows, macOS, android, ios ,…) literally every device that connected to the internet can use the web application through a web browser such as google chrome , mozilla firefox, or safari.

  Web applications allow multiple users around the world access to the same version of the application without any need to update it manually or to install any additional application on your local device.

- **How a web application works**
  Web applications are usually coded in browser-supported languages these languages rely on the browser to render the program executable. Some of

these applications are dynamic, requiring server-side processing. Others are completely static with no processing required at the server.

The web application requires a web server to manage requests from client , an application server to perform the tasks requested , and sometimes a database to store information .

Web applications typically have short development cycles. Most web applications are written in javascript , Html5 and Css3 .client-side programming typically utilizes these languages ,which help build an application front-end. Server-side programming is done to create scripts a web application will use languages such as python , java and Ruby these are commonly used in server-side programming

2.12 Related work

A Deep Learning Approach to Universal skin Disease Classification:

Diagnosis of skin diseases sometimes requires a high-level of expertise due to the variety of their visual aspects. As human judgment is often subjective and hardly reproducible, to achieve a more objective and reliable diagnosis, a computer aided diagnostic system should be considered.

Method Used:

CNN methods used to classify skin lesions are presented. CNNs can be used to classify skin lesions in two fundamentally different ways. On the one hand, a CNN pretrained on another large dataset, such as ImageNet, can be applied as a feature extractor. In this case, classification is performed by another classifier, such as k-nearest neighbors, support vector machines, or artificial neural networks. On the other hand, a CNN can directly learn the relationship between the raw pixel data and the class labels through end-to-end learning. In contrast with the classical workflow typically applied in machine learning, feature extraction becomes an integral part of classification and is no longer considered as a separate, independent processing step. If the CNN

is trained by end-to-end learning, the research can be additionally divided into two different approaches: learning the model from scratch or transfer learning.

Accuracy: 70%

Other Method:

CNNs can be scaled to achieve better Accuracy. However, the caling process was never thoroughly investigated. It entailed an iterative manual tuning process, either by arbitrarily increasing the depth or width of the CNN or by using a higher input image resolution. The **EfficientNet** family of architectures was developed by with an intent to find a suitable method to scale CNNs to achieve better Accuracy (i.e., model performance) and efficiency (i.e., model parameters and FLOPS). The authors propose a compound scaling method that uses a fixed set of coefficients to uniformly scale width, depth, and resolution. That method allowed authors to produce efficient CNN architecture, which they named **EfficientNet B7**.

Accuracy :80%

Other Method:

GoogleNet: Increasing the network size or its depth is the simplest way to improve deep neural network performance where the depth refers to the number (levels) of network layers.

Successful training of deeper models required a large number of labeleddata. There are two drawbacks in this way. First, this process involves the evaluation of a large number of parameters. Such parameters may lead to architecture overftting, especially when a limited labeled dataset is used for training. The second drawback is concerned with the computational cost, where this cost increased when using a network with many hidden layers.

the GoogleNet's first inception module as an example with 192 channels as input. It has 128 and 32 filters for each group with size 3 x 3 and 5 x 5, respectively. Thus, for the 5 x 5 filter, the computation order is 25 x 32 x 192. It is expandable when going deeper by increasing the width of both the network and the filter. A 1 x 1 convolution is used with only 16 filters to minimizethe dimensions of the input channel. The computations will reduce from 25 x 32 x 192 to 16 x 192 + 25 x 32 x 16. All of these changes enable a large width and depth to thenetwork.

The fully-connected layers are replaced with a simple pooling global average layer. This change in GoogleNet reduces the total number of parameters significantly

Accuracy : 90%

Chapter 3
Analysis and Design

3.1 System Overview

Skin diseases detection helps in identifying the affected cells by the skin illness that is caused by bacteria or even infection from other as it is found in animals, humans, and plants

It is important to identify these diseases early as it has various dangerous effects on the skin and keep on spreading over time it is become important to identify these diseases at their initial stage to control it from spreading

- **Web Application Flow**
    1. User triggers a request to the web server over the Internet, either through a web browser or the application's user interface.
    2. Web server forwards this request to the appropriate web application server
    3. Web application server performs the requested task – such as querying the database or processing the data – then generates the results of the requested data.
    4. Web application server sends results to the web server with the requested information or processed data.
    5. Web server responds back to the client with the requested information that then appears on the user's display

- **The steps of developing our web app (development flow)**

  We have used Html5, Css3, JavaScript for the front-end part and we have used responsive design techniques to make it work on difference screen sizes such as the laptop screens and the mobile screens as well.

  we have used python flask framework for the machine learning model deployment and for the back-end part. Flask is a micro web framework written in python.

  The flow goes as shown in the next figure

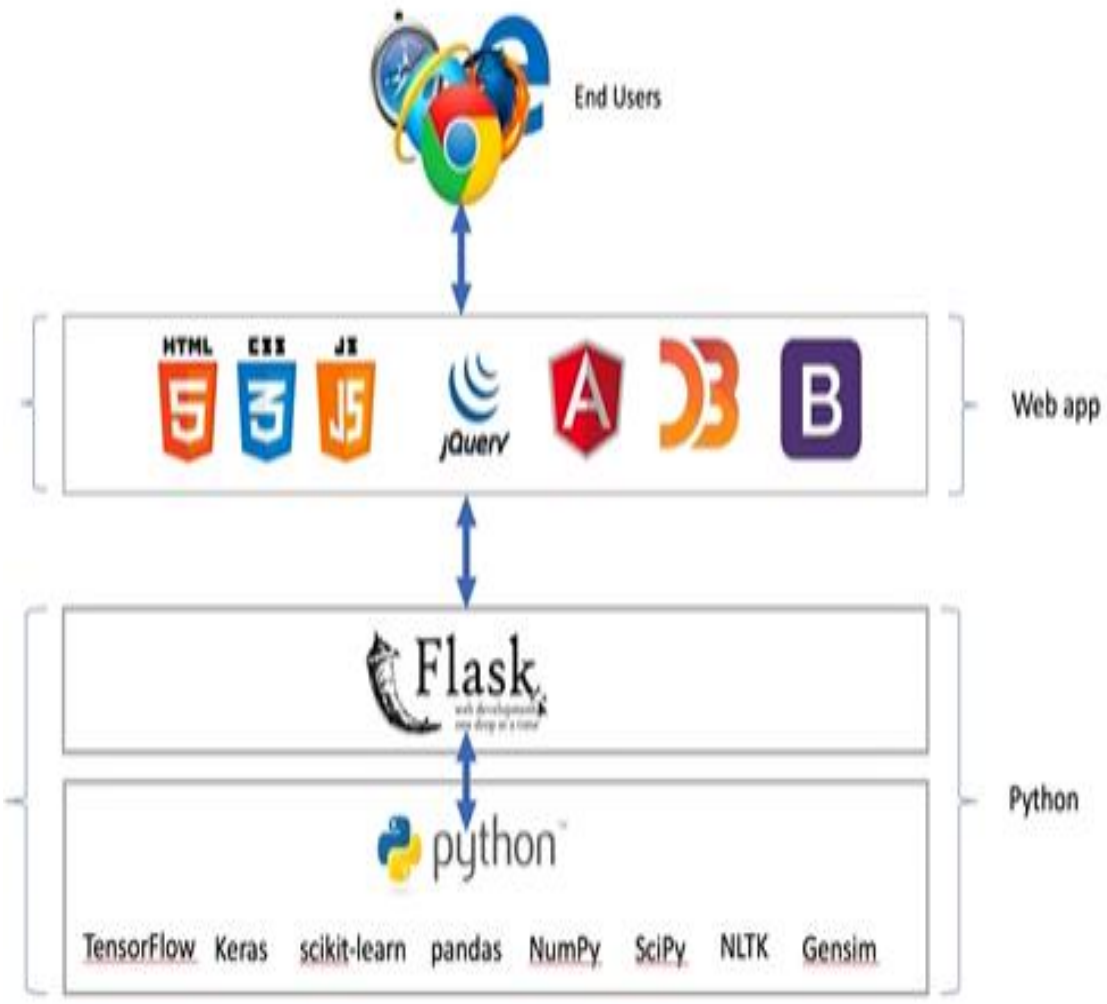


Fig 30: development web app flow

3.1.1 System Architecture

System contains three modules: proprocess on the input image , Train and Validate to extract model , classify the image to the appropriate skin diseases , and the figure below represent these modules as three ties architecture

