Project Report Format

1. **INTRODUCTION**
    1. Project Overview
    2. Purpose
2. **LITERATURE SURVEY**
    1. Existing problem
    2. References
    3. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
    1. Empathy Map Canvas
    2. Ideation & Brainstorming
    3. Proposed Solution
    4. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
    1. Functional requirement
    2. Non-Functional requirements
5. **PROJECT DESIGN**
    1. Data Flow Diagrams
    2. Solution & Technical Architecture
    3. User Stories
6. **PROJECT PLANNING & SCHEDULING**
    1. Sprint Planning & Estimation
    2. Sprint Delivery Schedule
    3. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    1. Feature 1
    2. Feature 2
    3. Database Schema (if Applicable)
8. **TESTING**
    1. Test Cases
    2. User Acceptance Testing
9. **RESULTS**
    1. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code

GitHub & Project Demo Link

TITLE:

AI-based localization and classification of skin disease with erythema

1.Introduction

1.Project Overview:

We will have the online website to upload the affected skin area the result will be displayed with the disease name and the solution. We will have the online website to upload the affected skin area the result will be displayed with the disease name and the solution.Then the datas will be stored in the ibm cloud.We have used CNN and keras as backend to analayse the model performance.We have used HAM10000dataset.It consist of 10015 dermatoscopic images which are released as a training set for academic machine learning purposes and are publicly available through the ISIC archive.

2.Purpose:

Now a day's people are suffering from skin diseases, More than 125 million people suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades especially Melanoma is most diversifying skin cancer. If skin diseases are not treated at an earlier stage, then it may lead to complications in the body including spreading of the infection from one individual to the other. The skin diseases can be prevented by investigating the infected region at an early stage. The characteristic of the skin images is diversified so that it is a challenging job to devise an efficient and robust algorithm for automatic detection of skin disease and its severity. Skin tone and skin colour play an important role in skin disease detection. Colour and coarseness of skin are visually different. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.

To overcome the above problem we are building a model which is used for the prevention and early detection of skin cancer, psoriasis. Basically, skin disease diagnosis depends on the different characteristics like colour, shape, texture etc. Here the person can capture the images of skin and then the image will be sent the trained model. The model analyses the image and detect whether the person is having skin disease or not.

2. LITERATURE SURVEY:

1.TITLE: Detection of Skin Diseases, **AUTHOR:** Md. Nazmul Hossen, Vijayakumari Panneerselvam/ 2022

PROPOSED SOLUTION:

An image augmentation strategy was followed to enlarge the dataset and make the model more general. It is used to detect the skin cancer diseases.

PROS AND LIMITATIONS:

The CNN-based skin disease classification merged with the federated learning approach is a breathtaking concept to classify human skin diseases while ensuring data security.

2.TITLE: A Simplified Approach for Melanoma Skin Disease Identification, **AUTHOR:** G. Glorindal,S. Arun Mozhiselvi,T. Ananth Kumar,K. Kumaran/2021

PROPOSED SOLUTION:

An image processing approach with an easily driven Application Programmable Interface commonly known as API, has been proposed to diagnose skin diseases at their earlier stages.

PROS AND LIMITATIONS:

The image processing follows preprocessing, segmentation, feature extraction, and classification steps, which apply contrast stretching and median filter, Fuzzy C Means, Grey Level CoOccurrence Matrix(GLCM), and Gabor filter, Support Vector Machine .

REFERENCES:

https://www.researchgate.net/publication/354161902_Intelligent_System_for_Skin_Disease_Prediction_using_Machine_Learning_Intelligent_System_for_Skin_Disease_Prediction_using_Machine_Learning

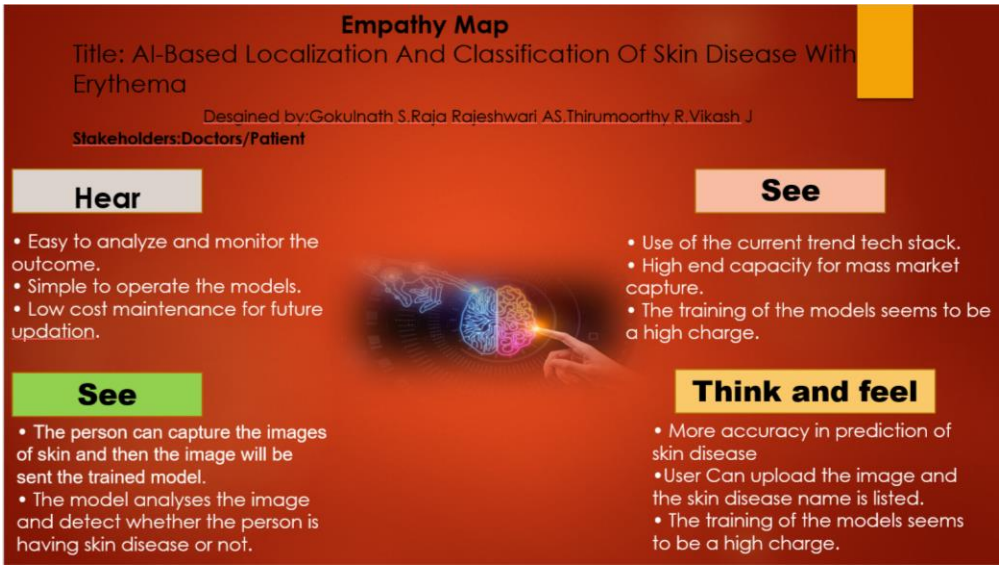
https://www.sciencedirect.com/science/article/pii/S2352914819300838

