END-TO-END DEEP LEARNING PROJECT FOR DETECTING MELANOMA DISEASES

TEAM MEMBERS : Abilash R (abilash.r2021@vitstudent.ac.in)
Jermish Justin (jermish.justin2021@vitstudent.ac.in)

1. INTRODUCTION

1.1 Project Overview

The objective of this project is to develop an end-to-end deep learning solution for detecting melanoma disease. The proposed solution involves the use of convolutional neural networks (CNNs) to extract relevant features from the input images and classify them into one of the nine categories.

1.2 Purpose

Detecting melanoma disease as soon as possible is crucial because it can have significant impacts on our skin health. Early detection allows humans to take required action to prevent the spread of the disease. Here are some reasons why detecting melanoma disease early is essential:

1. Improved Prognosis
2. Reduced Mortality
3. Less Aggressive Treatment
4. Reduced Cost and Time

LITERATURE SURVEY

2.1 Existing problem

In the past few years many researches investigated algorithms to diagnose skin cancer lesions where melanoma is the deadliest type of skin cancer. Melanoma is the most threatening and deadliest form of skin cancer, From 2008 to 2018, the annual number of melanoma cases has increased by 53%, partly due to increased UV exposure. The first step in the diagnosis of a malignant lesion by a dermatologist is visual examination of the suspicious skin area. A correct diagnosis is important because of the similarities of some lesion types, Skin cancer can be cured if early detected, but only highly-trained specialists are capable of accurately diagnosing skin cancer early. Without additional technical support, dermatologists have a 65%-80% accuracy rate in melanoma diagnosis

2.2 References

Brinker, T. J., Hekler, A., Utikal, J., Grabe, N., Schadendorf, D., Klode, J., Berking, C., Steeb, T., Enk, A., & Von Kalle, C. (2018, October 17). Skin Cancer Classification Using Convolutional Neural Networks: Systematic Review. Journal of Medical Internet Research. https://doi.org/10.2196/11936

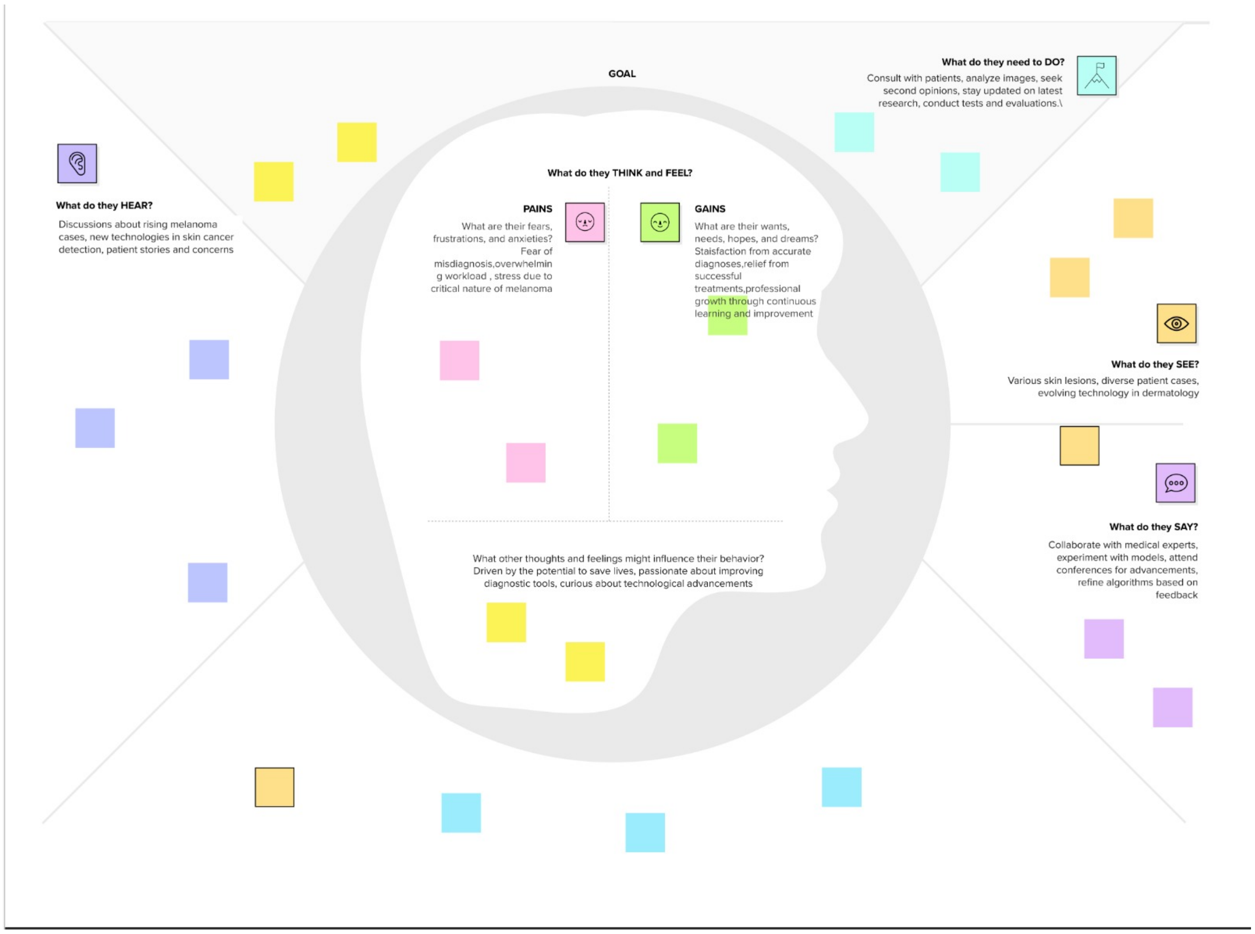
Medhat, S., Abdel-Galil, H., Aboutabl, A. E., & Saleh, H. (2022, March 1). Skin cancer diagnosis using convolutional neural networks for smartphone images: A comparative study. Journal of Radiation Research and Applied Sciences. https://doi.org/10.1016/j.jrras.2022.03.008

