International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 5 Issue 6, September-October 2021 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Skin Cancer Detection using Image-Processing in Real-Time

Sunami Dasgupta[1], Soham Das[1], Sayani Hazra Pal[2]

[1]Student, D.A.V. Public School, Kolkata, West Bengal, India
[2]Instructor, D.A.V. Public School, Kolkata, West Bengal, India

ABSTRACT

Machine learning is a fascinating topic; it's astonishing how a small change in the evaluation values may result in an unfathomable number of outcomes. The goal of this study is to develop a model that uses image processing to identify skin cancer. We will later use the model in real-life through an android application.

KEYWORDS: *Image Processing; Skin-Cancer Detection; Machine Learning; Artificial Intelligence*

How to cite this paper: Sunami Dasgupta | Soham Das | Sayani Hazra Pal "Skin Cancer Detection using Image-Processing in Real-Time" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-5 | Issue-6, October 2021, pp.271-280, URL: www.ijtsrd.com/papers/ijtsrd46384.pdf

IJTSRD46384

Copyright © 2021 by author(s) and International Journal of Trend in Scientific Research and Development Journal. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0) (http://creativecommons.org/licenses/by/4.0)

INTRODUCTION

What is image processing and why are we using it?

Image processing is a technique for performing operations on an image in order to improve it or extract useful information from it. It's a type of signal processing in which the input is an image and the output is either an image or its characteristics/features.

Image processing mainly include the following steps:

1. Importing the image via image acquisition tools;
2. Analysing and manipulating the image;
3. Output in which result can be altered image or a report which is based on analysing that image.

Image as a Matrix

As we know, images are represented in rows and columns. The syntax for representing images is as follows:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,N-1) \\ \vdots & \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & f(M-1,2) & \dots & f(M-1,N-1) \end{bmatrix}$$

In MATLAB:

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{bmatrix}$$

Phases of Image Processing

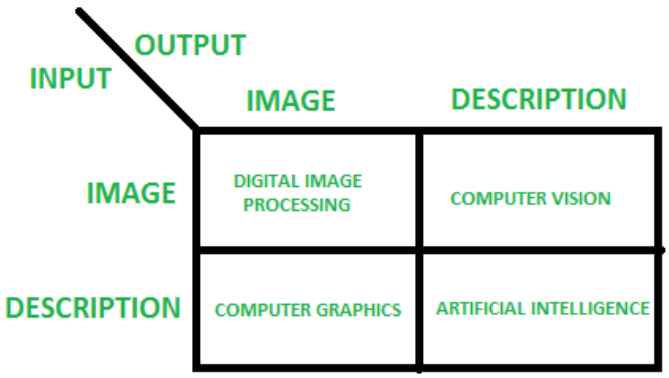
There are different phases to import and extract data from an Image. It consists of 10 major steps. They are as follows:

1. ACQUISITION– It could be as fundamental as being given a digital image to work with. The main work involves:

A. Scaling.
B. Colour conversion (RGB to Gray or vice-versa).

2. IMAGE ENHANCEMENT– It is one of the most basic and appealing aspects of Image Processing, and it is also used to extract some hidden elements from an image. It is subjective.
3. IMAGE RESTORATION– This likewise has to do with making an image appealing, but it is more objective (Restoration is based on mathematical or probabilistic model or image degradation).
4. COLOR IMAGE PROCESSING– This section covers pseudo-colour and full colour image processing, as well as colour models that can be used in digital image processing.
5. WAVELETS AND MULTI-RESOLUTION PROCESSING– This is the foundation for portraying images in a variety of ways.
6. IMAGE COMPRESSION—This procedure necessitates the creation of various functions. It primarily concerns image size or resolution.
7. MORPHOLOGICAL PROCESSING-It deals with tools for extracting picture components that can be used to represent and describe shape.
8. SEGMENTATION PROCEDURE-Partitioning an image into its constituent sections or objects is part of this procedure. The most demanding problem in Image Processing is autonomous segmentation.
9. REPRESENTATION & DESCRIPTION-It comes after the output of the segmentation step; selecting a representation is only one part of the solution for converting raw data into processed data.
10. OBJECT DETECTION AND RECOGNITION-It is a process that assigns a label to an object based on its descriptor.