https://iopscience.iop.org/article/10.1088/1742-6596/1724/1/012048

https://link.springer.com/article/10.1007/s00500-022-07435-8

14. **3. IDEATION & PROPOSED SOLUTION:**

1. Empathy Map Canvas



2.Proposed Solution:

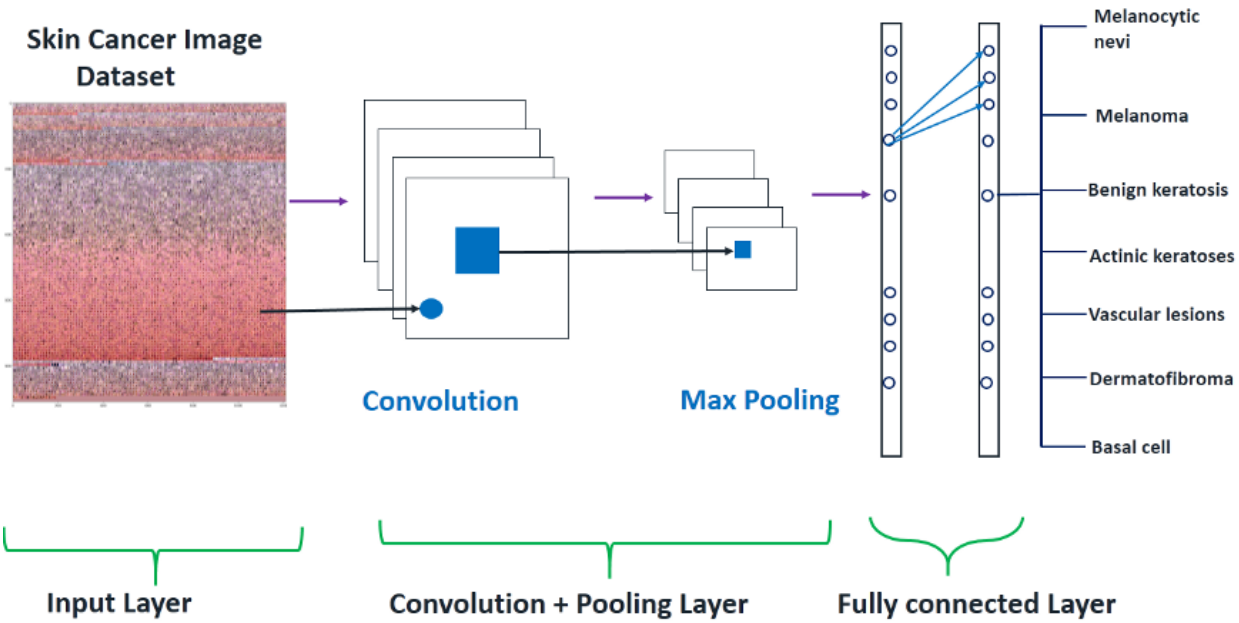
We will have the online website to upload the affected skin area the result will be displayed with the disease name and the solution. We will have the online website to upload the affected skin area the result will be displayed with the disease name and the solution.Then the datas will be stored in the ibm cloud.We have used CNN and keras as backend to analayse the model performance.We have used HAM10000dataset.It consist of 10015 dermatoscopic images which are released as a training set for academic machine learning purposes and are publicly available through the ISIC archive.

LIBRARIES USED:

Numpy

Keras

Tensorflow
Pandas
Matplotlib
Numpy
Flask
seaborn



3.**Problem Solution fit:**

Skin disease can appear in virtually any part of body and there is a lack of data required to form an association between the probability of a skin disease based on the body part. A Solution model used for the prevention and early detection of skin cancer and psoriasis by image analyses to detect whether the person is having skin disease or not. The location of the disease that is present in an image and improved performance by CNN model to focus on particular subsections of the images.

4. REQUIREMENT ANALYSIS

Functional requirements:

Image Acquisition, Pre-processing Steps such as Colour gradient generator on an image , Cropping and isolating region of interest and Thresholding and Clustering on image, Visual feature extraction, System Training YOLO Model for Skin disease

classification with deep learning and CNN, Separate access of application for admin,Diagnosis of Skin disease and Data retrieval and Data manipulation.

