SKIN LESION ANALYSIS BASED ON DENSELY CONNECTED CONVOLUTIONAL NEURAL NETWORKS

A Major Project Report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

Submitted by

G. Eeshapriya 221710308016

Under the esteemed guidance of

Mr. Joshi Vinay Kumar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM

(Deemed to be University)

HYDERABAD

MAY 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

I hereby declare that the project entitled " **Skin Lesion Analysis based on**

Densely connected Convoluted Neural Network" is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech, in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 11-05-2021

G. Eeshapriya

(221710308016)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project entitled "**Skin Lesion Analysis based on Densely connected Convoluted Neural Network**" is a bonafide record of work carried out by **G. Eeshapriya (221710308016)** submitted in partial fulfilment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering.

Project Guide	**Head of the Department**
Mr. Vinay Kumar Joshi	**Dr. S. Phani Kumar**
(Assistant Professor)	**(Professor)**

ACKNOWLEDGMENT

Our Major Project might now no longer were a success without the assist of numerous people. we would really like to thank the personalities who have been a part of our seminar in several ways, people who gave us exquisite assist from the start of the project.

We are extraordinarily grateful to our honourable Pro-Vice-Chancellor, **Prof. N. Siva Prasad** for offering the important infrastructure and sources for the accomplishment of our assignment.

We are pretty indebted to **Prof. N. Seetharamaiah**, Principal, School of Technology, for his help for the duration of the tenure of the assignment.

We are very plenty obliged to our loved **Prof. S. Phani Kum**ar, Head of the Department of Computer Science & Engineering for offering the possibility to adopt this assignment and encouragement with inside the finishing touch of this assignment.

We hereby desire to explicit our deep experience of gratitude to **Dr. S. Aparna**, Assistant Professor, Department of Computer Science and Engineering, School of Technology and to **Mr. Vinay Kumar Joshi**, Assistant Professor, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, ethical help and beneficial recommendation furnished via way of means of them for the fulfilment of the predominant mission.

We also are grateful to all of the group of worker's participants of Computer Science and Engineering branch who've cooperated in making our mission a fulfilment. We would love to thank all our mother and father and pals who prolonged their help, encouragement and ethical help both without delay or in a roundabout way in our mission work.

Sincerely,

G. Eeshapriya 221710308016

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES **PAGE NO**

1.ABSTRACT

Cancer is one of the deadliest illnesses, which is causing demise for one out of six persons. The possibility of recognizing malignancy at first would help give better treatment and lessen death cases. As innovation is improving continuously, the malignancy can be distinguished utilizing the strategy called DCNN, which is known to be Densely Connected Convolutional Neural Networks.

Here we will identify the disease of the skin by investigating the skin sores. It takes from our base paper known as Skin Lesion Segmentation with Improved Convolutional Neural Networks.Here they have segmented and analyzed only for skin disease Melonama.

In our base paper, they utilized a strategy of CNN. However, we use DCNN as this is profoundly effective in analysing skin lesion. We chose our dataset from Kaggle with 10015 pictures that helps providing efficient analysis in detecting seven kinds of skin diseases.

2.INTRODUCTION

A DCNN is a neural network that is used primarily for ordering images and performing object recognition in the scene. For example, densely connected convolutional neural networks recognize different parts of the face, people, road signs, tumors, platypus, and visual information.

Dermoscopy is a successful testing procedure for identifying skin lesions. By removing indications on the skin's surface and visually multiplying specific areas many times, it is easy to distinguish between the skin below the epidermis and the actual joints between the epidermis and the upper dermis, structures that are generally invisible to the naked human eye will do it.

Dermoscopy to detect colored skin lesions and inflammatory diseases, differential diagnosis and specific diagnosis of benign and dangerous skin tumors, and related skin monitoring and telemedicine ignore some degree of unwanted skin biopsy.

This provides the patient with a new and relevant skin identification method and reduces most of the pain of cumbersome tests. It also helps doctors recognize symptoms and evaluate diseases to identify causes.

In fact, the specific classification of skin damage in dermoscopic photographs is the enormous size of the diversity within classes and the similarities between classes, and the limited ability to presuppose, caused by differences in imaging strategies and pathology. clinic. it is still a demanding subject.

These days, deep learning-based techniques for grouping skin pain have been widely gained. The dense convolutional neural network has made a leap in clustering in image classification. The foundation of the organization is one of the tell-tale elements of CNN's design example.

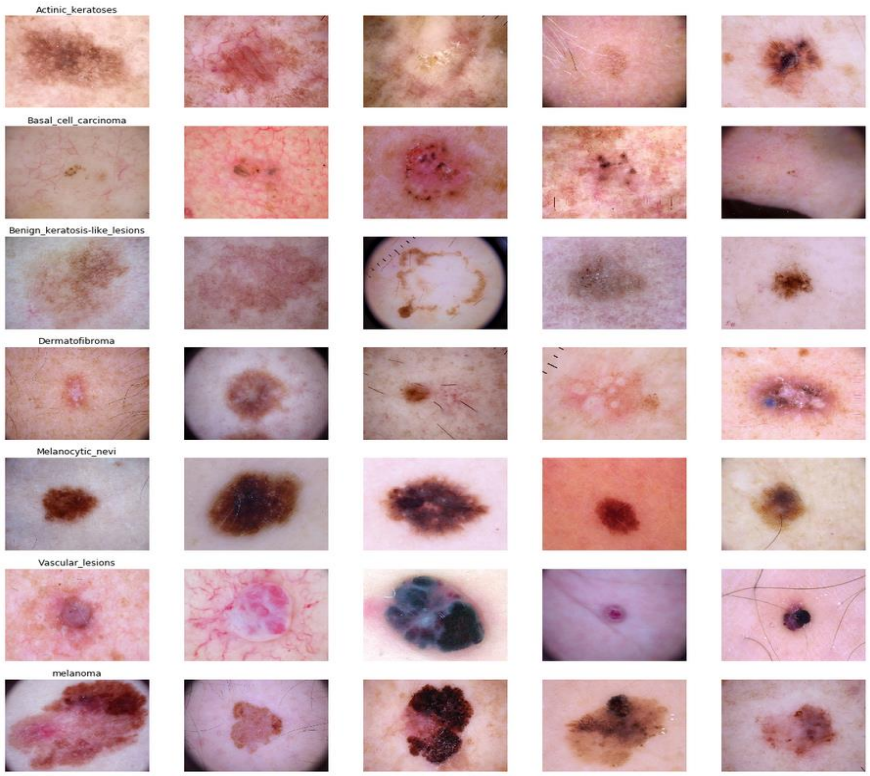


Fig. 2 Skin Cancer

2.1Purpose of Project

The project's primary purpose is to analyze skin lesions and detect seven different kinds of cancers. We have created the seven different classes and will detect which kind of cancer this is through images. This detection we are going to do using DCNN.

2.2Overview of Project

Here we will take 100015 images on which we are going to train and test the model. We are going to divide it into a training set and a testing set. After training the model with various images using DCNN, we will test it and predict the kind of cancer.

3.LITERATURE SURVEY

3.1 Survey on Skin Lesion analysis

There have been connected works done in the past to recognize skin illnesses utilizing AI and profound learning. Catarina Barata and Jorge S. Marques [Barata2019] directed an investigation on utilizing profound learning to analyze skin disease by utilizing various levels structures. They found that skin injury coordinate in various levels, which is considered by dermatologists when diagnosing them. Nonetheless, programmed frameworks do not utilize this data, playing out the determination in a one-versus - all methodology, where a wide range of sores thinks of.

The overview they proposed to imitate the clinical procedure and train a profound learning design to play out a progressive analysis. Their outcomes feature the advantages of tending to the arrangement of dermoscopy pictures in an organized manner. Also, they give a broad assessment of models that should consider in the improvement of analytic frameworks dependent on profound learning.

The primary CNN, comprising 20 convolution layers with 3×3 channels, is a VGG organization. The second make out of 20 systems administration organization (NiN) layers, altering the original structure. When preparing these organizations for human skin discovery, we consider fix-based and entire picture-based preparing.

The primary technique centers around neighborhood highlights, for example, skin tone and surface, and the second on the human-related shape includes just as shading and surface.

Investigations show that the proposed CNNs yield preferable execution over the traditional techniques and then the current profound learning-based strategy. Additionally, it tracks down that the NiN structure, for the most part, shows higher exactness than the VGG based design.

These examinations additionally show that the entire picture-based preparing that learns the shape highlights yields preferable precision over the fix put together, discovering that concentrations concerning neighborhood tone and surface as it were.

Le Thu Thao and Nguyen Hong Quang led an examination on programmed skin sore investigation towards melanoma discovery. Profound learning techniques for picture investigation have shown excellent execution as of late. The review introduced profound learning-based ways to tackle two skin injury investigation issues utilizing a dermoscopic picture containing skin tumors.

Jignesh Rathod et al. [Rathod2018] directed an investigation on skin illnesses finding utilizing Convolutional Neural Networks. This overview proposes that dermatology is perhaps the most unusual and troublesome territory to analyze due to its intricacy.

In dermatology, numerous multiple times, broad tests are to be completed to settle on the skin condition the patient might be confronting. The time may shift from one professional to another. It additionally founded on the experience of that individual as well. Thus, there is a need for a framework that can analyze skin infections with no of these limitations.