Overlapping Fields with Image Processing



1. According to block 1, if the input is an image and the output is an image, the process is known as digital image processing.
2. According to block 2, Computer Vision is defined as an input that is an image and an output that is some kind of information or description.
3. According to block 3, if input is some description or code and we get image as an output, then it is termed as Computer Graphics.
4. According to block 4, if input is description or some keywords or some code and we get description or some keywords as an output, then it is termed as Artificial Intelligence.

Skin Cancer Overview

Skin cancer, or the abnormal proliferation of skin cells, is most commonly found on sun-exposed skin. However, this frequent type of cancer can also develop on parts of your skin that aren't normally exposed to the sun.

Basal cell carcinoma, squamous cell carcinoma, and melanoma are the three most common kinds of skin cancer.

Limiting or avoiding ultraviolet (UV) radiation exposure can lower your risk of skin cancer. Skin cancer can be detected at an early stage if you check your skin for abnormal changes. Early skin cancer detection increases your chances of a successful skin cancer treatment.

1. Basal cell carcinoma - A kind of skin cancer known as basal cell carcinoma. Basal cell carcinoma starts in the basal cells, which are a type of skin cell that creates new skin cells when the old ones die. Basal cell carcinoma often appears as a slightly transparent bump on the skin, though it can take other forms. Basal cell carcinoma occurs most often on areas of the skin that are exposed to the sun, such as your head and neck.
2. Melanoma – It originates in the cells (melanocytes) that create melanin, the pigment that gives your skin its colour. It is the most dangerous type of skin cancer. Melanoma can also develop in the eyes and, in rare cases, inside the body, such as the nose or throat. Although the specific aetiology of all melanomas is unknown, ultraviolet (UV) radiation from the sun, tanning lights, and beds increases your risk of acquiring melanoma. Melanoma risk can be reduced by limiting your exposure to UV light.
3. Nonmelanoma skin cancer - It encompasses all skin cancers that aren't melanoma. Nonmelanoma

skin cancer encompasses a number of different forms of skin cancer, the most frequent of which are basal cell carcinoma and squamous cell carcinoma. Treatment for nonmelanoma skin cancer is determined by the type of malignancy. The most common therapy for skin cancer is surgery to remove the cancerous cells.

PREREQUISITES

1. Dataset for Training.
2. Dataset for Testing.
3. Machine learning model (that takes in digital Images as input and predicts cancer cells).

Dataset

We used the dataset provided by SIIM-ISIC Melanoma Classification.

The images are provided in DICOM format. This can be accessed using commonly-available libraries like PyDicom, and contains both image and metadata. It is a commonly used medical imaging data format.

Images are also provided in JPEG and TF Record format (in the jpeg and TF records directories, respectively). Images in TF Record format have been resized to a uniform 1024x1024.

EXPERIMENTATION

TensorFlow and Keras are used to build and create a machine learning model.

1. Data Exploration

We created algorithms and models to classify between benign and malignant skin cancers using Convolutional Neural Networks. We use Colab for source code editing.

```
# Machine Learning
import tensorflow as tf
import keras
from keras import initializers
from keras import regularizers
from keras import constraints
from keras import backend as K
from keras.activations import elu
from keras.optimizers import Adam
from keras.models import Sequential
from keras.engine import Layer, InputSpec
from keras.utils.generic_utils import get_custom_objects
from keras.callbacks import Callback, EarlyStopping, ReduceLROnPlateau
from keras.layers import Dense, Conv2D, Flatten, GlobalAveragePooling2D, Dropout,MaxPoolin
g2D,BatchNormalization,GlobalMaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import cohen_kappa_score

# Specify title of our final model
SAVED_MODEL_NAME = 'effnet_modelB5.h5'
```