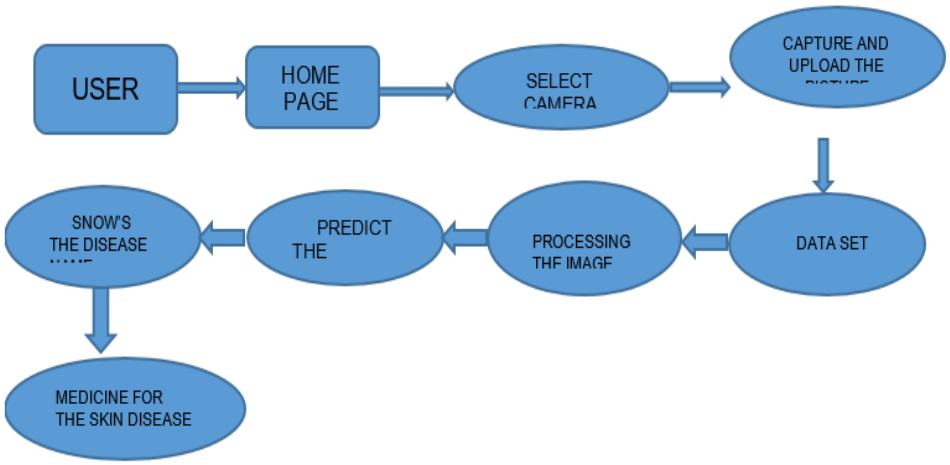
Non-Functional requirements:

SoftwareQualityAttributes,Prediction,Accuracy,usability,security,Reliability,Performance,Availability,Scalability.

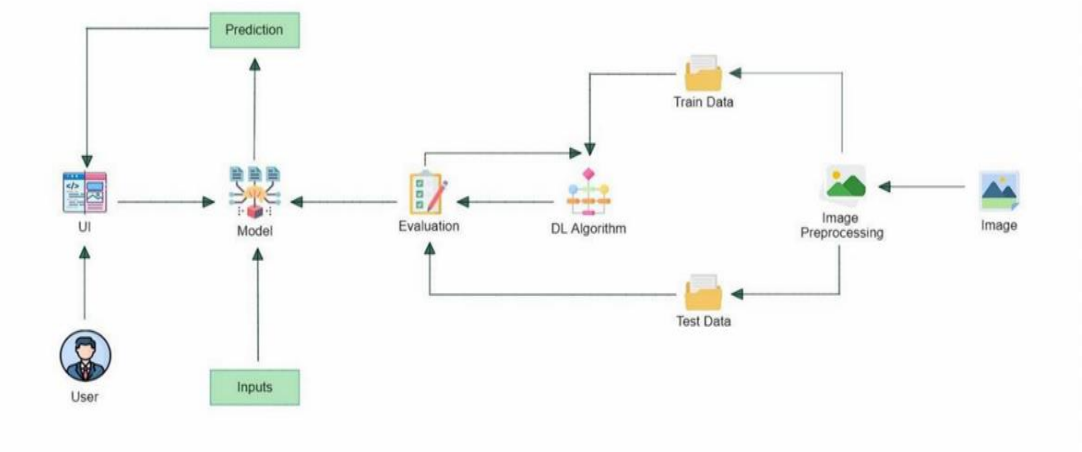
5. PROJECT DESIGN:

Data Flow Diagrams:

1.



Solution & Technical Architecture:



6.Project planning and Scheduling:

sprint planning and estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Data Preparation	USN-1	Preparing model architecture.	10	High	Thirumoorthy R,Rajarajeshwari AS
Sprint 1	Data Modelling	USN-2	Collecting skin disease dataset and pre-processing it	10	Medium	Vikash j,Gokulnath S
Sprint 2	Model Building	USN-4	Model Building And model Evaluation	20	High	Thirumoorthy R,Rajarajeshwari AS
Sprint 3	Application building	USN-5	Skin disease prediction webpage creation using	10	High	Vikash j,Gokulnath S

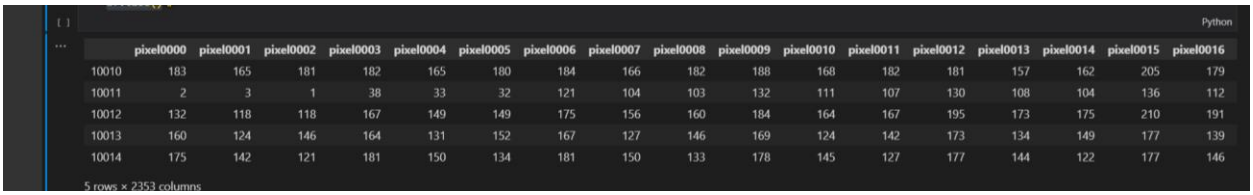
			HTML, CSS, JS and			
Sprint 3	Dashboard	USN-6	connecting it with flask and backend	10	High	Thirumoorthy R,Rajarajeshwari AS
Sprint 4	Deploying the model on IBM Cloud	USN-7	Using flask and deploy model finally in IBM cloud using IBM storage and Watson Studio	20	High	Thirumoorthy R,Rajarajeshwari AS