They proposed a robotized picture-based framework for the acknowledgment of skin sicknesses utilizing AI characterization. This framework will use a computational method to break down, measure, and consign the picture information predicated on different highlights of the pictures.

Skin pictures are separated to eliminate undesirable commotion and measure it for the upgrade of the picture. Highlight extraction utilizing complex strategies like Convolutional Neural Network (DCNN), arrange the picture dependent on the calculation of SoftMax classifier, and acquire the analysis report as a yield.

This framework will give more exactness and create results quicker than the conventional strategy, making this application a proficient and reliable framework for

dermatological illness location. Besides, this can likewise utilize as a dependable ongoing encouraging instrument for clinical understudies in the dermatology stream.

Information pictures contain both sound and melanoma injury parts. As this may misdirect the preparation of convolutional neural organizations, melanoma injuries ought to be isolated from solid regions (ordinary) of the skin. Notwithstanding, editing solid skin may cause the deficiency of essential data, such as shading contrast between sound skin and sore utilized to segregate melanoma from favorable conditions.

Division of the skin injury from dermoscopy pictures is a significant viewpoint in recognizable melanoma proof since dermatologists utilize the state of skin sore in acknowledgment (Yang et al., 2017). In this manner, current skin sore characterization follows three stages; 1) bitter division, ii) highlight extraction from the divided locale, and iii) sore order.

All the more frequently, preprocessing step is performed before division to decrease the commotion of the picture (Fornaciali, Carvalho, Bittencourt, Avila, and Valle, 2016). Thus the more significant part of specialists forces division to their picture order frameworks. Cavalcanti et al. have introduced a programmed grouping framework containing preprocessing, division, and highlight extraction steps.

They have utilized a two-phase classifier expecting to decrease the mistake pace of melanoma cases (explicitly, bogus negative cases) (Cavalcanti, Scharcanski, &Baranoski, 2013). Their proposed structure that joins the ABCD rule and melanin-variety includes the extraction method, which has shown 100% precision in melanoma arrangement.

A few explorations in writing turn out accomplished for melanoma characterization dependent on division, utilizing SVM and DCNN.Profound learning-based classifiers' application is far-reaching as they can surpass human ability in performing ordering errands for objects as a rule.

For example, Deep Neural Networks can figure complex assignments credited to nonlinear neuron handling and more major expectation power, delivering them reasonably to applies in clinical pictures.

The latest progressions in profound learning models incorporate VGG, AlexNet, ResNet, and Xception. Different examination works have additionally used these models in Computer-Aided Diagnose (CAD) as their exhibition is productive.

4.PROBLEM IDENTIFICATION& OBJECTIVE

The problem which we face in our project is get a good accuracy as there are a set of large amount of images where local system can’t perform with such a huge particulars.

4.1 Problem Statement

The main problem is that, this project need to work in real environment that is analyzing skin lesion on the different areas of human body. So, we need to create a model that is not complex and analyzed well.

4.2 Tools Used

4.2.1 Python

Python is well-known programming dialects. Indeed, it is more so than any other time. Python moved from the third spot to tie for second in the most recent positioning of programming language ubiquity distributed by the investigator firm RedMonk. It is the first occasion when that a language other than JavaScript, which stays number one in the company's evaluations, or Java, the other next in line, has entered the best two since RedMonk began gathering its rankings in 2012.

That achievement is even more huge given occasionally rough progress from the second form of Python, which the language's designers quit supporting this year, to the third form..

Fig. 4.2.1 Python

4.2.2 Anaconda

Anaconda is the information science stage for information researchers, IT experts and business heads of tomorrow. It is a dissemination of Python, R, and so on with in excess of 300 bundles for information science, it gets probably the best stage for any undertaking. In this python boa constrictor instructional exercise, we will examine how we can utilize boa constrictor for python programming.

Fig. 4.2.2 Anaconda

4.2.3 Jupyter

Jupyter is a task and local area whose objective is to "create open-source programming, open-principles, and administrations for intuitive figuring across many programming dialects".

Fig. 4.2.3 Jupyter-notebook

4.3 Components used

4.3.1 Nvidia CUDA

NVIDIA CUDA-empowered items, for example, the NVIDIA GPU module card.

NVIDIA has upheld this pattern by delivering the CUDA (Compute Unified Device Architecture) interface library to permit applications engineers to compose code that can be transferred into an NVIDIA-based card for execution by NVIDIA's hugely equal GPUs.

It permits application engineers to connect a teraflop-class, 480-processor, NVIDIA-based card, and transfer applications to run inside the NVIDIA GPU at far more superior speed than conceivable, even the quickest broadly applicable CPU on the motherboard.

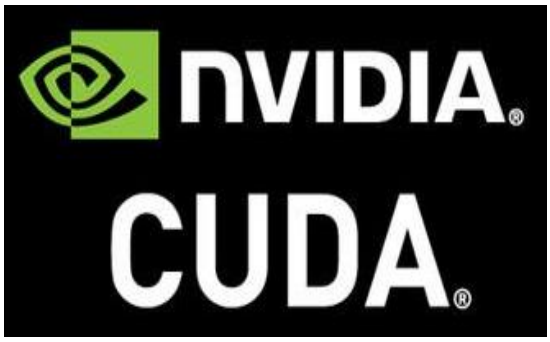


Fig. 4.3.1 Nvidia CUDA

4.4 Packages

4.4.1 Tensor-Flow

Itis a product library or system planned by the Google group to carry out AI and profound learning ideas in the simplest way. It joins the computational variable-based math of streamlining procedures for the simple estimation of numerous numerical articulations.

TensorFlow can execute profound neural organizations which are written by picture acknowledgment, word implanting, and different grouping models.

Fig. Tensor-Flow

4.4.2 Keras

The main motivations to utilize Keras come from its core values, principally the one about being easy to use.

Past simplicity of learning and simplicity of model structure, Keras offers the upsides of expansive reception, support for a wide scope of creation organization alternatives, joining with in any event five back-end motors and solid help for different GPUs and dispersed preparing.

Fig. Keras

4.4.3 Pillow

It bases on top of PIL (Python Image Library). PIL is one of the significant modules for picture preparation in Python.

Notwithstanding, the PIL module is not upheld since 2011 and does not uphold python 3.

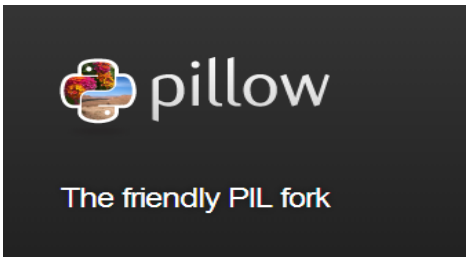


Fig. 4.4.3 Pillow

5.DESIGN- SYSTEM METHODOLOGY

5.1Introduction

UML is a visual language that is fundamentally utilized to plan. It helps programmers, finance managers, and framework drafting technicians to show, plan and dissect the framework.

Envisioning client connections, measures, and constructing the framework you are attempting to fabricate will help save time down the line and ensure everybody in the group is in total agreement.

5.2 Use case UML description

An arrangement of activities that produce a recognizable outcome for a particular actor.A set of situations integrated by a typical client objective. It gives construction to social things. It acknowledges through a joint effort.

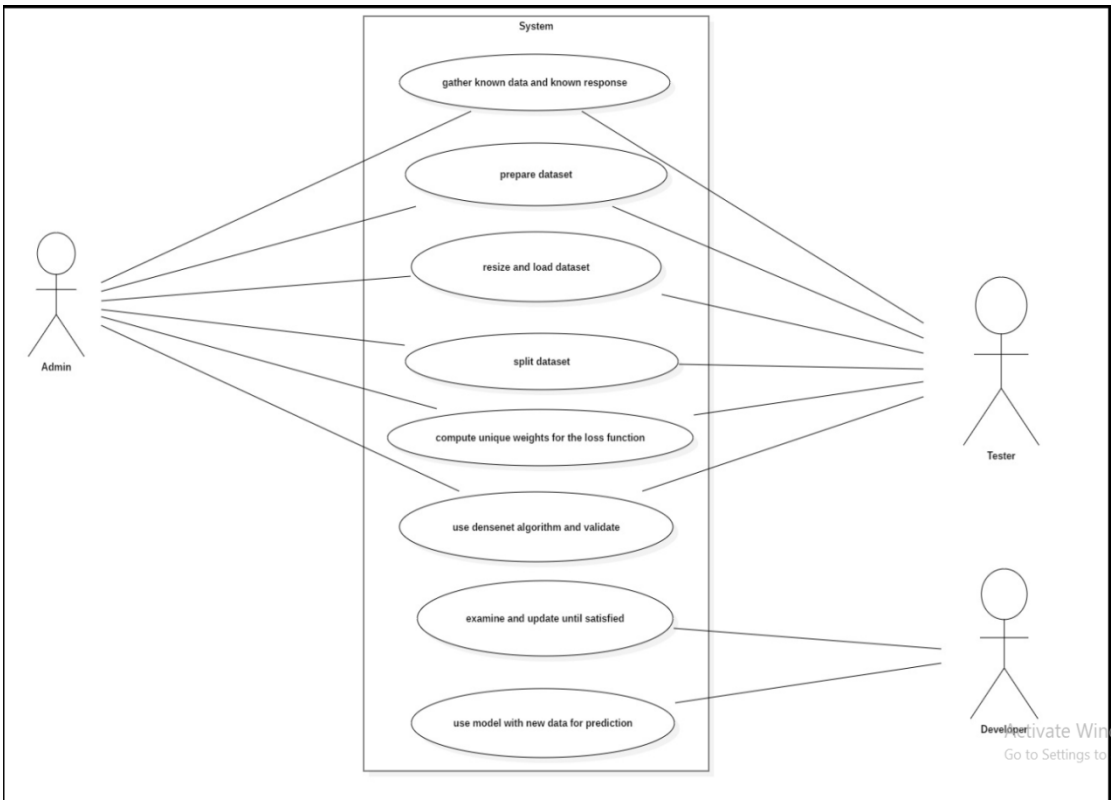


Fig. 5.2 use case UML diagram

5.3 Activity UML Description

Here the show stream was starting with one movement then onto the next activity. It comprises of activity states,action states, and transitions. It is a continuous non-nuclear execution. It comprises of start state and stops state. It additionally comprises fook, join, merge, and choices.