2. Seeding

The seed sets the starting number used to generate a sequence of random numbers it ensures that you get the same result if you start with that same seed each time you run the same process.

```
# Set seed for reproducability
seed = 1234
rn.seed(seed)
np.random.seed(seed)
tf.random.set_seed(seed)
os.environ['PYTHONHASHSEED'] = str(seed)

# For keeping time. GPU limit for this competition is set to ± 9 hours.
t_start = time.time()
```

3. Initializing Model

Model initialization is the process of determining the necessary model parameters such as the basic value, the trend value, and the seasonal indices for the selected forecast model.

```
# Initialize model
model = build_model()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Functi (None, 8, 8, 1280)        2257984
_________________________________________________________________
global_average_pooling2d (Gl (None, 1280)              0
_________________________________________________________________
dropout (Dropout)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 1)                 1281
=================================================================
Total params: 2,259,265
Trainable params: 2,225,153
Non-trainable params: 34,112
_________________________________________________________________
None
```

4. Loading the Data

We'll use the skin cancer dataset's binary classification of benign and malignant skin cancers. Following that, we must label.

In [7]:
```
import pandas as pd
df=pd.read_csv("../input/jpeg-melanoma-256x256/train.csv")
```

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

train, valid = train_test_split(df, test_size=0.2)
```

5. Dataset

Let take a look at our dataset.

In [9]:
```
train
```

Out[9]:

	image_name	patient_id	sex	age_approx	anatom_site_general_challenge	diagnosis	benign_malignant
17064	ISIC_5210074	IP_5698400	female	25.0	torso	nevus	benign
7157	ISIC_2236868	IP_8667520	male	60.0	torso	unknown	benign
3266	ISIC_1085358	IP_3650745	male	65.0	upper extremity	unknown	benign
7791	ISIC_2428554	IP_1476317	female	35.0	torso	nevus	benign
28672	ISIC_8659884	IP_4983809	male	70.0	upper extremity	unknown	benign
...	...	...	...	...	...	...	...
9449	ISIC_2929993	IP_1172238	female	50.0	lower extremity	nevus	benign
32399	ISIC_9785002	IP_5365464	female	70.0	torso	unknown	benign
17048	ISIC_5207738	IP_7489375	male	45.0	lower extremity	unknown	benign
23924	ISIC_7264251	IP_7160012	male	65.0	lower extremity	unknown	benign
27439	ISIC_8307569	IP_7688774	female	25.0	lower extremity	unknown	benign

26500 rows × 11 columns

6. Data Processing

We set up generators to read images from our source folders, then we transformed them to float32 tensors, and fed them to our network.

In our case, we'll change the pixel values to the [0, 1] range before pre-processing the images (all values are now in the [0, 255] range). As required by the networks, the input data must be scaled to 224x224 pixels as an input.

```
train_generator = train_datagen.flow_from_dataframe(
        dataframe=train,
        directory='../input/jpeg-melanoma-256x256/train/',
        x_col="image_name",
        y_col="target",
        target_size=(256, 256),
        batch_size=32,
        class_mode='binary')

validation_generator = test_datagen.flow_from_dataframe(
        dataframe=valid,
        directory='../input/jpeg-melanoma-256x256/train/',
        x_col="image_name",
        y_col="target",
        target_size=(256, 256),
        batch_size=16,
        class_mode='binary')
```

```
Found 26500 validated image filenames belonging to 2 classes.
Found 6626 validated image filenames belonging to 2 classes.
```

7. Building the Model

It's as simple as layering a linear classifier on top of the feature extractor with the Hub module. For speed, we start with a non-trainable feature extractor.