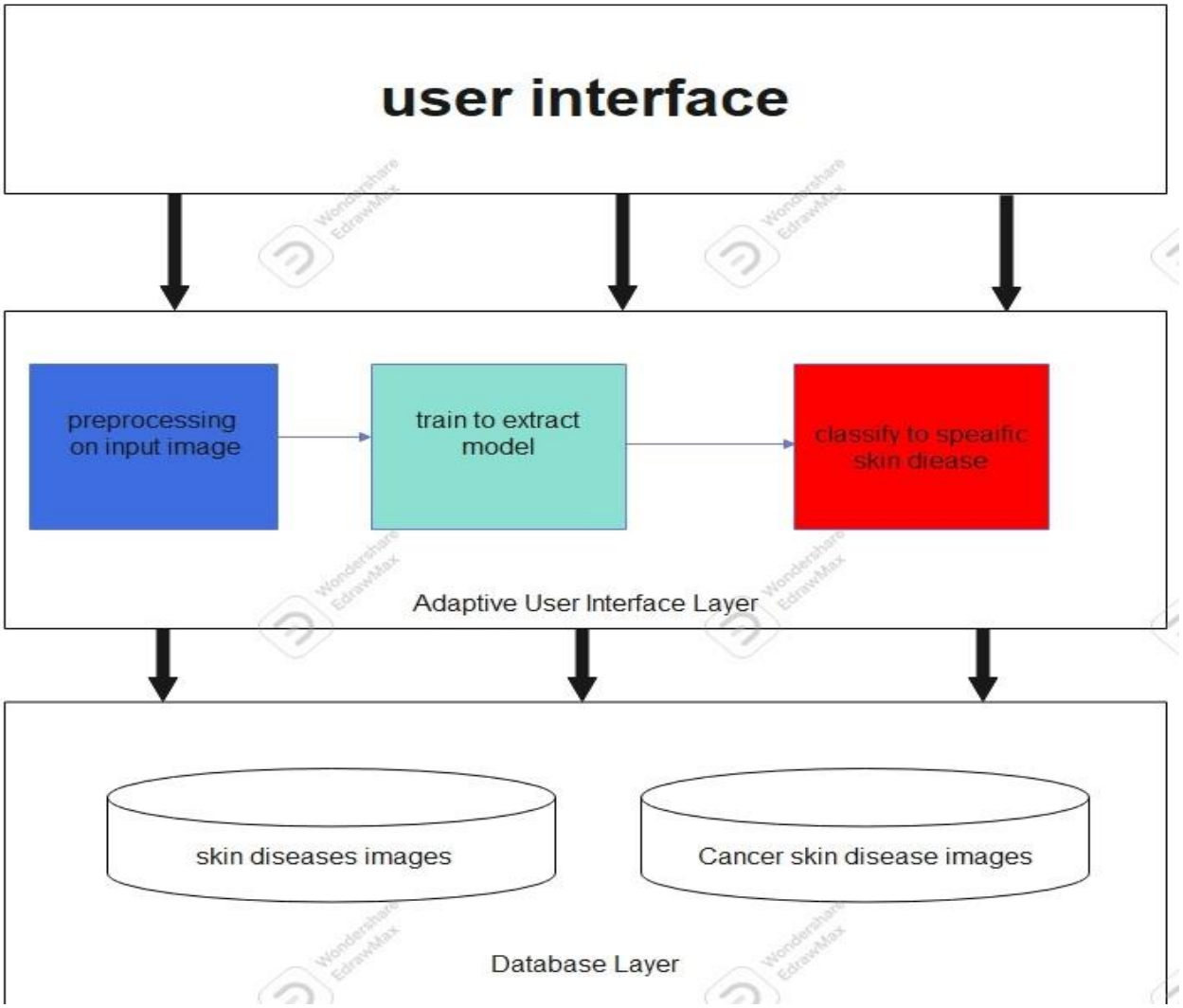


Fig 31: System Architecture

Main Modules

1. Preprocessing on image: We make some preprocessing steps before using image to increase the probability of being correctly classified
2. Train dataset: we divide our data into 80% for Training and 20% for Validating and Testing, Using CNN, and Mobile Net to generate the model
3. Classification: We classify the image to get top 3 accuracy diseases

3.1.2 Functional Requirements

Input Image:

- Description: The system upload image
- Input: affected skin image
- Source: User
- Output: confirm by showing the input image in the application

Choose Model to predict

- Description: The system offers input photos in models for prediction (Cancer skin- General skin)
- Input: Selected model for prediction.
- Source: User.
- Output: Confirm the user selection.

Predict& Display Classification Result

- Description: The system classification the input image and get three top accuracies to display for user as the result of classification
- Input: Uploaded image.
- Source: system
- Output: Show classification results with probability for each displayed disease

3.1.3 Nonfunctional Requirements

Performance:

The system must be interactive, and the delays involved must be less. In every action-response of the system, there are no immediate delay. In case of opening forms, popping error messages there is a delay much below 1.5 seconds.
In case of opening database, there is 3 seconds to get the history of classification. In case of classifying new images is 0.25 seconds delay.

Reliability:

As system reliable enough to sustain in any condition. should give consistently correct results.

Maintainability:

It is easy to add code to existing system, and upgrade for new features and new technologies from time to time. it is cost-effective and easy.

Safety Requirements:

Information transmission should be securely transmitted to server without any changes.

Security Requirements:

The system shall provide a certain type of security to make sure of the validity of data of the user

Usability:

As the system is friendly easy to handle and navigates in the most expected way with no delays. in that case the system program reacts accordingly and transverses quickly between its states

3.1.4 System Users

A. Intended Users:

The application will be used by patients who want to identify the disease to know whether it is an intimate or malignant disease.

Also, the application will used by the doctors, so it helps them to classify the affected skin of the patients and get the tops accuracy to be classified correctly so it helps them with time reduction of the diagnosis process.