7.CODING & SOLUTIONING (Explain the features added in the project along with code)

```
Importing Essential Libraries
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/96.0.4664.110 Safari/537.36" --header="Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q
=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-
US,en;q=0.9" --header="Referer: https://www.kaggle.com/"
"https://storage.googleapis.com/kaggle-data-
sets/54339/104884/compressed/hmnist_28_28_RGB.csv.zip?X-Goog-Algorithm=GOOG4-RSA-
SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-
161607.iam.gserviceaccount.com%2F20220105%2Fauto%2Fstorage%2Fgoog4_request&X-
Goog-Date=20220105T054322Z&X-Goog-Expires=259199&X-Goog-SignedHeaders=host&X-
Goog-
Signature=80620f7dba9840baa232aea442aebc1fe2926c0318402ae7381538eb2ca96c6aafff103
c8b554579da29c70267957bdc814fc23e0a74ab5ad09f0cb7709369a0a84e038fff173c659409e003
7c9d302977c39b5bf84ee1f7e397d28aea1500aa53d45afd02725873aa36082c69eed3ed7ae2bcb61
321b412d9401466462c4aea684392a71db67d00a01e699945c2065b3acdd32108a21fd6fde57ad44f
cb95e7bd1e75b5c7a23944e242d59815e4564f4da02eb39ef1b761ace726071509cca0656aafdf470
6b4e9d6d4ea5c3abf9116b7bd89534e1ba26037614a53be1baf41e3f6dd45c7cd57056cafd545d099
72e9d341be86f6eafcc4ef34afcb3ea304ce" -c -O 'hmnist_28_28_RGB.csv.zip'
```

```
Loading data and Making labels
```

```
df=pd.read_csv(path)
df.tail()
```



	pixel0000	pixel0001	pixel0002	pixel0003	pixel0004	pixel0005	pixel0006	pixel0007	pixel0008	pixel0009	pixel0010	pixel0011	pixel0012	pixel0013	pixel0014	pixel0015	pixel0016
10010	183	165	181	182	165	180	184	166	182	188	168	182	181	157	162	205	179
10011	2	3	1	38	33	32	121	104	103	132	111	107	130	108	104	136	112
10012	132	118	118	167	149	149	175	156	160	184	164	167	195	173	175	210	191
10013	160	124	146	164	131	152	167	127	146	169	124	142	173	134	149	177	139
10014	175	142	121	181	150	134	181	150	133	178	145	127	177	144	122	177	146

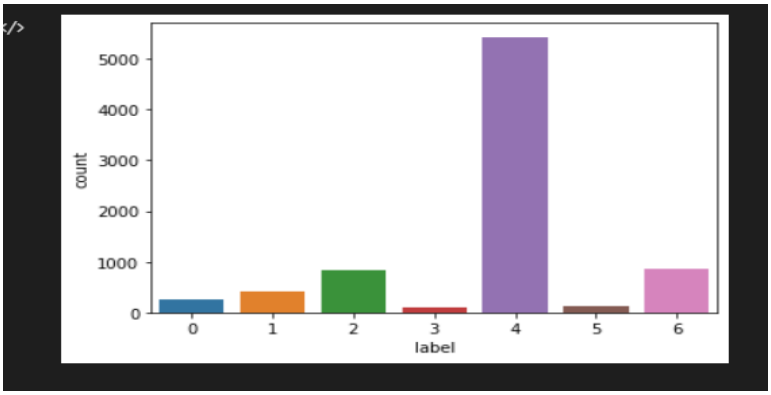
5 rows × 2353 columns

```
Train Test Split
fractions=np.array([0.8,0.2])

df=df.sample(frac=1)

train_set, test_set = np.array_split(df, (fractions[:-1].cumsum() *
len(df)).astype(int))
```

```
# reference: https://www.kaggle.com/kmader/skin-cancer-mnist-
ham10000/discussion/183083
classes={
    0:('akiec', 'actinic keratoses and intraepithelial carcinomae'),

    1:('bcc' , 'basal cell carcinoma'),

    2:('bkl', 'benign keratosis-like lesions'),

    3:('df', 'dermatofibroma'),

    4:('nv', ' melanocytic nevi'),

    5:('vasc', ' pyogenic granulomas and hemorrhage'),

    6:('mel', 'melanoma'),
}
```