```
feature_extractor = hub.KerasLayer(MODULE_HANDLE,input_shape=IMAGE_SIZE+(3,), output
do_fine_tuning = False
if do_fine_tuning:
  feature_extractor.trainable = True
  for layer in base_model.layers[-30:]:
    layer.trainable =True

else:
  feature_extractor.trainable = False

print("Building model with", MODULE_HANDLE)
model = tf.keras.Sequential([
    feature_extractor,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(rate=0.2),
    tf.keras.layers.Dense(train_generator.num_classes, activation='softmax',
                          kernel_regularizer=tf.keras.regularizers.l2(0.0001))
])

model.summary()
```

```
Building model with https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/2
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer_10 (KerasLayer)  (None, 1280)              2257984
_________________________________________________________________
flatten_11 (Flatten)         (None, 1280)              0
_________________________________________________________________
dense_22 (Dense)             (None, 512)               655872
_________________________________________________________________
dropout_11 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_23 (Dense)             (None, 2)                 1026
=================================================================
Total params: 2,914,882
Trainable params: 656,898
Non-trainable params: 2,257,984
```

8. Training Model

We validated each step by training the model with the validation dataset.

```
LEARNING_RATE = 0.001
model.compile(
   optimizer=tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE),
   loss='categorical_crossentropy',
   metrics=['accuracy'])
```

```
EPOCHS=15
history = model.fit(
        train_generator,
        steps_per_epoch=train_generator.samples//train_generator.batch_size,
        epochs=EPOCHS,
        validation_data=validation_generator,
        validation_steps=validation_generator.samples//validation_generator.batch_si
```

```
Epoch 1/15
1/1 [==============================] - 7s 7s/step - loss: 1.0256 - accuracy: 0.3750 - val_loss: 1.5997 - val_accuracy: 0.5000
Epoch 2/15
1/1 [==============================] - 2s 2s/step - loss: 1.5474 - accuracy: 0.5625 - val_loss: 0.4876 - val_accuracy: 0.6875
Epoch 3/15
1/1 [==============================] - 1s 1s/step - loss: 0.6645 - accuracy: 0.6250 - val_loss: 0.6079 - val_accuracy: 0.6250
Epoch 4/15
1/1 [==============================] - 2s 2s/step - loss: 0.2712 - accuracy: 0.9375 - val_loss: 0.4773 - val_accuracy: 0.7500
Epoch 5/15
1/1 [==============================] - 2s 2s/step - loss: 0.1947 - accuracy: 0.9375 - val_loss: 0.4923 - val_accuracy: 0.7500
Epoch 6/15
1/1 [==============================] - 2s 2s/step - loss: 0.0931 - accuracy: 1.0000 - val_loss: 0.4161 - val_accuracy: 0.6875
Epoch 7/15
1/1 [==============================] - 2s 2s/step - loss: 0.0243 - accuracy: 1.0000 - val_loss: 0.5859 - val_accuracy: 0.6875
Epoch 8/15
1/1 [==============================] - 1s 1s/step - loss: 0.0417 - accuracy: 1.0000 - val_loss: 0.7603 - val_accuracy: 0.6250
Epoch 9/15
1/1 [==============================] - 1s 1s/step - loss: 0.0194 - accuracy: 1.0000 - val_loss: 0.7461 - val_accuracy: 0.7500
Epoch 10/15
1/1 [==============================] - 2s 2s/step - loss: 0.0566 - accuracy: 1.0000 - val_loss: 0.6342 - val_accuracy: 0.7500
Epoch 11/15
1/1 [==============================] - 1s 1s/step - loss: 0.0197 - accuracy: 1.0000 - val_loss: 1.0554 - val_accuracy: 0.7500
Epoch 12/15
1/1 [==============================] - 1s 1s/step - loss: 0.0231 - accuracy: 1.0000 - val_loss: 1.0372 - val_accuracy: 0.6875
Epoch 13/15
1/1 [==============================] - 1s 1s/step - loss: 0.1632 - accuracy: 0.8750 - val_loss: 0.8556 - val_accuracy: 0.6250
Epoch 14/15
1/1 [==============================] - 1s 1s/step - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.5263 - val_accuracy: 0.6875
Epoch 15/15
1/1 [==============================] - 1s 1s/step - loss: 0.0158 - accuracy: 1.0000 - val_loss: 0.3994 - val_accuracy: 0.8125
```

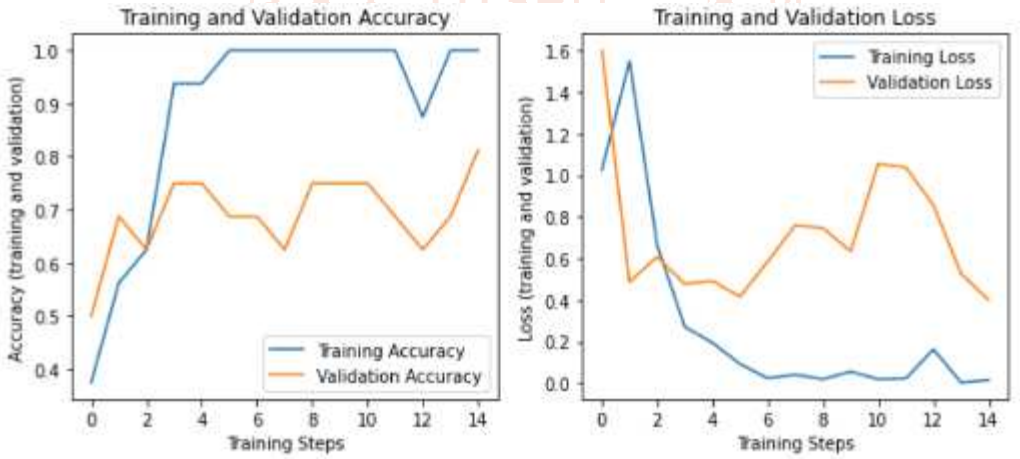
Model training

9. Checking Performance

We plotted the training and validation, accuracy and loss.