B. User Characteristics

None defined images capture images and record data and symptoms, and the application will classify images and output the classification. result

3.2 System Analysis & Design

3.2.1 Use Case Diagram

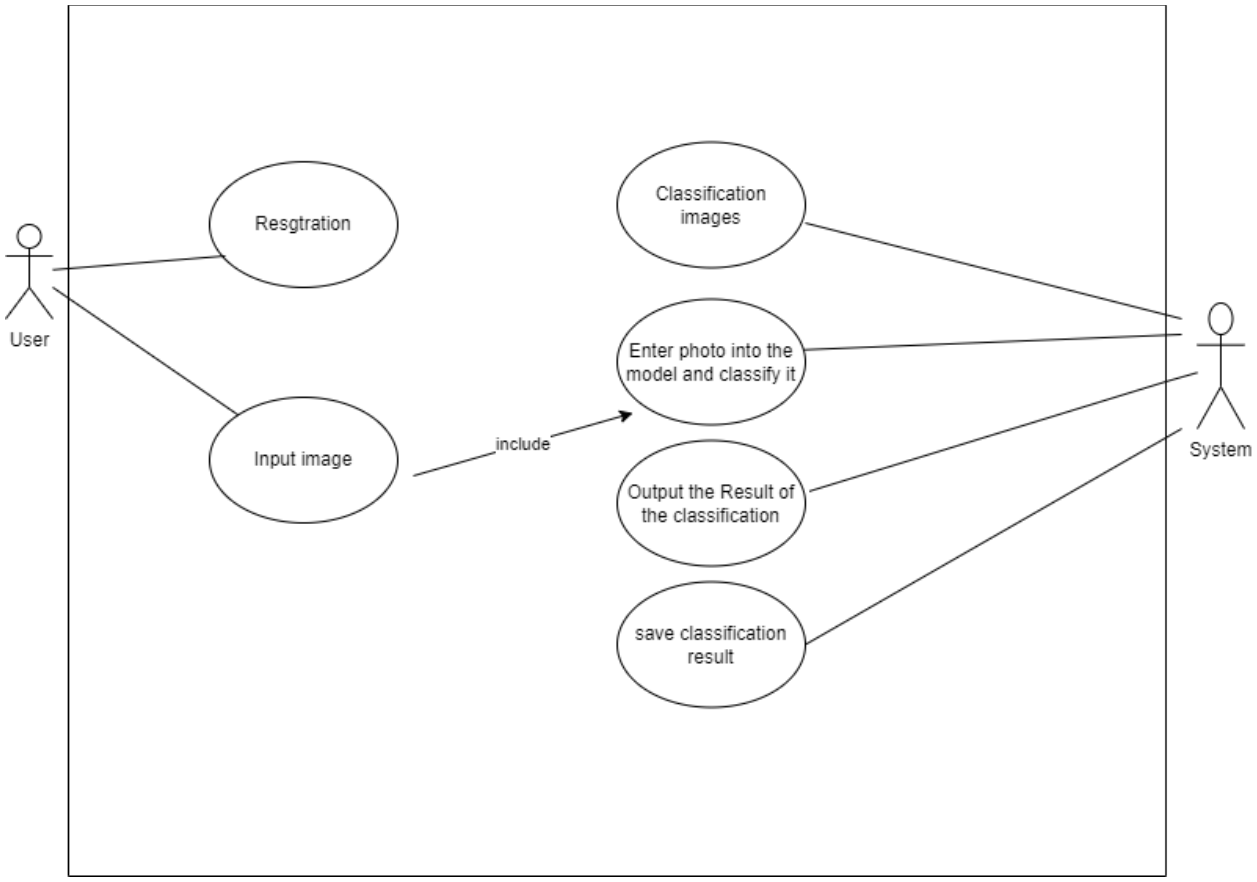


Fig 32: Use Case Diagram

3.2.2 Class Diagram

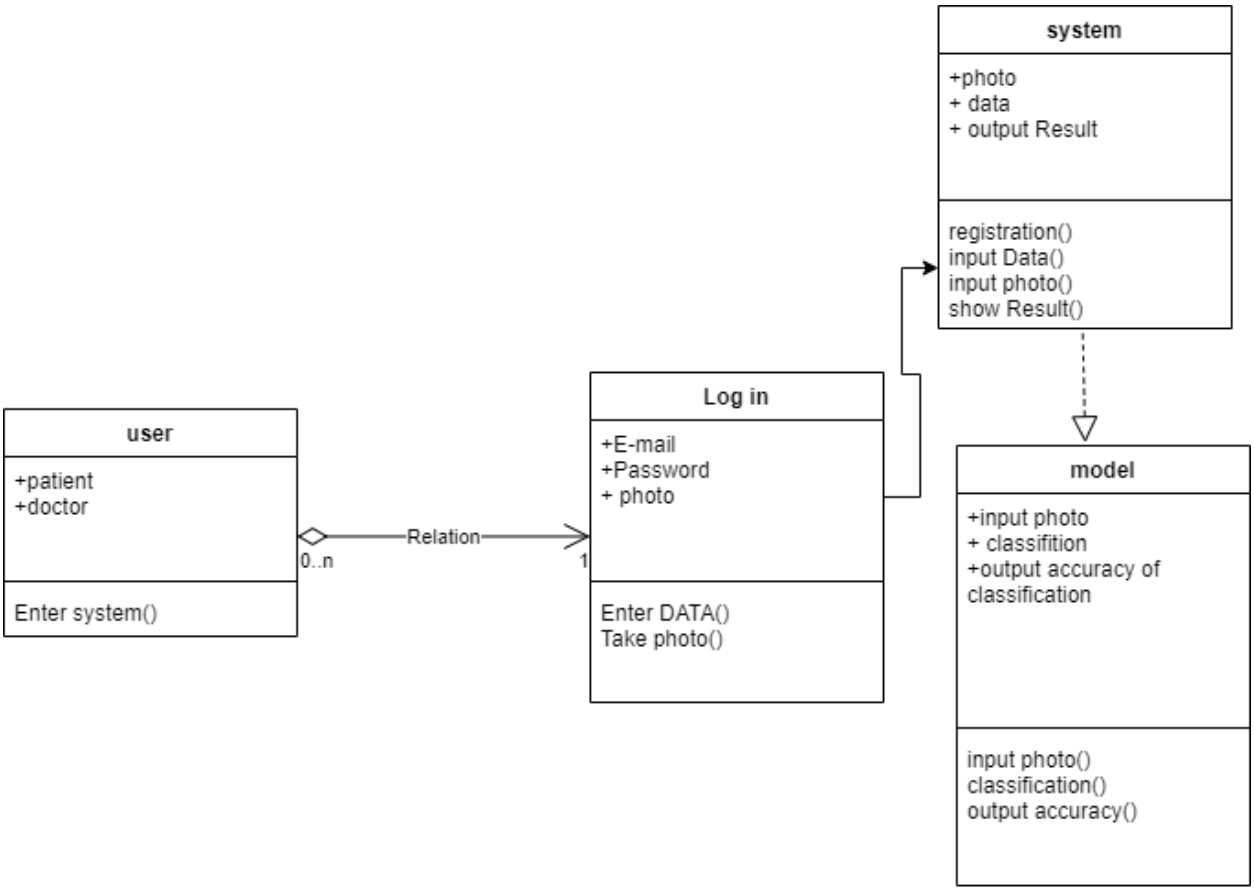


Fig 33: Class Diagram

User:

- Description: patient or doctor
- Flow of the event: enter the system

System

- Description: The user enters the system to record the data
- Flow of the event: it makes a registration for the data, enters the data and sees the result

Model

- Description: take photo, make classification, and output the accuracy than show the Result
- Flow of the event: model take photo make classification and compare it with other photo and show the result

3.2.3 Sequence Diagrams

Log-in Sequence Diagrams:

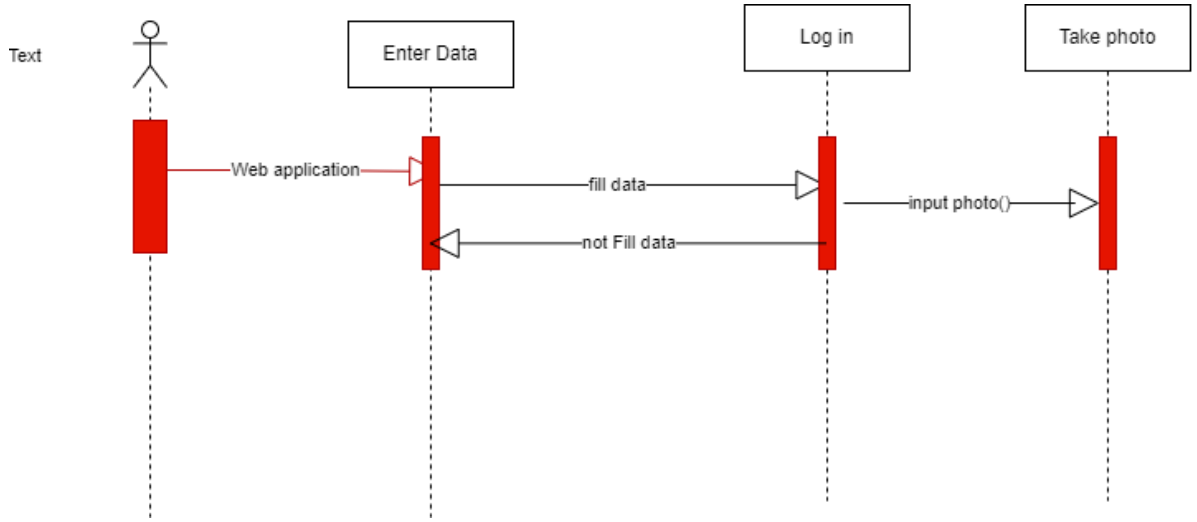


Fig 34: Login Sequence Diagrams

Input Image

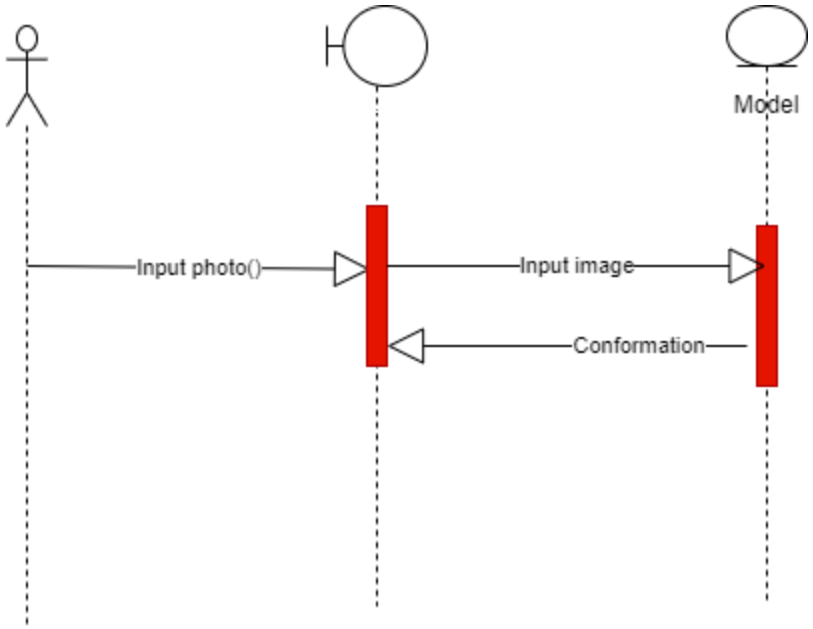


Fig 35: Input Image

Classification sequence Diagram

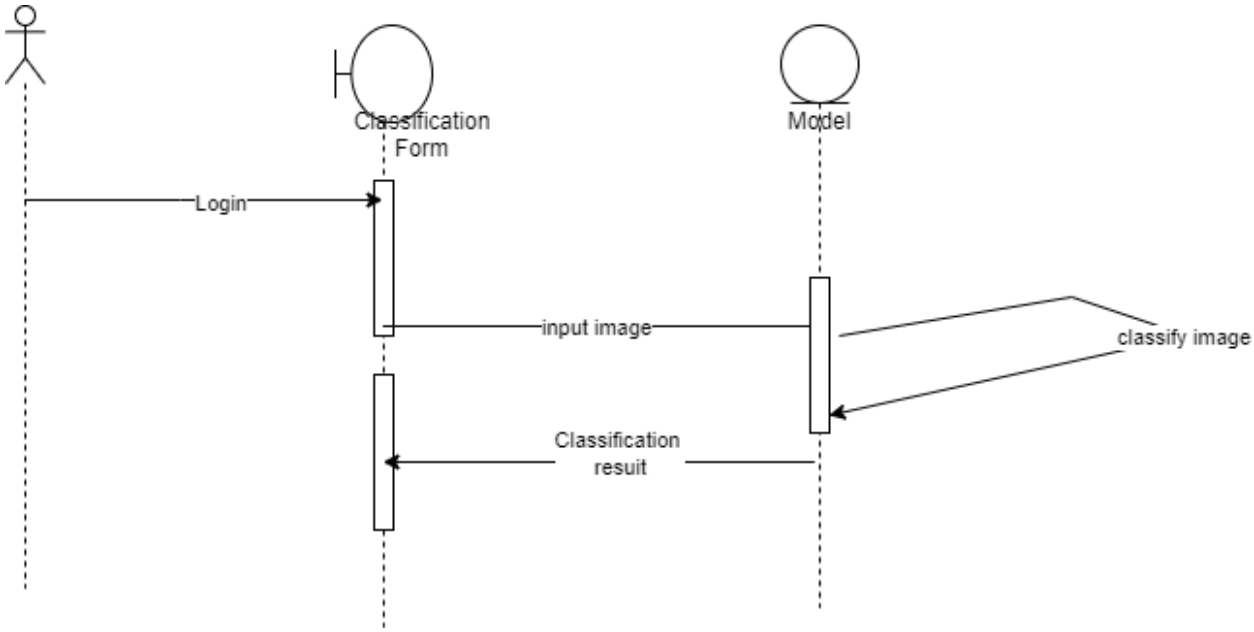


Fig 36: Classification sequence Diagram

System Sequence Diagrams

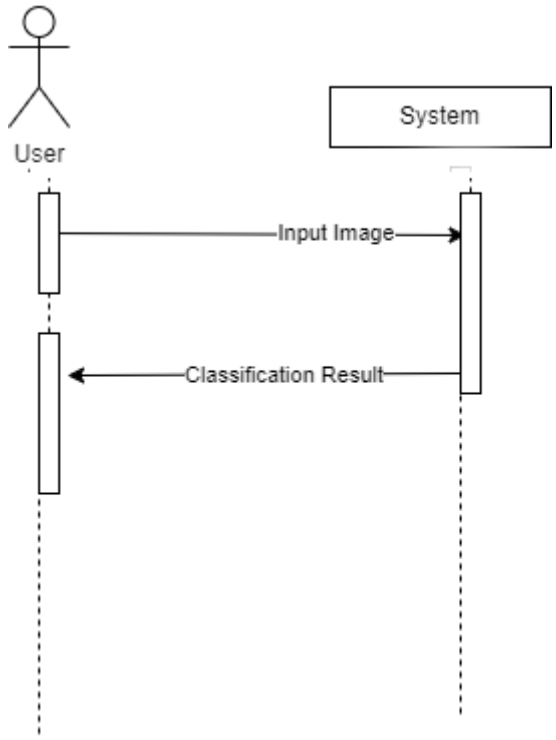


Fig 37: System Sequence Diagrams

3.2.4 Database Diagram

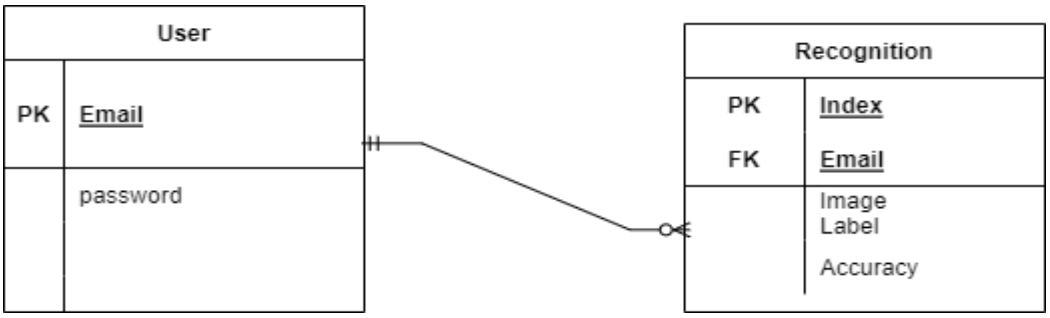


Fig 38: Database Diagram

Recognition:

This table contains the main attributes of each recognition/classification done by the user as the user can have more than one recognition.

Chapter 4
Implementation

4.1 Data Preprocessing

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

4.1.1 Replace missing value

In a real world dataset, there will always be some data missing. This mainly associates with how the data was collected. Missing data plays an important role creating a predictive model, because there are algorithms which does not perform very well with missing dataset.

We replace the missing values in the dataset with the simple Imputer Simple Imputer is a scikit-learn class which is helpful in handling the missing data in the predictive model dataset. It replaces the Nan values with a specified placeholder. using the mean and mode of the data as replacing the missing values in the age column with the mean of the column and replace the missing values in the sex, anatom_site_general with the mode of each column.

4.1.2 Data Normalization

Normalization is used to scale the data of an attribute so that it falls in a smaller range, such as -1.0 to 1.0 or 0.0 to 1.0. It is generally useful for classification algorithms.

Normalization is generally required when we are dealing with attributes on a different scale, otherwise, it may lead to a dilution in effectiveness of an important equally important attribute (on lower scale) because of other attribute having values on larger scale.

In simple words, when multiple attributes are there but attributes have values on different scales, this may lead to poor data models while performing data mining operations. So they are normalized to bring all the attributes on the same scale.

4.1.3 Data Augmentation

Data augmentation is the process of increasing the amount and diversity of data. We do not collect new data, rather we transform the already present data.