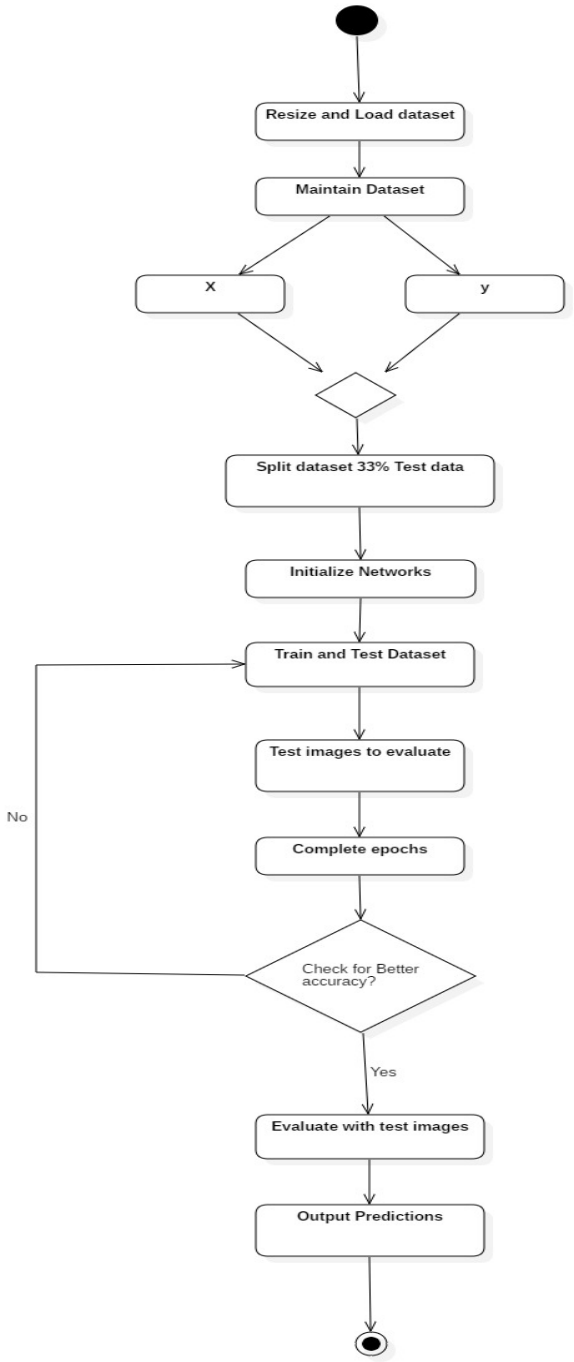


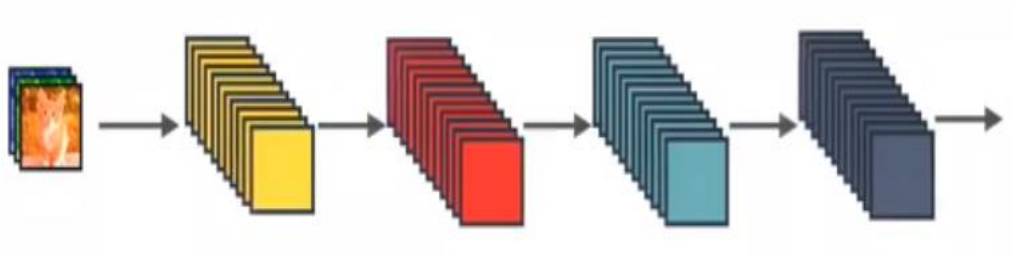
Fig. 5.3 Activity UML diagram

6. OVERVIEW OF TECHNOLOGIES

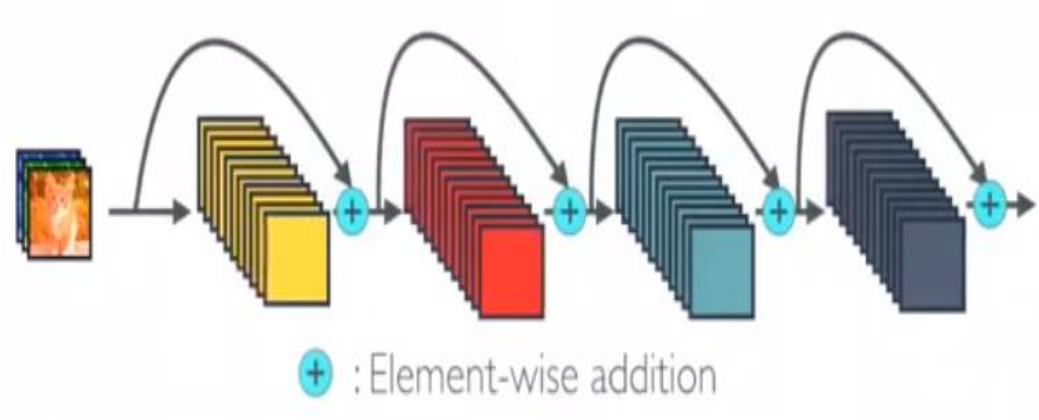
The overview they proposed mirrors the clinical methodology and trains a profound learning design to play out a progressive finding. Their outcomes feature the advantages of tending to the arrangement of dermoscopy pictures in an organized manner. Also, they give a broad assessment of standards that should consider in the advancement of symptomatic frameworks dependent on profound learning.

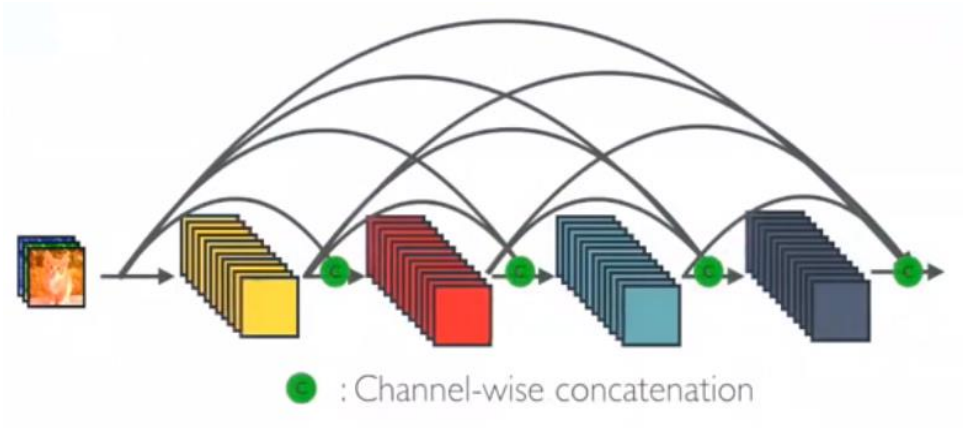
.

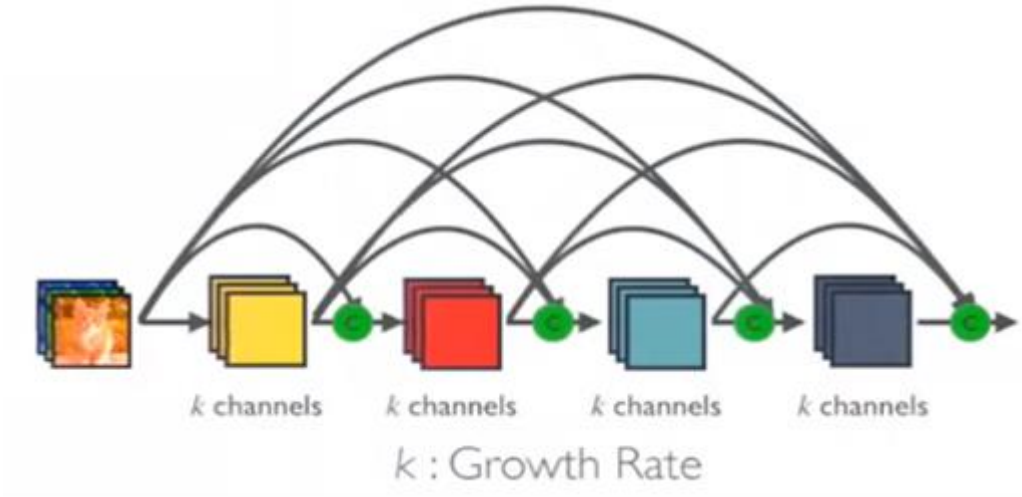
6.1Dense Block

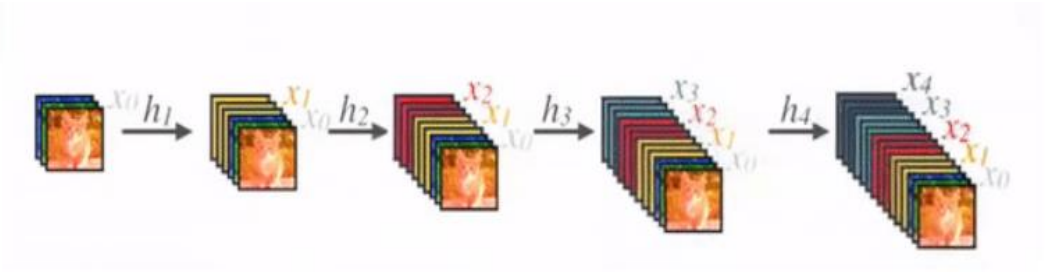


During derivation, an E-layer thick square stores O(E) yield. The halfway component mapping used to process the particular yield includes and does not should put away. The GPU memory utilization is along these lines direct in the profundity of the organization.