```
import matplotlib.pylab as plt
import numpy as np

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(EPOCHS)

plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.ylabel("Accuracy (training and validation)")
plt.xlabel("Training Steps")

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.ylabel("Loss (training and validation)")
plt.xlabel("Training Steps")
plt.show()
```

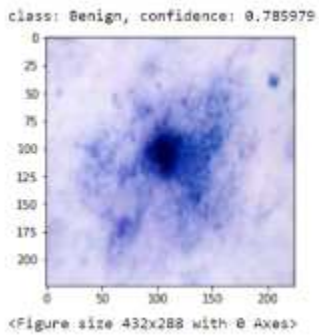
The Graph:

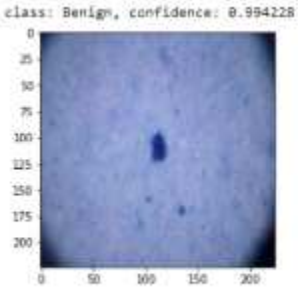


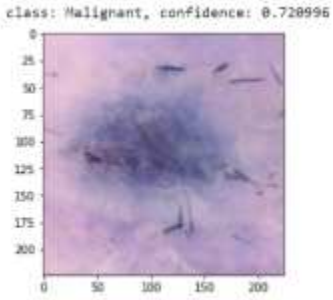
10. Testing Model

We picked five images at random from the validation dataset and made the following prediction:

```
import random
import cv2
def upload(filename):
    img = cv2.imread(os.path.join(train_dir, filename))
    img = cv2.resize(img, (224, 224) )
    img = img /255
    return img

def pre_result(image):
    x = model.predict(np.asarray([img]))[0]
    classx = np.argmax(x)
    return {Labels[classx]: x[classx]}
images = random.sample(validation_generator.filenames, 16)