Data augmentation is an integral process in deep learning, as in deep learning we need large amounts of data and in some cases it is not feasible to collect thousands or millions of images, so data augmentation comes to the rescue.

It helps us to increase the size of the dataset and introduce variability in the dataset.

The most commonly used operations are: Rotation, Shearing, Zooming. Cropping, Flipping and Changing the brightness level.
We use the data augmentation to balance the dataset for no bias for a specific lesion and enlarge it for a better accuracy in training the model.

4.1.4 Generate training and validation data

The Training Set:
It is the set of data that is used to train and make the model learn the hidden features/patterns in the data.

In each epoch, the same training data is fed to the neural network architecture repeatedly, and the model continues to learn the features of the data.
The training set should have a diversified set of inputs so that the model is trained in all scenarios and can predict any unseen data sample that may appear in the future.

The Validation Set:

The validation set is a set of data, separate from the training set, that is used to validate our model performance during training.
This validation process gives information that helps us tune the model's hyperparameters and configurations accordingly. It is like a critic telling us whether the training is moving in the right direction or not.
The model is trained on the training set, and, simultaneously, the model evaluation is performed on the validation set after every epoch.
The main idea of splitting the dataset into a validation set is to prevent our model from overfitting i.e., the model becomes really good at classifying the samples in the training set but cannot generalize and make accurate classifications on the data it has not seen before.

The Test Set:

The test set is a separate set of data used to test the model after completing the training.
It provides an unbiased final model performance metric in terms of accuracy, precision, etc. To put it simply, it answers the question of "How well does the model perform?"

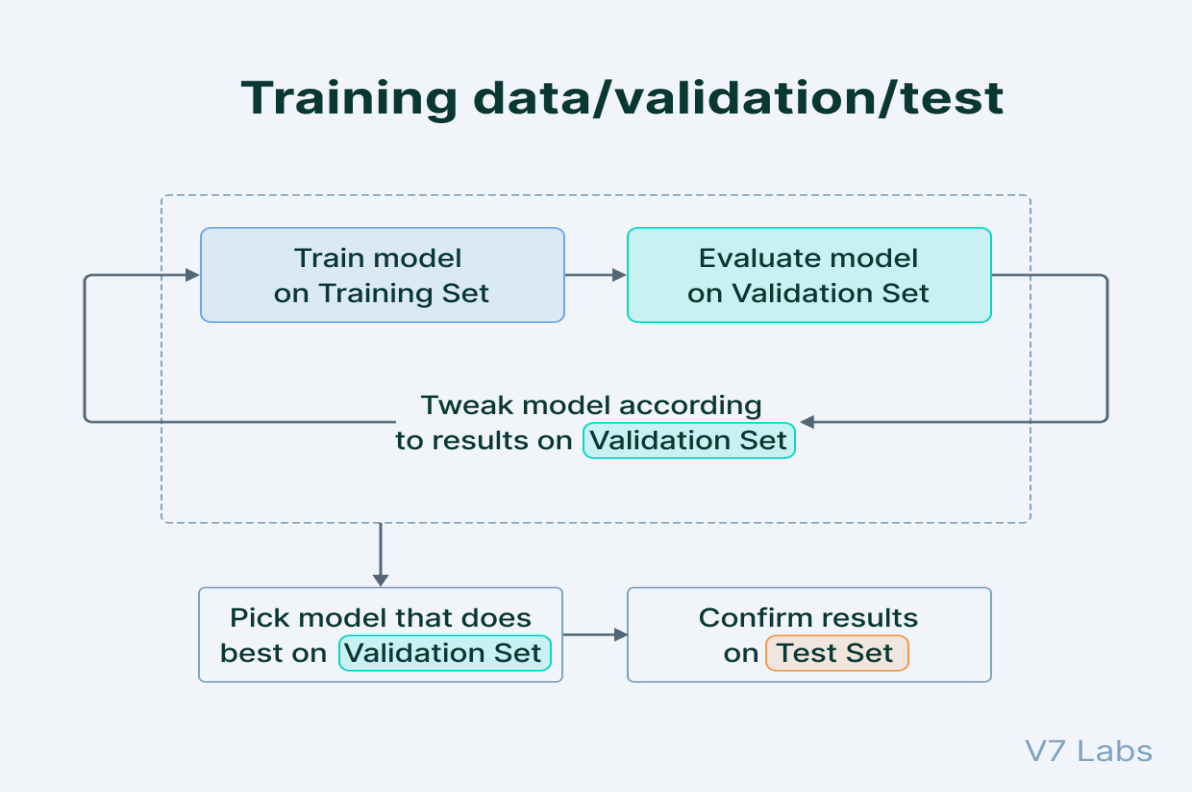


Fig 39: Data train, test, validate split

4.2 Image Preprocessing

Many times, while working on computer vision problems, we encounter situations where we need to apply some form of transformation to our entire dataset. However, it becomes difficult to apply custom transformations that are not available in Keras.