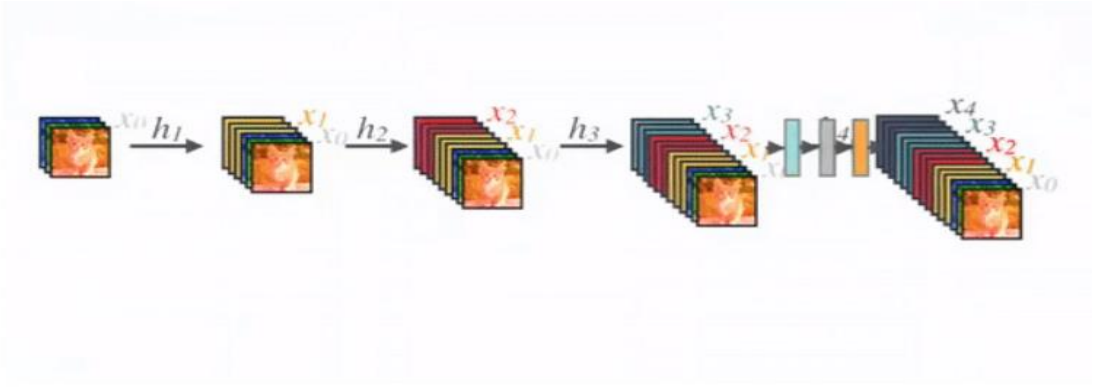


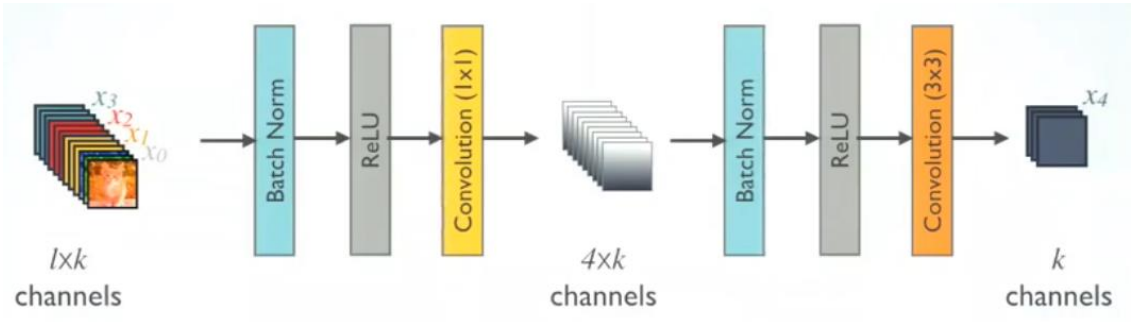




Concatenation during Forward Propagation

6.1.1 DenseNet Architecture





We can pre-dispense solitary memory support that will eventuallycontain all the yield highlight maps of a thick square to stay away from this excess.

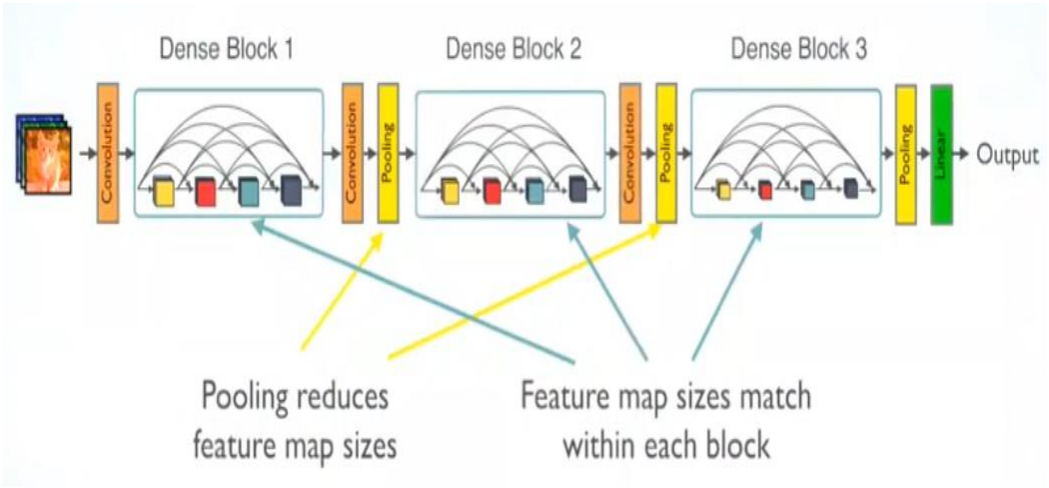


Fig. 6.1.1 DenseNet Block

7.IMPLEMENTATION

7.1Implement through DCNN

By and large DCNN designing includes a heap of convolutional layers, the pooling layers, and the complete related layers. In each of the convolutional layer, essential models in close by regions of information sources are removed by convolving a channel over the information sources. The convolution result is taken care of in a part map, an extent of how well the channel arranges with each piece of the information sources.

After the convolution, the pooling layer plays out a subsampling step to add up to estimations of these features, diminishes the parts of data, and mitigates over fitting the issues. The non-linear institution work is applied to create the yield of a layer.

The load of the convolutional and the pooling layers trails by the totally related layers, which will further absolute the close by information gained in the convolutional likewise, the pooling layers for class isolation.

By trading the topographies of the convolutional and the pooling layers, fantastic DCNN can work for unequivocal applications, for instance, the LeNet, the AlexNet, and the GoogLeNet, With the no lack of misrepresentation, we use LeNet-5 that is, the fifth time of the LeNet for the digits affirmation in this discussion what is more, tests all through the paper and a proposed plan the method can be oblige other DCNN as well Gear Based Neuron Cell. As all the principal structure block in the DCNN, the neuron will perform the three focal errands, that is, convolution, establishment, and pooling.

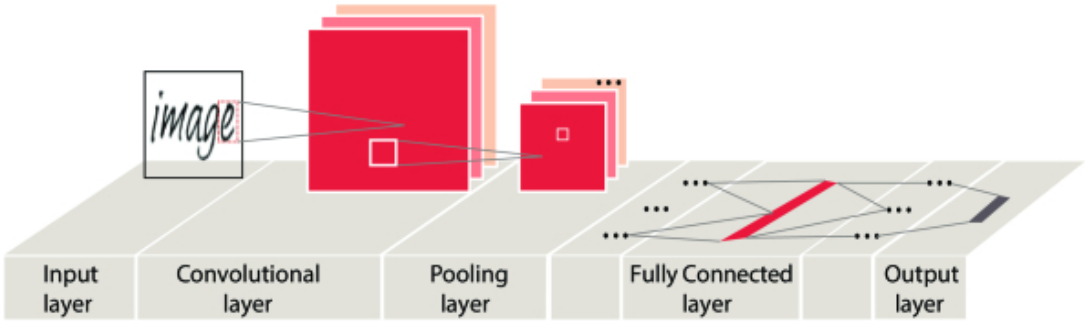


Fig. 7.1 Architecture of DCNN

7.2Test and Validate

The reasonableness of convolutional nets in picture apparent reason why the world has woke up adequacy of huge studying. It says DCNNs is the motivation driving why huge learning celebrates.

The achievement of a critical convolutional design that is AlexNet in the 2012 ImageNet dispute was the shot heard around the world. DCNN is empowering basic advance in PC vision, which has an accurate applications for the self-driving vehicles, mechanical development, drones, clinical judgements, security, and medications for the apparently weakened.

A typical measurement for testing the nature of your testing suite is called code inclusion. Code inclusion is a number that passes on the level of your advancement code that covers by your test work. It assists you with getting the corner cases you may have missed testing.

A few associations use code inclusion as a piece of their advancement pipeline, and no submit is permitted to except if it accomplishes the necessary code inclusion (at least 90% across most tasks).

While code inclusion is significant and significant, it is anything but a good measurement for checking the nature of the test suite. It essentially checks for the number of lines of advancement code that are trying.