for idx, filename in enumerate(images):
    img = upload(filename)
    prediction = pre_result(img)
    print("class: %s, confidence: %f" % (list(prediction.keys())[0], list(prediction
    plt.imshow(img)
    plt.figure(idx)
    plt.show()
```

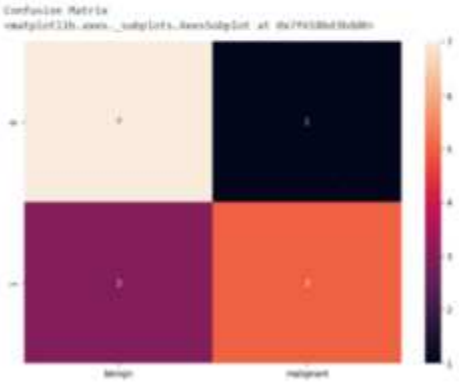






11. Confusion Matrix

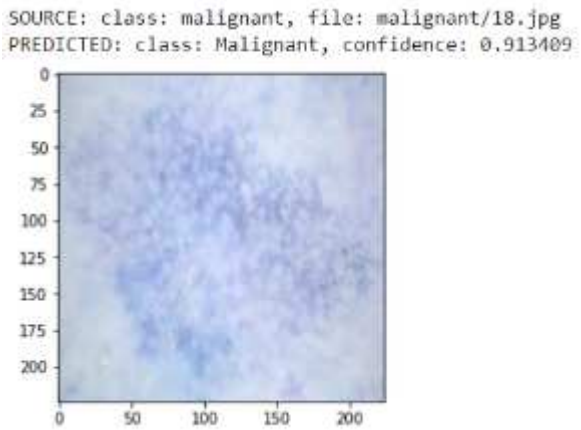
```
import pandas as pd
import numpy as np
import seaborn as sn
print('Confusion Matrix')
cm = confusion_matrix(validation_generator.classes, y)
df = pd.DataFrame(cm, columns=validation_generator.class_indices)
plt.figure(figsize=(10,7))
sn.heatmap(df, annot=True)
```

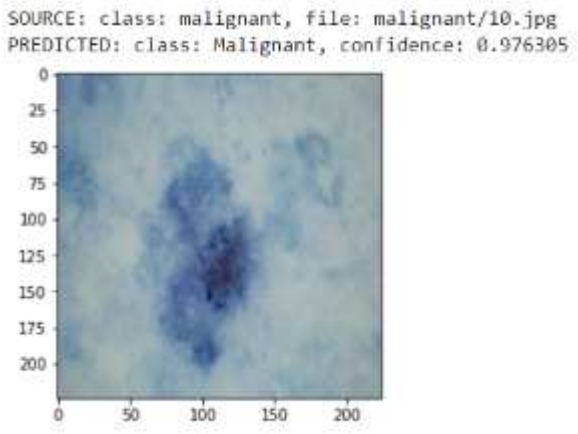


Confusion Matrix

12. Export Model and converting it to TFlite

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

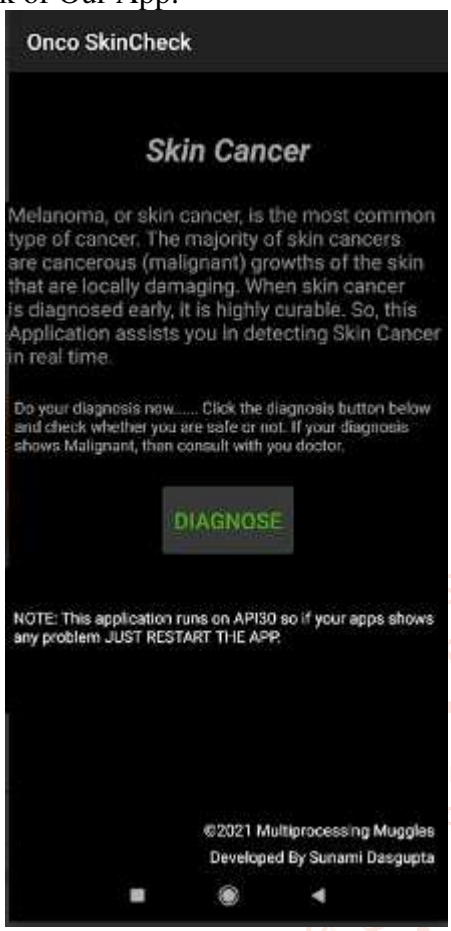




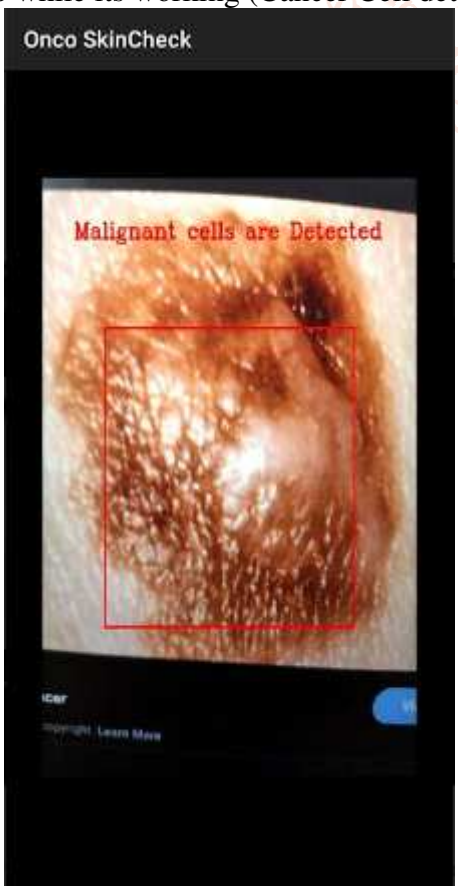
IMPLEMENTATION

We Later converted the model to TF lite format to implement on some real-life imaging. We used the app on an android app that we made using Java.

Out Look of Our App:



Our App while its working (Cancer Cell detected):



When Cell is benign:

The app was detecting Cancer cells at a 97% accuracy.

CONCLUSION

Looking back on this project, the overall outcome of results to be observed. This can be evaluated by looking at how well our objectives were met. We have successfully constructed our learning model that can predict the cancer cells by image processing. The performance at the training phase does not give the complete picture, we have to test it on real-data. So we later introduced it to an android application. This passion project helped us to get the In-depth knowledge and experience on machine learning and image processing.