The ImageDataGenerator class in Keras provides a variety of transformations such as flipping, normalizing, etc.

We use some image preprocessing techniques to remove some noise, enhance images, highlight the details and remove unnecessary features. All of these techniques lead to improve the accuracy.

Pre-processing the input image data to convert it into meaningful floating-point tensors for feeding into Convolutional Neural Networks. Just for the knowledge

tensors are used to store data, they can be assumed as multidimensional arrays. A tensor representing a 100 X 125 image having 3 channels will have its dimensions (100, 125, 3).
It may seem a bit fussy, but Keras has utilities to take over this whole algorithm and do the heavy lifting for you. Keras has a module with image-processing helping tools, located at keras. Preprocessing image. It contains the class ImageDataGenerator, which lets you quickly set up Python generators that can automatically turn image files on disk into batches of preprocessed tensors.

4.3 Model Implementation

We implemented on Kaggle notebook with python programming language. It provides jupyter notebook service that requires no setup to use while providing free access to computing resources including GPU.
We created the model architecture using GoogleNet model from tensorflow. keras. Tensorflow is a python friendly open-source library for numerical computation that makes machine learning faster and easier.
Keras is one of the leading high level neural network API it is written in python and supports multiple backend neural network computation engine. Keras is a high-level machine learning framework build on top of TensorFlow

Transfer Learning: is a machine learning technique, which means a model is used with another task related to what it has trained for the first time. Domain adaptation and transfer learning refers to the condition in which the training of one set is used to improve generalization in another setting. Due to the enormous or massive resources required for deep

learning and the extensive number of images, transfer learning is a well-known approach in deep networks.
In addition to the limited numbers of available skin lesions datasets, these datasets are not suitable to train a deep network from the beginning, due to their small number of images.
Transfer learning was the best solution to overcome this Problem

4.3.1 Reduce learning rate

The optimization problem addressed by stochastic gradient descent for neural networks is challenging and the space of solutions (sets of weights) may be comprised of many good solutions (called global optima) as well as easy to find, but low in skill solutions (called local optima). the amount of change to the model during each step of this search process, or the step size, is called the “learning rate” and provides perhaps the most important hyperparameter to tune for your neural network in order to achieve good performance on your problem.

Learning rate controls how quickly or slowly a neural network model learns a problem.
Specifically, the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0.
During training, the backpropagation of error estimates the amount of error for which the weights of a node in the network are responsible. Instead of updating the weight with the full amount, it is scaled by the learning rate.

This means that a learning rate of 0.1, a traditionally common default value, would mean that weights in the network are updated 0.1 * (estimated weight error) or 10% of the estimated weight error each time the weights are updated.
Generally, a large learning rate allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights but may take significantly longer to train

4.3.2 GoogleNet Model

GoogleNet is a deep convolution network. It is also known as inception architecture. The main idea behind inception architecture is how to estimate and distribute the dense components in the optimum sparse structure of a convolutional network.

The GoogleNet architecture is very different from previous state-of-the-art architectures such as AlexNet and ZF-Net. It uses many different kinds of methods such as *1x1* convolution and global average pooling that enables it to create deeper architecture.

Convolutional Module Implementation:
We construct the ‘conv_module’ which is a series of convolutional and a batch normalization layer that is ultimately passed through a Relu activation.

1. Input: Input to be processed
2. No of filters: No of filters that should be in the Conv2D layer.
3. FilterX and FilterY: Size of the filters.

4. Stride
5. Channel dimension
6. Padding: Predefined to ‘same’ for the whole model

Inception Module Implementation:

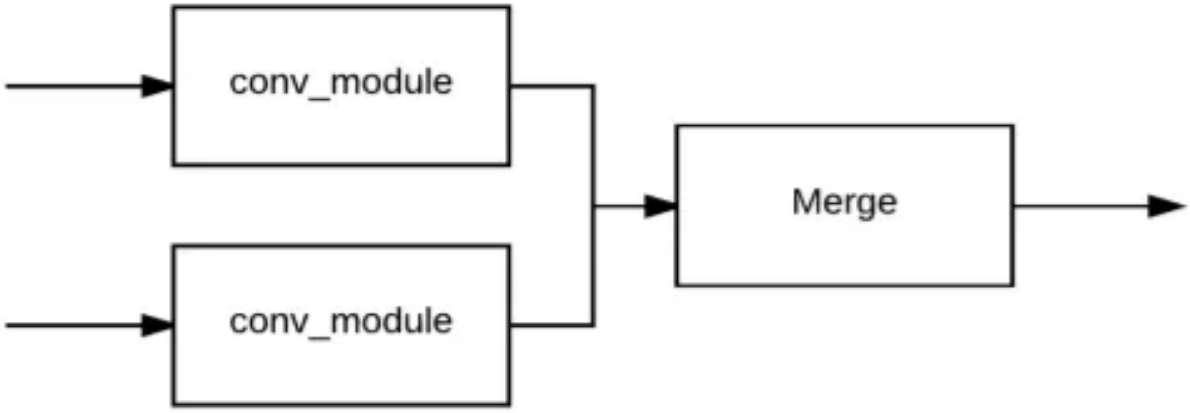


Fig 40: Inception Module

We define our modified inception module by using two conv_modules. The first module is initialized with:

1. 1×1 filters (used to learn local features in images).
2. The second and third with 3×3 and 5×5 respectively (responsible for learning general features).
3. Define the pool projection layer using a global pooling layer.
4. After that, we concatenate the layer outputs along the channel dimension(chanDim).

Down sample module:

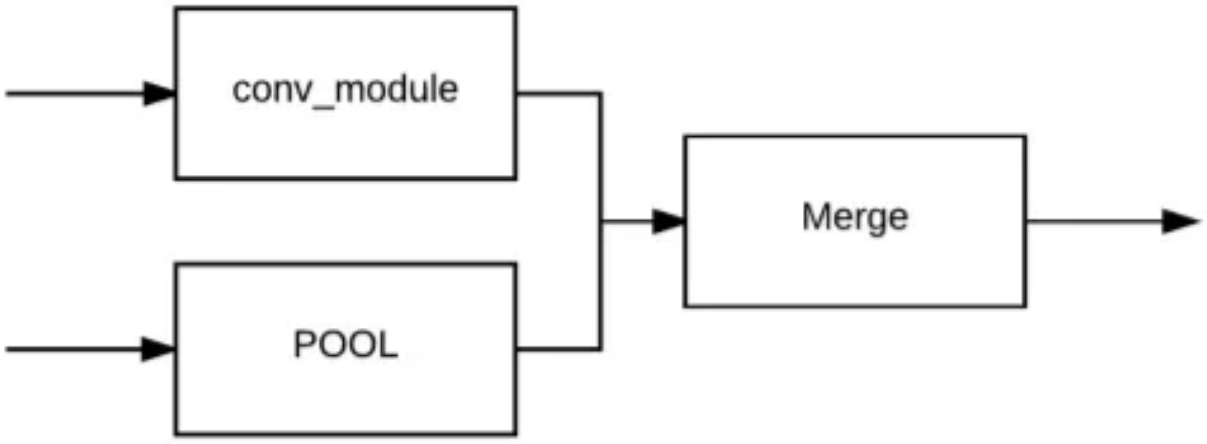


Fig 41: Down sample module

As we could see in the architecture which is supposed to be built above the model is very deep. Thus, it requires downsampling so that the no of trainable parameters can be controlled. For that function, we use a downsample_module. It is basically the concatenation of the output of a max-pooling layer and a conv_module (with 3×3 filters).