For instance: If you are trying a capacity for tracking down the most extreme in a cluster, you might be trying the usefulness of the capacity and get 100% code inclusion yet not experiments, for example, when the exhibit is outside the allowed boundaries. Consequently, manual testing is likewise significant.

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under t
he input directory

import os
# for dirname, _, filenames in os.walk('/kaggle/input'):
#     for filename in filenames:
#         print(os.path.join(dirname, filename))

print(os.listdir("/kaggle/input/"))
print(os.listdir("/kaggle/input/skin-cancer-mnist-ham10000/"))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as out
put when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the cur
rent session
```

```
['skin-cancer-mnist-ham10000']
['hmnist_8_8_RGB.csv', 'hmnist_28_28_RGB.csv', 'HAM10000_images_part_1', 'ham10000_images_pa
rt_1', 'hmnist_8_8_L.csv', 'HAM10000_images_part_2', 'ham10000_images_part_2', 'hmnist_28_28
_L.csv', 'HAM10000_metadata.csv']
```

In [2]:

```
#read metadata
df_skin = pd.read_csv('/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_metadata.csv')

df_skin.head()
```

Out[2]:

	lesion_id	image_id	dx	dx_type	age	sex	localization
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear

In [3]:

```
# lesion names are given in the description of the challenge
lesion_type_dict = {
    'nv': 'Melanocytic nevi',
    'mel': 'Melanoma',
    'bkl': 'Benign keratosis-like lesions ',
    'bcc': 'Basal cell carcinoma',
    'akiec': 'Actinic keratoses',
    'vasc': 'Vascular lesions',
    'df': 'Dermatofibroma'
}

lesion_ID_dict = {
    'nv': 0,
    'mel': 1,
    'bkl': 2,
    'bcc': 3,
    'akiec': 4,
    'vasc': 5,
    'df': 6
}

lesion_names = ['Melanocytic nevi','Melanoma','Benign keratosis-like lesions ',
                'Basal cell carcinoma','Actinic keratoses','Vascular lesions',
                'Dermatofibroma']

lesion_names_short = ['nv','mel','bkl','bcc','akiec','vasc','df']

df_skin['lesion_type']=df_skin['dx'].map(lesion_type_dict)
df_skin['lesion_ID'] = df_skin['dx'].map(lesion_ID_dict)

print('Total number of images',len(df_skin))
print('The problem is unbalanced, since Melanocytic nevi is much more frequent that other label
s')

df_skin['lesion_type'].value_counts()
```

Out[3]:

```
Total number of images 10015
The problem is unbalanced, since Melanocytic nevi is much more frequent that other labels


Melanocytic nevi                6705
Melanoma                        1113
Benign keratosis-like lesions    1099
Basal cell carcinoma             514
Actinic keratoses                327
Vascular lesions                 142
Dermatofibroma                   115
Name: lesion_type, dtype: int64
```

In [4]:

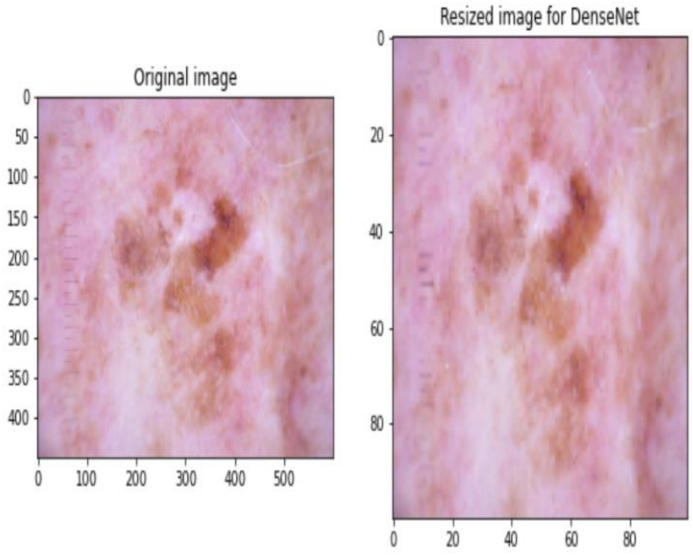
```
# read the first image
fname_images = np.array(df_skin['image_id'])
file_to_read ='/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/'+str(fname_imag
es[0])+'.jpg'

import cv2
from cv2 import imread, resize

img = imread(file_to_read)
img2 = resize(img,(100,100))

# show one exampe image

plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.imshow(img[:,:,::-1])
plt.title('Original image')
plt.subplot(1,2,2)
plt.imshow(img2[:,:,::-1])
plt.title('Resized image for DenseNet')
plt.show()
```

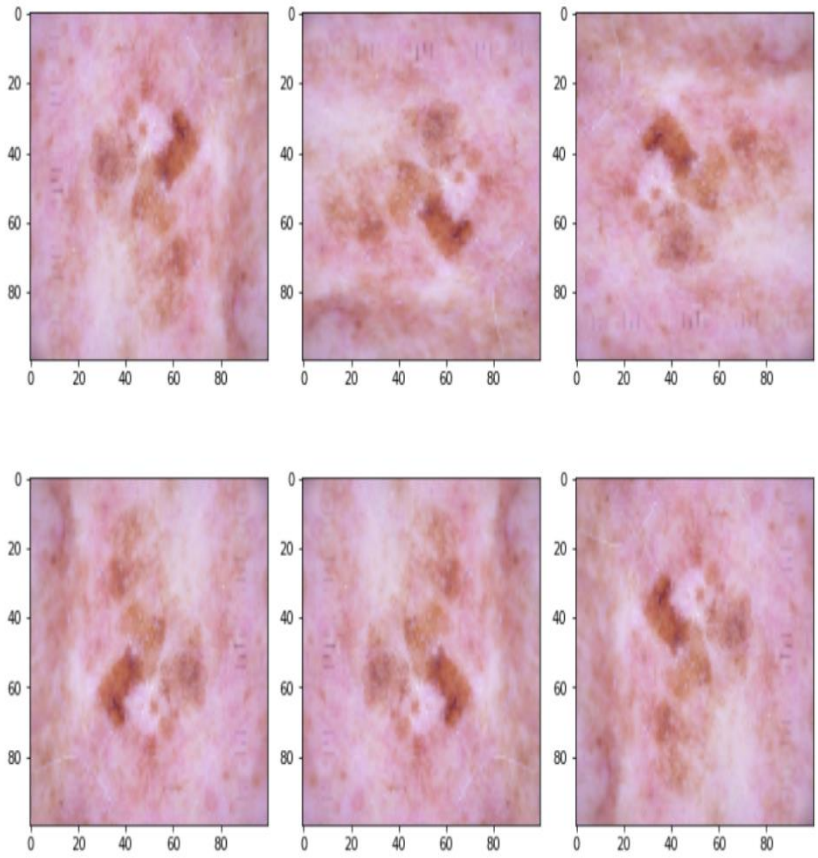


In [5]:

```
def produce_new_img(img2):
    # produce new images by rotating of flipping the original one
    # this helps to increase the dimension of the dataset, avoiding overfitting of a single class
    imga = cv2.rotate(img2,cv2.ROTATE_90_CLOCKWISE)
    imgb = cv2.rotate(img2,cv2.ROTATE_90_COUNTERCLOCKWISE)
    imgc = cv2.rotate(img2,cv2.ROTATE_180)
    imgd = cv2.flip(img2,0)
    imge = cv2.flip(img2,1)
    return imga,imgb,imgc,imgd,imge

new_img = produce_new_img(img2)

plt.figure(figsize=(10,8))
plt.subplot(2,3,1)
plt.imshow(img2[:,:,::-1])
for i in range(5):
    plt.subplot(2,3,2+i)
    plt.imshow(new_img[i][:,:,::-1])
plt.tight_layout()
plt.show()
```



In [6]:

```
# import images from 2 different folders

X = []
y = []

lista1 = os.listdir('/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/')
lista2 = os.listdir('/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_2/')


#import images from folder 1
for i in range(len(lista1)):
    fname_image = lista1[i]
    fname_ID = fname_image.replace('.jpg','')

    #features
    file_to_read ='/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/'+str(fname
_image)
    img = imread(file_to_read)
    img2 = resize(img,(100,100))
    X.append(img2)

    #targets
    output = np.array(df_skin[df_skin['image_id'] == fname_ID].lesion_ID)
    y.append(output[0])

    # add more images for class between 1-6, rotating them
    if output != 0:
        new_img = produce_new_img(img2)
        for i in range(5):
            X.append(new_img[i])
            y.append(output[0])

    if i % int(100) == 0:
        print(i,'images loaded')

# import images from folder 2
for i in range(len(lista2)):
    fname_image = lista2[i]
    fname_ID = fname_image.replace('.jpg','')

    #features
    file_to_read ='/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_2/'+str(fname
_image)
    img = imread(file_to_read)
    img2 = resize(img,(100,100))
    X.append(img2)

    #targets
    output = np.array(df_skin[df_skin['image_id'] == fname_ID].lesion_ID)
    y.append(output[0])