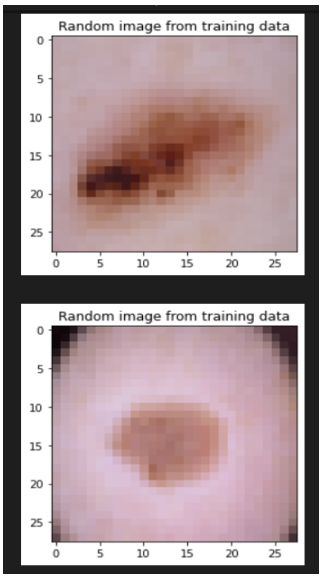
```
y_train=train_set['label']

x_train=train_set.drop(columns=['label'])

y_test=test_set['label']

x_test=test_set.drop(columns=['label'])

columns=list(x_train)
```

```
Exploratory Data Analysis and Preprocessing
import seaborn as sns

sns.countplot(train_set['label'])
```




```
from imblearn.over_sampling import RandomOverSampler

oversample = RandomOverSampler()

x_train,y_train  = oversample.fit_resample(x_train,y_train)
```

```
import matplotlib.pyplot as plt
import random

num=random.randint(0,8000)
x_train=np.array(x_train, dtype=np.uint8).reshape(-1,28,28,3)

plt.imshow(x_train[num].reshape(28,28,3))
plt.title("Random image from training data")
plt.show()
num=random.randint(0,8000)
plt.imshow(x_train[num].reshape(28,28,3))
plt.title("Random image from training data")
plt.show()

num=random.randint(0,8000)
plt.imshow(x_train[num].reshape(28,28,3))
plt.title("Random image from training data")
plt.show()
```



Model Building (CNN)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D
import tensorflow as tf


%time

model = Sequential()

model.add(Conv2D(16,
                 kernel_size = (3,3),
                 input_shape = (28, 28, 3),
                 activation = 'relu',
                 padding = 'same'))

model.add(MaxPool2D(pool_size = (2,2)))
model.add(tf.keras.layers.BatchNormalization())

model.add(Conv2D(32,
                 kernel_size = (3,3),
                 activation = 'relu'))

model.add(Conv2D(64,
                 kernel_size = (3,3),
                 activation = 'relu'))

model.add(MaxPool2D(pool_size = (2,2)))
```

```
model.add(tf.keras.layers.BatchNormalization())

model.add(Conv2D(128,
                 kernel_size = (3,3),
                 activation = 'relu'))

model.add(Conv2D(256,
                 kernel_size = (3,3),
                 activation = 'relu'))

model.add(Flatten())
model.add(tf.keras.layers.Dropout(0.2))
model.add(Dense(256,activation='relu'))

model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Dropout(0.2))
model.add(Dense(128,activation='relu'))

model.add(tf.keras.layers.BatchNormalization())
model.add(Dense(64,activation='relu'))

model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Dropout(0.2))
model.add(Dense(32,activation='relu'))

model.add(tf.keras.layers.BatchNormalization())
model.add(Dense(7,activation='softmax'))

model.summary()
```

```
Output exceeds the size limit. Open the full output data in a text editor
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 6.44 µs
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 28, 28, 16)        448

 max_pooling2d_4 (MaxPooling  (None, 14, 14, 16)       0
 2D)

 batch_normalization_12 (Bat  (None, 14, 14, 16)       64
 chNormalization)

 conv2d_11 (Conv2D)          (None, 12, 12, 32)        4640

 conv2d_12 (Conv2D)          (None, 10, 10, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 batch_normalization_13 (Bat  (None, 5, 5, 64)         256
 chNormalization)

 conv2d_13 (Conv2D)          (None, 3, 3, 128)         73856
...
Total params: 504,103
Trainable params: 502,983
```

8.TESTING

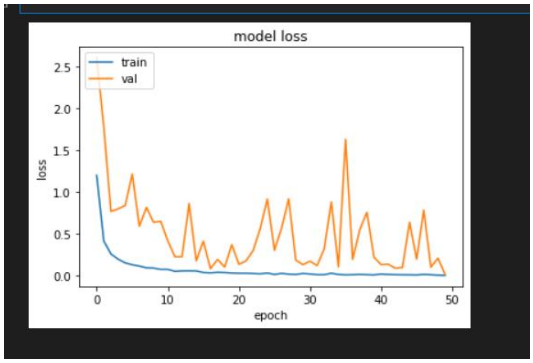
```
from datetime import datetime

start_time = datetime.now()

history = model.fit(x_train,
                    y_train,
                    validation_split=0.2,
                    batch_size = 128,
                    epochs = 50,
                    shuffle=True,
                    callbacks=[callback])

end_time = datetime.now()