The model implementation:

1. Define an input layer with the width, height, and depth parameters of the function.
2. Use two inception modules along with a downsampling module.
3. Use Five inception modules along with a downsampling module.
4. Use two inception modules with pooling and dropout.
5. Flatten layer, with a dense layer (with no of units equal to no of classes) along with an activation layer with softmax classifier.

Chapter 5
User Manual

we have designed this application to be as easy as it can be and efficient as well.

1- What users do and see:

1- First: the user put the URL of the web application in the browser.
2- Second: the user uploads the image of the skin.
3- third: the user put all needed information for the best results such as the anatomy site of the skin lesion.
4- 4th: it presses process to make the app start the operation.
5- 5th: the result will show -> the highest possible disease that the model detect will be shown on the screen and the percentage of having it.
6- 6th: a URL for this skin disease from a certified pages will appear so that the user can know more about the disease that he might have.

2- What happens in the back ground of the system

a. the system takes the photo from the user then make it go through image processing techniques such as hair removal and vignette removal to get the best results from the model.
b. The system takes the information from the user and fed it to the model
c. The system takes the highest skin lesion probability and show it to the user.

d. The system directs the user to the best web page taking about the disease of his through a link shown on the screen.

Fig 42: welcome page of application

This is the first page that the user sees when he opens the web application “the welcome page “ it is just shown for 2 seconds then direct hem to the first page “the main page”

the web app design

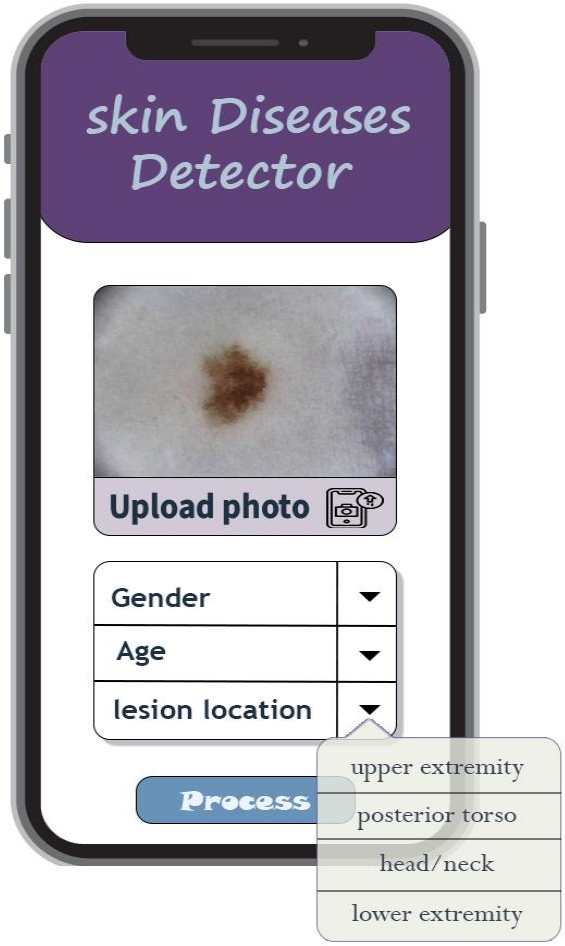


Fig 43: main page of application

This is the main page: the page that all the work is done.

The user inters his information “the same type of information that have been used to train the model in the first place”.

The gender -> male, female

The age -> as it is an integer

The lesion location “the general anatomy site of the skin lesion”

It can be -> upper extremity, posterior torso, anterior torso, head/neck, lower extremity.