    # add more images for class between 1-6
    if output != 0:
        new_img = produce_new_img(img2)
        for i in range(5):
            X.append(new_img[i])
            y.append(output[0])

    if i % int(100) == 0:

        print(len(lista1)+i,'images loaded')
```

```
0 images loaded
100 images loaded
200 images loaded
300 images loaded
400 images loaded
500 images loaded
600 images loaded
700 images loaded
800 images loaded
1100 images loaded
1200 images loaded
1500 images loaded
1600 images loaded
1800 images loaded
2100 images loaded
2300 images loaded
2400 images loaded
2600 images loaded
2700 images loaded
2800 images loaded
2900 images loaded
3000 images loaded
3100 images loaded
3200 images loaded
3500 images loaded
3600 images loaded
3700 images loaded
3900 images loaded
4100 images loaded
4500 images loaded
4600 images loaded
4700 images loaded
4800 images loaded
4900 images loaded
5000 images loaded
5100 images loaded
5300 images loaded
5400 images loaded
5700 images loaded
5900 images loaded
6000 images loaded
6100 images loaded
6200 images loaded
6400 images loaded
6600 images loaded
6700 images loaded
6800 images loaded
6900 images loaded
7000 images loaded
7100 images loaded
7200 images loaded
7300 images loaded
7400 images loaded
7500 images loaded
7600 images loaded
7700 images loaded
7800 images loaded
8000 images loaded
8100 images loaded
8300 images loaded
8400 images loaded
8700 images loaded
8900 images loaded
9100 images loaded
9200 images loaded
9400 images loaded
9500 images loaded
9600 images loaded
9700 images loaded
9800 images loaded
9900 images loaded
```

In [7]:
```
from keras.utils.np_utils import to_categorical

X = np.array(X)
y = np.array(y)

y_train = to_categorical(y, num_classes=7)

# #convert targets in dummy variables, as required by softmax activation function
# y_dumm = np.array(pd.get_dummies(y))
```

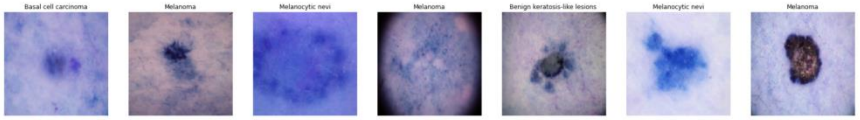
In [8]:
```
from sklearn.model_selection import train_test_split

# split in 80% training and 20% test data
X_train, X_test, y_train, y_test = train_test_split(X, y_train, test_size=0.33, random_state=5
0,stratify=y)


print('Train dataset shape',X_train.shape)
print('Test dataset shape',X_test.shape)
```

```
Train dataset shape (17798, 100, 100, 3)
Test dataset shape (8767, 100, 100, 3)
```

In [9]:
```
fig, ax = plt.subplots(1, 7, figsize=(30, 30))
for i in range(7):
    ax[i].set_axis_off()
    ax[i].imshow(X_train[i])
    ax[i].set_title(lesion_names[np.argmax(y_train[i])])
```



In [10]:
```
import keras
from keras.models import Sequential, load_model
from keras.callbacks import EarlyStopping,ModelCheckpoint
from keras.layers.core import Dropout, Activation
from keras.layers import Conv2D,BatchNormalization,MaxPool2D,Flatten,Dense
```

In [11]:
```
from sklearn.utils.class_weight import compute_class_weight
y_id = np.array(df_skin['lesion_ID'])

# compute weights for the loss function, because the problem is unbalanced
class_weights = np.around(compute_class_weight(class_weight='balanced',classes=np.unique(y_i
d),y=y),2)
class_weights = dict(zip(np.unique(y_id),class_weights))

print('The problem is unbalanced. We need to provide class_weights ')
print(class_weights)
```

```
The problem is unbalanced. We need to provide class_weights
{0: 0.57, 1: 0.57, 2: 0.58, 3: 1.23, 4: 1.93, 5: 4.45, 6: 5.5}
```

In [12]:

```
import keras
from keras.models import Model
from keras.layers import Conv2D, MaxPooling2D, Dense, Input, Activation, Dropout, GlobalAverag
ePooling2D, \
    BatchNormalization, concatenate, AveragePooling2D
from keras.optimizers import Adam



def conv_layer(conv_x, filters):
    conv_x = BatchNormalization()(conv_x)
    conv_x = Activation('relu')(conv_x)
    conv_x = Conv2D(filters, (3, 3), kernel_initializer='he_uniform', padding='same', use_bias
=False)(conv_x)
    conv_x = Dropout(0.2)(conv_x)

    return conv_x


def dense_block(block_x, filters, growth_rate, layers_in_block):
    for i in range(layers_in_block):
        each_layer = conv_layer(block_x, growth_rate)
        block_x = concatenate([block_x, each_layer], axis=-1)
        filters += growth_rate

    return block_x, filters


def transition_block(trans_x, tran_filters):
    trans_x = BatchNormalization()(trans_x)
    trans_x = Activation('relu')(trans_x)
    trans_x = Conv2D(tran_filters, (1, 1), kernel_initializer='he_uniform', padding='same', us
e_bias=False)(trans_x)
    trans_x = AveragePooling2D((2, 2), strides=(2, 2))(trans_x)

    return trans_x, tran_filters


def dense_net(filters, growth_rate, classes, dense_block_size, layers_in_block):
    input_img = Input(shape=(100, 100, 3))
    x = Conv2D(24, (3, 3), kernel_initializer='he_uniform', padding='same', use_bias=False)(in
put_img)

    dense_x = BatchNormalization()(x)
    dense_x = Activation('relu')(x)

    dense_x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(dense_x)
    for block in range(dense_block_size - 1):
        dense_x, filters = dense_block(dense_x, filters, growth_rate, layers_in_block)
        dense_x, filters = transition_block(dense_x, filters)

    dense_x, filters = dense_block(dense_x, filters, growth_rate, layers_in_block)
    dense_x = BatchNormalization()(dense_x)
    dense_x = Activation('relu')(dense_x)
    dense_x = GlobalAveragePooling2D()(dense_x)

    output = Dense(classes, activation='softmax')(dense_x)

    return Model(input_img, output)
```

In [13]:

```
dense_block_size = 6
layers_in_block = 8

growth_rate = 36
classes = 7
model = dense_net(growth_rate * 2, growth_rate, classes, dense_block_size, layers_in_block)
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 100, 100, 3) 0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 100, 100, 24) 648         input_1[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 100, 100, 24) 0           conv2d[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 50, 50, 24)   0           activation[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 50, 50, 24)   96          max_pooling2d[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 50, 50, 24)   0           batch_normalization_1[0]
[0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 50, 50, 36)   7776        activation_1[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 50, 50, 36)   0           conv2d_1[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 50, 50, 60)   0           max_pooling2d[0][0]
                                                                 dropout[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 50, 50, 60)   240         concatenate[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 50, 50, 60)   0           batch_normalization_2[0]
[0]
__________________________________________________________________________________________________
```

In [14]:

```
# training
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau


# anne = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5, patience=5, verbose=1, min_lr=1e-
3)
# checkpoint = ModelCheckpoint('model.h5', verbose=1, save_best_only=True)

# datagen = ImageDataGenerator(zoom_range = 0.2, horizontal_flip=True, shear_range=0.2)


# datagen.fit(xtrain)
# # Fits-the-model
# history = model.fit(datagen.flow(xtrain, ytrain, batch_size=128),
#                 steps_per_epoch=xtrain.shape[0] //128,
#                 epochs=50,
#                 verbose=2,
#                 callbacks=[anne, checkpoint],
#                 validation_data=(xtrain, ytrain))

early_stopping_monitor = EarlyStopping(patience=100,monitor='val_accuracy')
model_checkpoint_callback = ModelCheckpoint(filepath='model.h5',
                                            save_weights_only=False,
                                            monitor='val_accuracy',
                                            mode='auto',
                                            save_best_only=True,
                                            verbose=1)

batch_size = 32
epochs = 100
optimizer = Adam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=1e-3)
model.compile(optimizer = optimizer, loss = 'categorical_crossentropy', metrics=['accuracy'])

datagen = ImageDataGenerator(zoom_range = 0.2, horizontal_flip=True, shear_range=0.2)


# datagen = ImageDataGenerator(
#     featurewise_center=False,  # set input mean to 0 over the dataset
#     samplewise_center=False,  # set each sample mean to 0
#     featurewise_std_normalization=False,  # divide inputs by std of the dataset
#     samplewise_std_normalization=False,  # divide each input by its std
#     zca_whitening=False,  # apply ZCA whitening
#     rotation_range=45,  # randomly rotate images in the range (degrees, 0 to 180)
#     width_shift_range=0.2,  # randomly shift images horizontally (fraction of total width)
#     height_shift_range=0.2,  # randomly shift images vertically (fraction of total height)
#     horizontal_flip=True,  # randomly flip images
#     vertical_flip=False)  # randomly flip images

datagen.fit(X_train)
# # Fits-the-model
# history = model.fit(datagen.flow(xtrain, ytrain, batch_size=128),
#                 steps_per_epoch=xtrain.shape[0] //128,
#                 epochs=50,
#                 verbose=2,
#                 callbacks=[anne, checkpoint],
#                 validation_data=(xtrain, ytrain))

history=model.fit(datagen.flow(X_train,y_train), epochs=epochs, batch_size=batch_size, shuffle
=True, callbacks=[early_stopping_monitor,model_checkpoint_callback], validation_data=(X_test,
y_test), class_weight=class_weights)
```