print('Duration: {}'.format(end_time - start_time))
```

```
Epoch 1/50
238/238 [==============================] - ETA: 0s - loss: 1.1990 - accuracy: 0.5744WARNING:tensorflow:Can save best model only with val_acc available, skipping.
238/238 [==============================] - 8s 24ms/step - loss: 1.1990 - accuracy: 0.5744 - val_loss: 2.6021 - val_accuracy: 0.0402
Epoch 2/50
236/238 [============================>.] - ETA: 0s - loss: 0.4169 - accuracy: 0.8618WARNING:tensorflow:Can save best model only with val_acc available, skipping.
238/238 [==============================] - 5s 20ms/step - loss: 0.4168 - accuracy: 0.8618 - val_loss: 1.7786 - val_accuracy: 0.3873
Epoch 3/50
236/238 [============================>.] - ETA: 0s - loss: 0.2618 - accuracy: 0.9108WARNING:tensorflow:Can save best model only with val_acc available, skipping.
238/238 [==============================] - 4s 19ms/step - loss: 0.2621 - accuracy: 0.9108 - val_loss: 0.7704 - val_accuracy: 0.6166
Epoch 4/50
237/238 [============================>.] - ETA: 0s - loss: 0.2004 - accuracy: 0.9324WARNING:tensorflow:Can save best model only with val_acc available, skipping.
238/238 [==============================] - 4s 19ms/step - loss: 0.2003 - accuracy: 0.9324 - val_loss: 0.7994 - val_accuracy: 0.6335
Epoch 5/50
236/238 [============================>.] - ETA: 0s - loss: 0.1560 - accuracy: 0.9472WARNING:tensorflow:Can save best model only with val_acc available, skipping.
238/238 [==============================] - 5s 20ms/step - loss: 0.1559 - accuracy: 0.9473 - val_loss: 0.8395 - val_accuracy: 0.5850
Epoch 6/50
236/238 [============================>.] - ETA: 0s - loss: 0.1329 - accuracy: 0.9538WARNING:tensorflow:Can save best model only with val_acc available, skipping.
238/238 [==============================] - 4s 19ms/step - loss: 0.1331 - accuracy: 0.9538 - val_loss: 1.2162 - val_accuracy: 0.5410
Epoch 7/50
236/238 [============================>.] - ETA: 0s - loss: 0.1176 - accuracy: 0.9601WARNING:tensorflow:Can save best model only with val_acc available, skipping.
238/238 [==============================] - 4s 19ms/step - loss: 0.1175 - accuracy: 0.9601 - val_loss: 0.5929 - val_accuracy: 0.7492
Epoch 8/50
237/238 [============================>.] - ETA: 0s - loss: 0.0961 - accuracy: 0.9653WARNING:tensorflow:Can save best model only with val_acc available, skipping.
238/238 [==============================] - 5s 20ms/step - loss: 0.0963 - accuracy: 0.9653 - val_loss: 0.8171 - val_accuracy: 0.6011
Epoch 9/50
...
Epoch 50/50
238/238 [==============================] - ETA: 0s - loss: 0.0066 - accuracy: 0.9981WARNING:tensorflow:Can save best model only with val_acc available, skipping.
238/238 [==============================] - 5s 21ms/step - loss: 0.0066 - accuracy: 0.9981 - val_loss: 0.0176 - val_accuracy: 0.9961
Duration: 0:04:24.568061
```

Model Evaluation

```python
#plot of accuracy vs epoch

plt.plot(history.history['accuracy'])

plt.plot(history.history['val_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'val'], loc='upper left')

plt.show()
#plot of loss vs epoch

plt.plot(history.history['loss'])

plt.plot(history.history['val_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')
```

```
plt.legend(['train', 'val'], loc='upper left')

plt.show()
```



2. User Acceptance Testing:

```
15.from sklearn.metrics import confusion_matrix
16.
17.y_pred = model.predict(x_test)
18.y_pred = np.argmax(y_pred, axis=1)
19.
20.conf_mat = confusion_matrix(y_test, y_pred)
21.
```