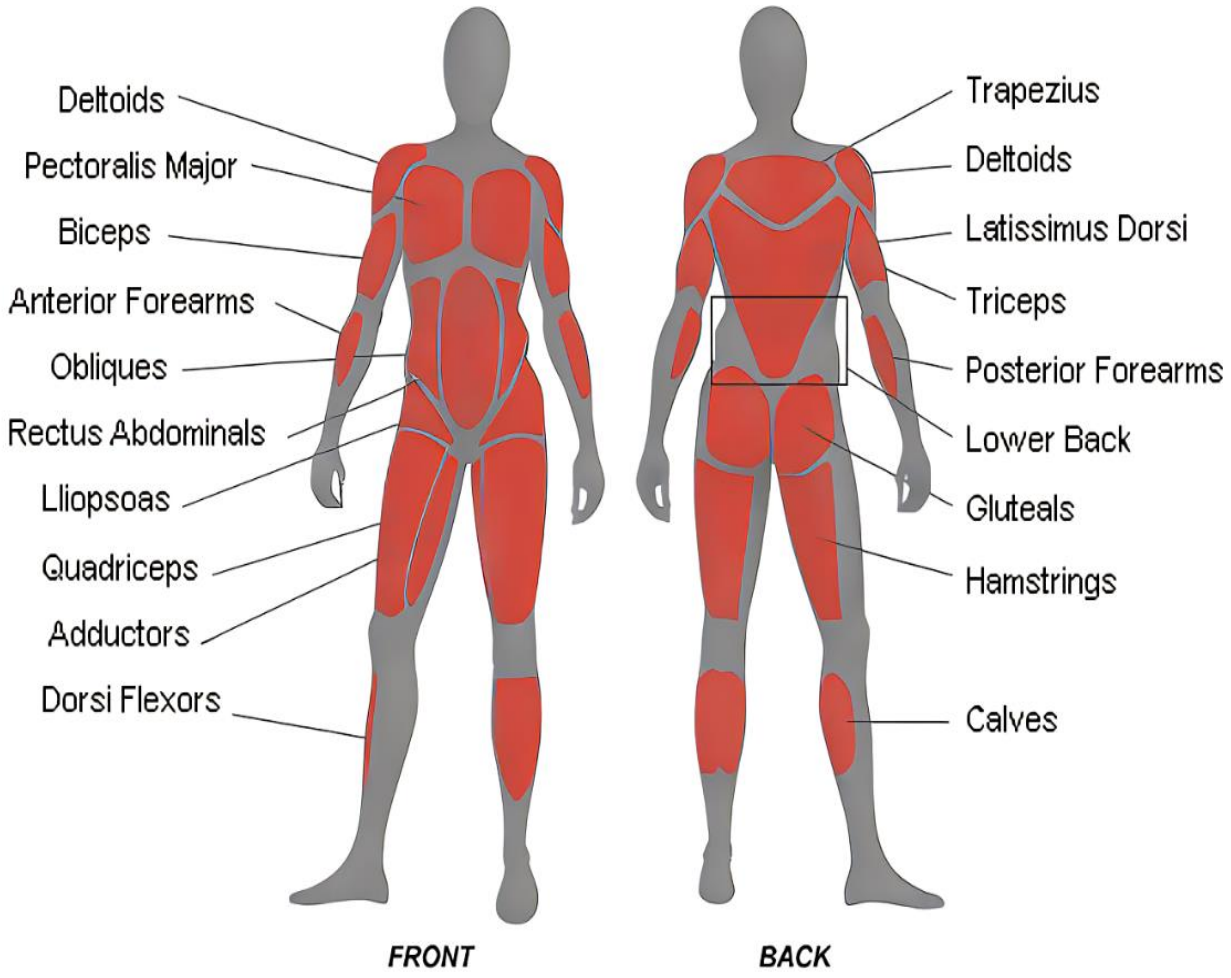


Fig 44: the anatomy of the body

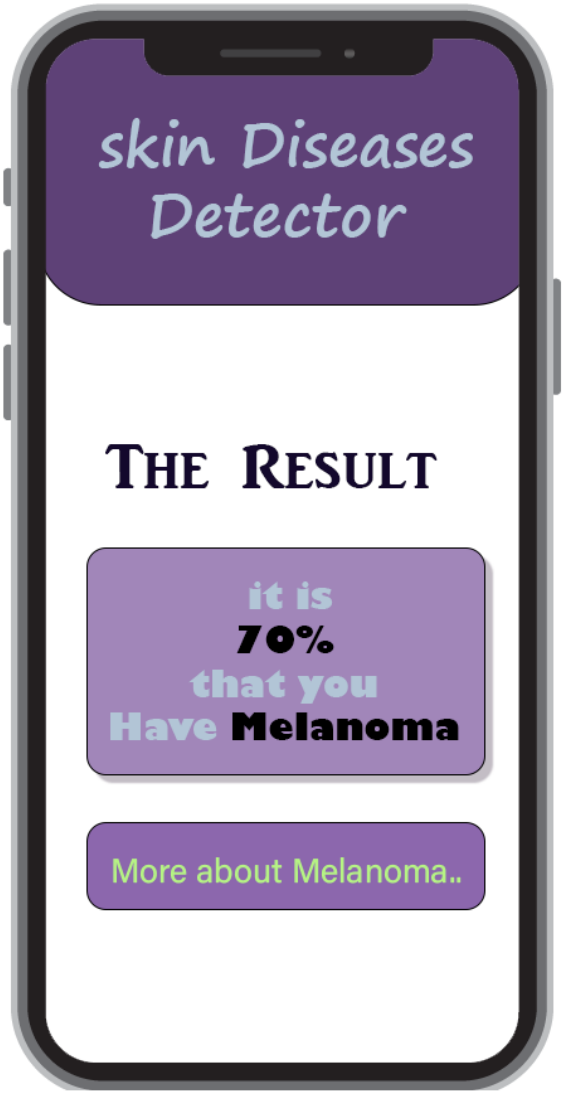


Fig 45: Result Page of the application

This is the result page that shows the percentage of the skin disease and the link of the directed to page to know more about that disease.

Chapter 6
Conclusions and Future Work

6.1 Conclusions

In the end, the system we have accomplished until now is ideally designed for medical diagnosis in health care
for skin cancer diseases. we have tried and succeeded in accomplishing a user-friendly web application containing
a strong model to classify skin cancer disease only by using little information such as the image of the skin
lesion and some of the basic information about the patient such as gender and age. all of that comes to a great result
due to the well-trained model and all the image processing and cleaning techniques that have been done, and data science and
machine learning approaches. we have exceeded the target accuracy of 90% in the first place and the web application is now available for personal use. we hope it helps people to diagnose their disease as fast as they take a photo of it and help them cure it in its early phase.

6.2 Future Work

In the future, We hope that combine all dataset of all years without repetition and with one bigger dataset and that will increase the accuracy of prediction diseases.
It will increase the number of predicted diseases by adding pictures of new diseases in the dataset.
Increase the number of pictures of existing diseases, by enabling user to upload pictures of their diseases that they are sure of their diagnose without disclosing their personal information. This will help increase accuracy.

Improving images preprocessing by many steps such as: image segmentation process to remove the background of the image and eliminate noise in the form of hair and air bubbles.

Adding many features to the application:

- Connect the user with the best doctors
- The use has medical record that can be shared with his doctor

- Using the image processing techniques to take excellent shot affected area of the skin when the area captures a photo.
- Notify the user to do some periodic checkups.

References

[1] Tschandl P., Rosendahl C. & Kittler H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci. Data 5, 180161 doi.10.1038/sdata.2018.161 (2018)

[2] Noel C. F. Codella, David Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, Allan Halpern: "Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC)", (2017)

[3] Marc Combalia, Noel C. F. Codella, Veronica Rotemberg, Brian Helba, Veronica Vilaplana, Ofer Reiter, Allan C. Halpern, Susana Puig, Josep Malvehy: "BCN20000: Dermoscopic Lesions in the Wild", 2019

[4] Kassem MA, Hosny KM, Fouad MM. Skin lesions classification into eight classes for ISIC 2019 using deep convolutional neural network and transfer learning. IEEE Access. 2020 Jun 19;8:114822-32.

[5] Ross-Howe, Sara, and Hamid R. Tizhoosh. "The effects of image pre-and post-processing, wavelet decomposition, and local binary patterns on U-nets for skin lesion segmentation." 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018.

[6] Ross-Howe, Sara, and Hamid R. Tizhoosh. "The effects of image pre-and post-processing, wavelet decomposition, and local binary patterns on U-nets for skin lesion segmentation." 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018.

[7] Shahin, Ahmed H., Ahmed Kamal, and Mustafa A. Elattar. "Deep ensemble learning for skin lesion classification from dermoscopic images." 2018 9th Cairo International Biomedical Engineering Conference (CIBEC). IEEE, 2018.

[8] Pugazhenthi, V., et al. "Skin disease detection and classification." International Journal of Advanced Engineering Research and Science (IJAERS) 6.5 (2019): 396-400.

[9] S. Lu, Z. Lu, and Y.-D. Zhang, ‘‘Pathological brain detection based on AlexNet and transfer learning,’’ J. Comput. Sci., vol. 30, pp. 41–47, Jan. 2019.

[10] Kadampur, M.A. and Al Riyaee, S., 2020. Skin cancer detection: Applying a deep learning based model driven architecture in the cloud for classifying dermal cell images. Informatics in Medicine Unlocked, 18, p.100282.

[11] Emuoyibofarhe, Justice O., et al. "Early Skin Cancer Detection Using Deep Convolutional Neural Networks on Mobile Smartphone." International Journal of Information Engineering & Electronic Business 12.2 (2020).

[12] Pacheco, Andre GC, Abder-Rahman Ali, and Thomas Trappenberg. "Skin cancer detection based on deep learning and entropy to detect outlier samples." arXiv preprint arXiv:1909.04525 (2019).

[13] Hasan, Mahamudul, et al. "Skin cancer detection using convolutional neural network." Proceedings of the 2019 5th international conference on computing and artificial intelligence. 2019.

[14] Benbrahim, Houssam, Hanaâ Hachimi, and Aouatif Amine. "Deep convolutional neural network with tensorflow and keras to classify skin cancer images." Scalable Computing: Practice and Experience 21.3 (2020): 379-390.

[15] Dermnet Skin Disease Atla. Accessed: May 13, 2020. [Online]. Available: http://www.dermnet.com/

[16] DermNetZ. [Online] http://www.dermnetnz.org

[17] Kaggle. [Online] https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000

[18] Britannica. [Online] https://www.britannica.com/science/human-skin-disease/Generalized-skin-diseases