```
Epoch 1/100
557/557 [==============================] - 118s 193ms/step - loss: 1.6311 - accuracy: 0.414
5 - val_loss: 1.6052 - val_accuracy: 0.5091

Epoch 00001: val_accuracy improved from -inf to 0.50907, saving model to model.h5
Epoch 2/100
557/557 [==============================] - 104s 187ms/step - loss: 1.1094 - accuracy: 0.559
3 - val_loss: 1.2223 - val_accuracy: 0.5829

Epoch 00002: val_accuracy improved from 0.50907 to 0.58287, saving model to model.h5
Epoch 3/100
557/557 [==============================] - 105s 188ms/step - loss: 0.9878 - accuracy: 0.588
7 - val_loss: 1.1414 - val_accuracy: 0.5865

Epoch 00003: val_accuracy improved from 0.58287 to 0.58652, saving model to model.h5
Epoch 4/100
557/557 [==============================] - 105s 188ms/step - loss: 0.9169 - accuracy: 0.610
4 - val_loss: 1.1518 - val_accuracy: 0.6055

Epoch 00004: val_accuracy improved from 0.58652 to 0.60545, saving model to model.h5
Epoch 5/100
557/557 [==============================] - 105s 188ms/step - loss: 0.8081 - accuracy: 0.649
1 - val_loss: 1.2003 - val_accuracy: 0.5739

Epoch 00005: val_accuracy did not improve from 0.60545
Epoch 6/100
557/557 [==============================] - 105s 189ms/step - loss: 0.7627 - accuracy: 0.664
0 - val_loss: 1.0931 - val_accuracy: 0.6205

Epoch 00006: val_accuracy improved from 0.60545 to 0.62051, saving model to model.h5
Epoch 7/100
557/557 [==============================] - 126s 227ms/step - loss: 0.7089 - accuracy: 0.679
0 - val_loss: 1.2726 - val_accuracy: 0.5842

Epoch 00007: val_accuracy did not improve from 0.62051
Epoch 8/100
557/557 [==============================] - 108s 194ms/step - loss: 0.6456 - accuracy: 0.698
3 - val_loss: 1.2407 - val_accuracy: 0.5999

Epoch 00008: val_accuracy did not improve from 0.62051
Epoch 9/100
557/557 [==============================] - 105s 188ms/step - loss: 0.5820 - accuracy: 0.716
3 - val_loss: 0.8465 - val_accuracy: 0.6902

Epoch 00009: val_accuracy improved from 0.62051 to 0.69020, saving model to model.h5
Epoch 10/100
557/557 [==============================] - 127s 227ms/step - loss: 0.5492 - accuracy: 0.733
0 - val_loss: 1.1914 - val_accuracy: 0.6026

Epoch 00010: val_accuracy did not improve from 0.69020
Epoch 11/100
557/557 [==============================] - 108s 194ms/step - loss: 0.5507 - accuracy: 0.733
2 - val_loss: 1.1971 - val_accuracy: 0.6286

Epoch 00011: val_accuracy did not improve from 0.69020
Epoch 12/100
557/557 [==============================] - 105s 188ms/step - loss: 0.5030 - accuracy: 0.756
8 - val_loss: 0.9379 - val_accuracy: 0.6704

Epoch 00012: val_accuracy did not improve from 0.69020
Epoch 13/100
557/557 [==============================] - 106s 190ms/step - loss: 0.4729 - accuracy: 0.772
2 - val_loss: 0.8970 - val_accuracy: 0.6933
```

```
Epoch 00091: val_accuracy did not improve from 0.89038
Epoch 92/100
557/557 [==============================] - 106s 190ms/step - loss: 0.0329 - accuracy: 0.982
3 - val_loss: 0.7939 - val_accuracy: 0.8052

Epoch 00092: val_accuracy did not improve from 0.89038
Epoch 93/100
557/557 [==============================] - 104s 187ms/step - loss: 0.0557 - accuracy: 0.973
0 - val_loss: 0.7449 - val_accuracy: 0.8076

Epoch 00093: val_accuracy did not improve from 0.89038
Epoch 94/100
557/557 [==============================] - 104s 187ms/step - loss: 0.0326 - accuracy: 0.984
3 - val_loss: 0.4374 - val_accuracy: 0.8799

Epoch 00094: val_accuracy did not improve from 0.89038
Epoch 95/100
557/557 [==============================] - 106s 191ms/step - loss: 0.0306 - accuracy: 0.985
0 - val_loss: 0.6253 - val_accuracy: 0.8402

Epoch 00095: val_accuracy did not improve from 0.89038
Epoch 96/100
557/557 [==============================] - 106s 190ms/step - loss: 0.0418 - accuracy: 0.977
6 - val_loss: 0.4236 - val_accuracy: 0.8834

Epoch 00096: val_accuracy did not improve from 0.89038
Epoch 97/100
557/557 [==============================] - 104s 186ms/step - loss: 0.0309 - accuracy: 0.985
4 - val_loss: 0.4766 - val_accuracy: 0.8693

Epoch 00097: val_accuracy did not improve from 0.89038
Epoch 98/100
557/557 [==============================] - 106s 191ms/step - loss: 0.0315 - accuracy: 0.983
8 - val_loss: 0.5529 - val_accuracy: 0.8480

Epoch 00098: val_accuracy did not improve from 0.89038
Epoch 99/100
557/557 [==============================] - 104s 186ms/step - loss: 0.0338 - accuracy: 0.982
4 - val_loss: 0.4898 - val_accuracy: 0.8754

Epoch 00099: val_accuracy did not improve from 0.89038
Epoch 100/100
557/557 [==============================] - 106s 190ms/step - loss: 0.0321 - accuracy: 0.985
1 - val_loss: 0.4900 - val_accuracy: 0.8753

Epoch 00100: val_accuracy did not improve from 0.89038
```

In [15]:

```
scores = model.evaluate(X_test, y_test, verbose=1)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
274/274 [==============================] - 13s 49ms/step - loss: 0.4900 - accuracy: 0.8753
Accuracy: 87.53%
```

In [16]:

```
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

In [17]:

```
y_pred = model.predict(X_test)

total = 0
accurate = 0
accurateindex = []
wrongindex = []

for i in range(len(y_pred)):
    if np.argmax(y_pred[i]) == np.argmax(y_test[i]):
        accurate += 1
        accurateindex.append(i)
    else:
        wrongindex.append(i)

    total += 1

print('Total-test-data;', total, '\taccurately-predicted-data:', accurate, '\t wrongly-predict
ed-data: ', total - accurate)

print('Accuracy:', round(accurate/total*100, 3), '%')
```

```
Total-test-data; 8767   accurately-predicted-data: 7674         wrongly-predicted-data:  1
093
Accuracy: 87.533 %
```

In [18]:

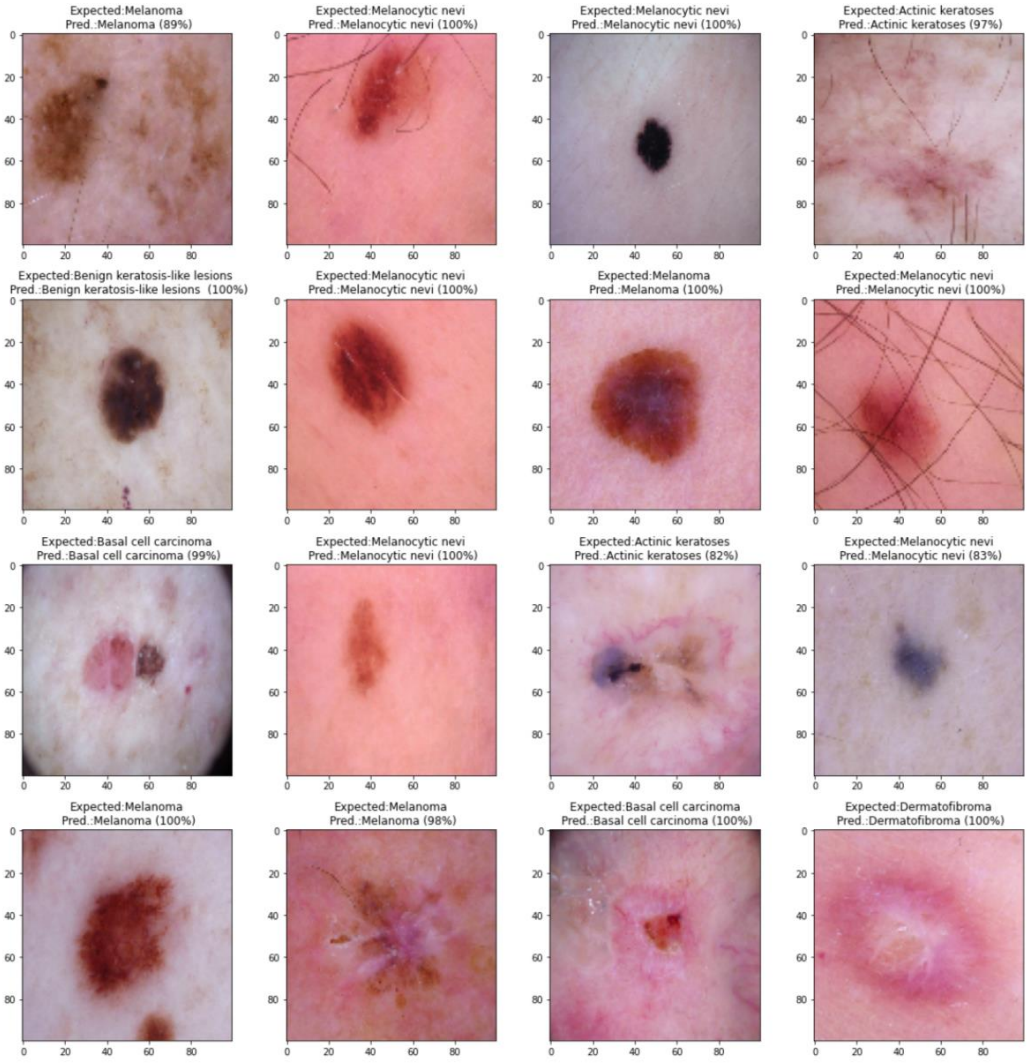
```
best_model = load_model('model.h5')

# compute predictions
y_pred_prob = np.around(best_model.predict(X_test),3)
y_pred = np.argmax(y_pred_prob,axis=1)

y_test2 = np.argmax(y_test,axis=1)
```