```
print(conf_mat)
```

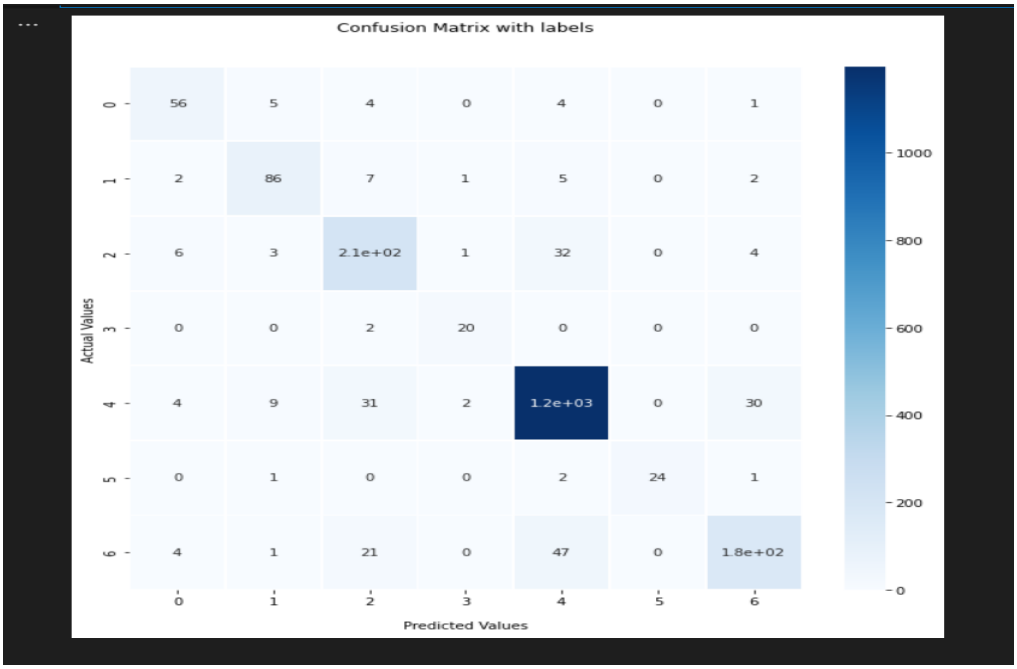
```
[[  56    5    4    0    4    0    1]
 [   2   86    7    1    5    0    2]
 [   6    3  211    1   32    0    4]
 [   0    0    2   20    0    0    0]
 [   4    9   31    2 1198    0   30]
 [   0    1    0    0    2   24    1]
 [   4    1   21    0   47    0  176]]
```

9.RESULTS

1. Performance Metrics

```
22.import seaborn as sns
23.import matplotlib.pyplot as plt
24.fig, ax = plt.subplots(figsize=(10,10))
25.
```

```
ax = sns.heatmap(conf_mat, annot=True, cmap='Blues', linewidths=.9, ax=ax)

ax.set_title('Confusion Matrix with labels\n\n');

ax.set_xlabel('\nPredicted Values')

ax.set_ylabel('Actual Values ');

ax.xaxis.set_ticklabels(['0','1','2','3','4','5','6'])

ax.yaxis.set_ticklabels(['0','1','2','3','4','5','6'])

plt.show()
```




```
#https://pillow.readthedocs.io/en/stable/

import PIL

image=PIL.Image.open('test.jpg')

image=image.resize((28,28))

img=x_test[1]

img=np.array(image).reshape(-1,28,28,3)
```

```
result=model.predict(img)

print(result[0])

result=result.tolist()

max_prob=max(result[0])

class_ind=result[0].index(max_prob)

print(classes[class_ind])