REFERENCE

[1] Jhuria, Ashwani Kumar, and Rushikesh Borse, "Image Processing for Smart Farming: Detection of Disease and Fruit Grading", Proceedings of the 2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013)

[2] Chunxia Zhang, Xiuqing Wang, Xudong Li, "Design of Monitoring and Control Plant Disease System Based on DSP&FPGA", 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing.

[3] Dr. K. Thangadurai, K. Padmavathi, "Computer Visionimage Enhancement For Plant Leaves Disease Detection", 2014 World Congress on Computing and Communication Technologies.

[4] Peidle J., Stokes C., Hart R., Franklin M., Newburgh R., Pahk J., Rueckner W. & Samuel AD, (2009) "Inexpensive microscopy for introductory laboratory courses," American Journal of Physics Vol. 77 pp. 931-938.

[5] Joshi, A., Boyat, A. and Joshi, B. K. (2014) "Impact of Wavelet Transform and Median Filtering on removal of Salt and Pepper noise in Digital Images," IEEE International Conference on Issues and Challenges in Intelligant Computing Teachniques, Gaziabad.

[6] Salivahanan S., Vallavaraj A. &Gnanapriya C. (2008) "Digital Signal Processing," Tata McgrawHill, Vol. 23, NewDelhi.

[7] Bovick A. (2000) "Handbook of Image and Video processing," Acedemic press, New York.

[8] Astola J. & Kuosmanen P. (1997) "Fundamentals of nonlinear digital filtering," CRC Press, Boca Raton.

[9] Radenovic A., "Brownian motion and single particle tracking," Advanced Bioengineering methods laboratory, Ecole polyteachenique federal de Lausanne.

[10] Hosseini H. &Marvasti F., (2013) "Fast restoration of natural images corrupted by high-density impulse noise," EURASIP Journal on Image and Video Processing. doi: 10.1186/1687-5281-2013-15.