In [19]:

```
plt.figure(figsize=(16,16))
for i in range(16):
    plt.subplot(4,4,i+1)
    index = i+100
    plt.imshow(X_test[index,:,:,::-1])
    label_exp = lesion_names[y_test2[index]]  #expected label
    label_pred = lesion_names[y_pred[index]]  #predicted label
    label_pred_prob = round(np.max(y_pred_prob[index])*100)
    plt.title('Expected:'+str(label_exp)+'\n Pred.:'+str(label_pred)+' ('+str(label_pred_prob)
+'%)')
plt.ylabel('')
plt.tight_layout()
plt.savefig('final_figure.png',dpi=300)
plt.show()
```



In [20]:

```
print('Accuracy for label equal to 0')
print(np.mean(y_test2[y_test2 == 0] == y_pred[y_test2 == 0]))

print('Accuracy for label different from 0')
print(np.mean(y_test2[y_test2 != 0] == y_pred[y_test2 != 0]))
```

```
Accuracy for label equal to 0
0.8820605512878446
Accuracy for label different from 0
0.8931949954226427
```

In [21]:

```
acc_tot= []

for i in range(7):
    acc_parz = round(np.mean(y_test2[y_test2 == i] == y_pred[y_test2 == i]),2)
    lab_parz = lesion_names[i]
    print('accuracy for',lab_parz,'=',acc_parz)
    acc_tot.append(acc_parz)
```

```
accuracy for Melanocytic nevi = 0.88
accuracy for Melanoma = 0.9
accuracy for Benign keratosis-like lesions  = 0.92
accuracy for Basal cell carcinoma = 0.84
accuracy for Actinic keratoses = 0.79
accuracy for Vascular lesions = 0.98
accuracy for Dermatofibroma = 0.98
```

In [22]:

```
acc_tot = np.array(acc_tot)
freq = np.unique(y_test2,return_counts=True)[1]

np.sum(acc_tot*freq)/np.sum(freq)
```

Out[22]:

```
0.8894753051214782
```

8.RESULTS

In the wake of preparing the individual DCNNs on the expanded dataset, we have formed outfits from them dependent on every one of the models depicted in and the weighted ones to build the general precision of characterization. We have examined various combination models like this. After their extensive assessment, we have chosen the most exact one for this undertaking; for culmination individual, DCNN exhibitions have checks likewise.

Since the elaborate DCNN's apply as proper skin injury models, we have considered them best-in-class arrangements and remembered them for the quantitative correlation. Along these lines, for a reasonable far-reaching assessment, we have considered their last individual exactnesses after preparing them on the equivalent dataset portrayed with the uniform preparing boundary settings.

Every individual DCNN and their groups have assessed the test set's general score determined as the joint space under the recipient working trademark bend relating to the seven diseases characterization results. As regular execution measures, precision, affectability, and explicitness have also determined the certainty accuracy of87.53%. For the far-reaching visual assessment, the beneficiary working trademark bends the DCNNs and their group's plot.

In [16]:

```
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

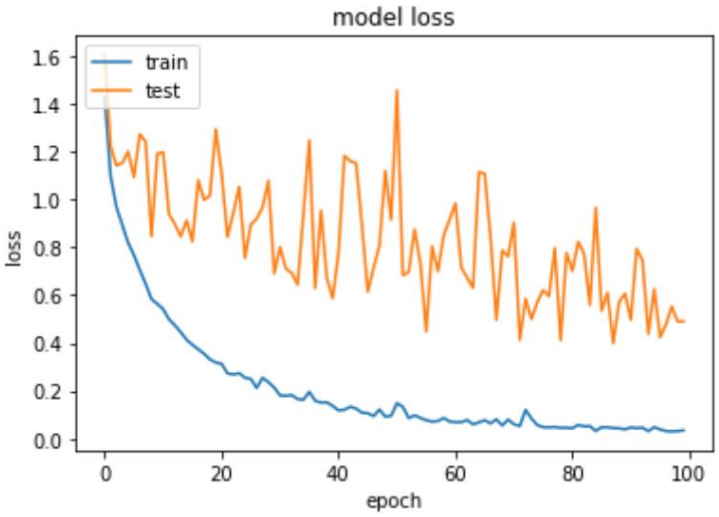


Fig. 8.1 Model Loss Graph

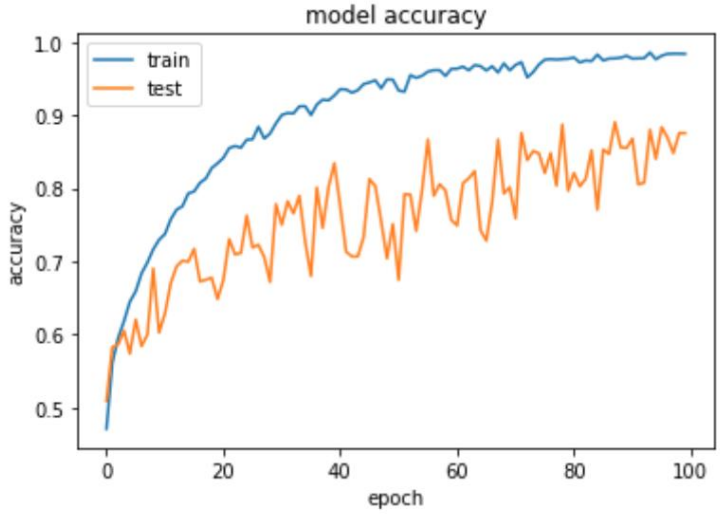


Fig. 8.2 Model Accuracy Graph

9.CONCLUSION

The use of DCNNs for sore skin identification is promising; nonetheless, the absence of enormous clarified datasets to prepare their models is as yet an obstruction to create techniques appropriate for clinical application. This project has explored the conceivable outcomes of making groups of profound neural organizations raise order exactness by joining their designs to profit by their qualities while beating their shortcomings when the quantity of explained pictures accessible for preparing is inadequate.

The main viable inspiration was to build up a programmed technique to characterize skin injury pictures as Benign keratosis, actinic keratosis, melanocytic nevi, melanoma, Basal cell carcinoma, dermafibroma or vascular lesions. In this examination, we utilized a picture set that contained 10015 pictures for preparing, out of which 33% pictures for testing. The augmentation of the preparation set extra pictures not thought of because we have tended to upgrade the exhibition of the CNNs in another manner. To be specific, we have proposed the thought of making outfits of them. We have discovered that if we weigh the DCNNs properly, let them cast a ballot to just one class, and decide the last class name of the information picture dependent on the amount of the most extreme certainty levels, and then this gathering beats the precision of the individual DCNNs in this grouping task.

We note that the proposed gathering-based DCNN system is secluded; that is, it tends to expand utilizing different DCNN organizations, and it very well may be tweaked for genuine use. As such, the system can set as it ought not to reason pointless concern for the patient or miss early severe injuries that are simpler to fix effectively. For instance, the system's yield should support affectability to avoid missing any possible diseases, to the detriment of bogus positives that a routine visit might identify to the dermatologist. Then again, patients at great danger for disease could screen themselves, such as utilizing their cell phones; for this situation, particularity should support. In the two situations, the proposed electronic structure ought to adjust to the objective of its application.

10.REFERENCES

1. Skin Lesion Segmentation with Improved Convolutional Neural Network - basepaper name.
   URL:https://doi.org/10.1007/s10278-020-00343-z

2. Skin Lesion data Set:
   URL:https://www.kaggle.com/kmader/skin-cancer-mnist-ham10000

3. Skin Lesion Classification with ensembles of DCNN
   URL: https://www.sciencedirect.com/science/article/pii/S1532046418301618

4. Skin Lesion Analysis by multi target DNN
   URL: https://pubmed.ncbi.nlm.nih.gov/30440620/

5. Al-Masni MA, Al-Antari MA, Choi MT, Han SM, Kim TS: Skin lesion segmentation in dermoscopy images via deep full resolution convolutional networks. Comput Methods Programs Biomed 162: 221-231,2018

6. SiegelRL,MillerKD,JemalA:Cancerstatistics,2019.CACancer J Clin 69(1):7-34,2019