RESULT:
[9.4849083e-06 2.5722728e-05 4.0849736e-05 5.5863543e-06 9.9334717e-01
1.0107847e-05 6.5611666e-03] ('nv', ' melanocytic nevi')
```

Advantages:

- High Enhancement.
- No therapeutic limitation.
- Easily incorporate depot,Delivery not majorly affected by diseased state of skin.

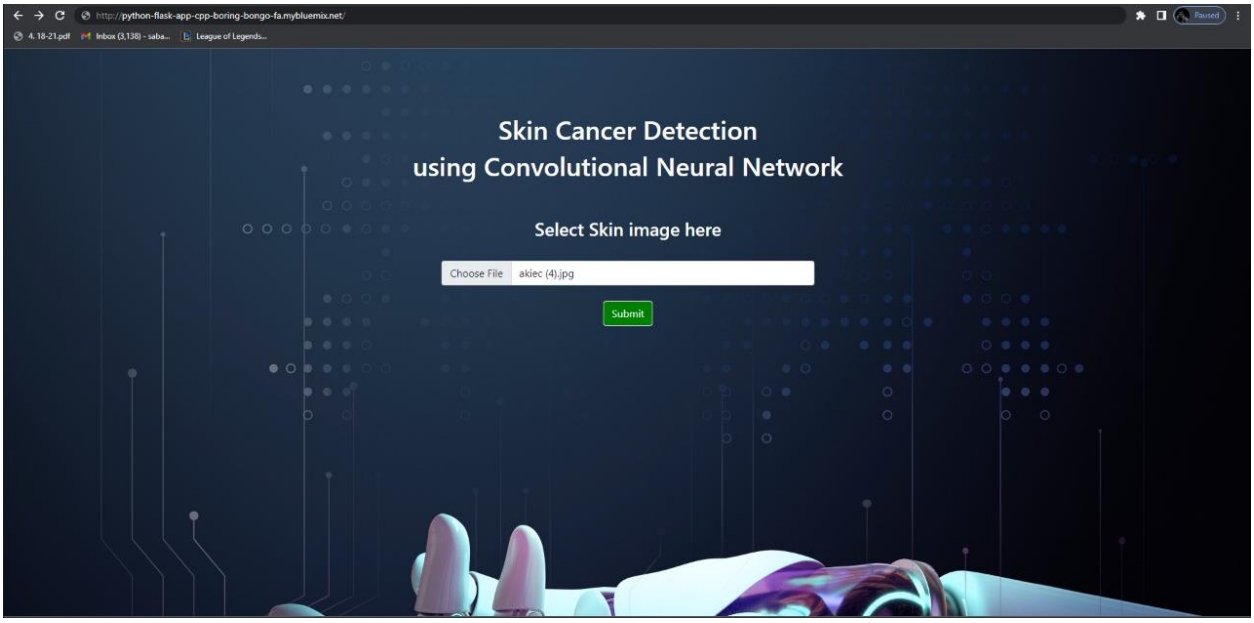
Disadvantages:

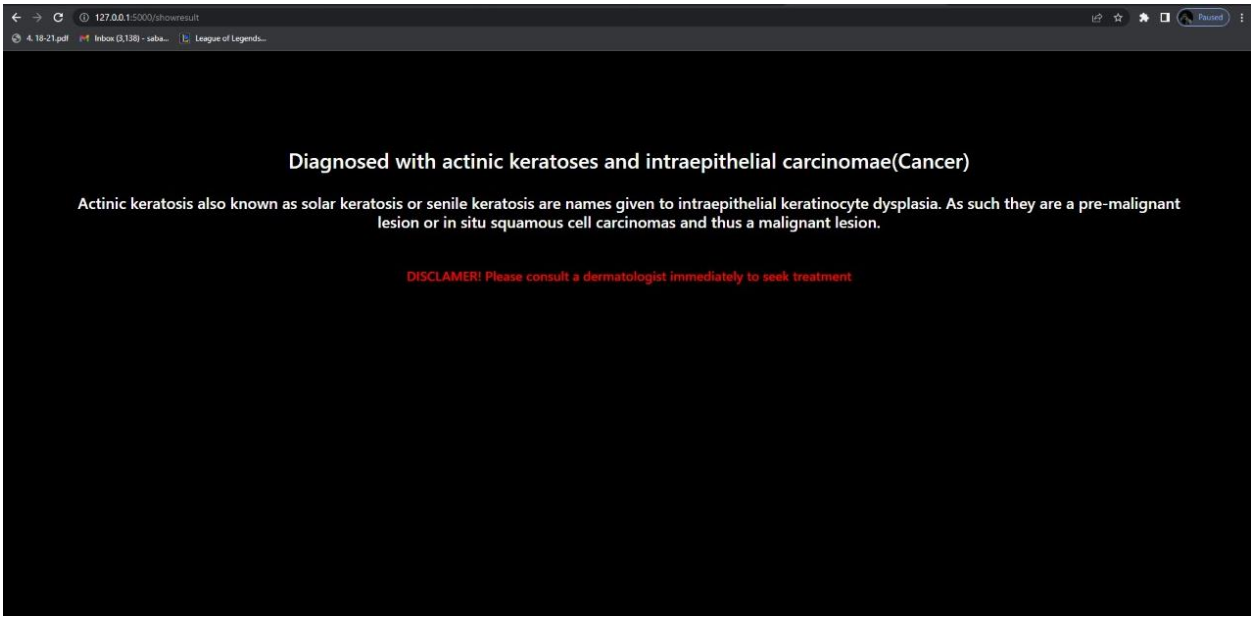
- High cost.
- Security concern.

10.Conclusion:

Skin diseases are a bit like the common cold.They vary enormously from mild conditions which may affect only the appearance of the skin to severe disease which are totally incapacitating.The degree of treatment required or even sought varies accordinglyIn addition, we have shown that current state-of-the-art CNN models can outperform models created by previous research, through proper data pre-processing, self-supervised learning, transfer learning, and special CNN architecture techniques. Furthermore, with accurate segmentation, we gain knowledge of the location of the disease, which is useful in the pre-processing of data used in classification, as it allows the CNN model to focus on the area of interest. Lastly, unlike previous studies, our method provides a solution to classify multiple diseases within a single image. With higher quality and a larger quantity of data, it will be viable to use state-of-the-art models to enable the use of CAD in the field of dermatology.

OUTPUT SCREENSHOTS:





11.Future Scope:

In future, this machine learning model may bind with a variouswebsite which can provide real-time data for
skin disease prediction. Also, we may add large historical data on skin diseasewhich can help to improve
theaccuracy of the machine learning model. We can build an android app as a userinterface for interactingwith the user. For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates, and train it on clusters of data ratherthan the whole dataset.

GitHub & Project Demo Link:

https://github.com/IBM-EPBL/IBM-Project-25967-1659977818

Demo link:

https://drive.google.com/file/d/19VqW1GPkF09QkPlx4Yl4IhIW87BCOWPR/view?usp=sharing