2.3 Problem Statement Definition

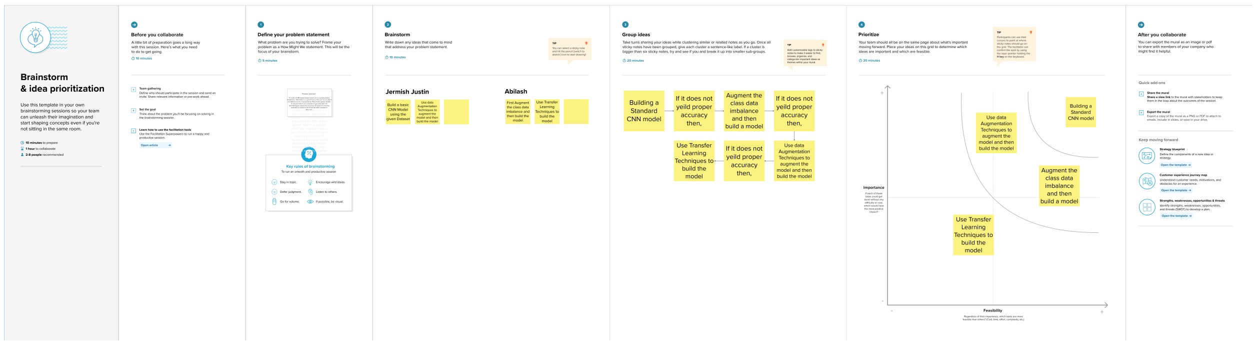
To build a CNN based model which can accurately detect melanoma. Melanoma is a type of cancer that can be deadly if not detected early. It accounts for 75% of skin cancer deaths. A solution which can evaluate images and alert the dermatologists about the presence of melanoma has the potential to reduce a lot of manual effort needed in diagnosis.

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas


GOAL
What do they need to DO?
Consult with patients, analyze images, seek second opinions, stay updated on latest research, conduct tests and evaluations.\
What do they HEAR?
Discussions about rising melanoma cases, new technologies in skin cancer detection, patient stories and concerns
What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties? Fear of misdiagnosis,overwhelming workload , stress due to critical nature of melanoma
GAINS
What are their wants, needs, hopes, and dreams? Staisfaction from accurate diagnoses,relief from successful treatments,professional growth through continuous learning and improvement
What other thoughts and feelings might influence their behavior? Driven by the potential to save lives, passionate about improving diagnostic tools, curious about technological advancements
What do they SEE?
Various skin lesions, diverse patient cases, evolving technology in dermatology
What do they SAY?
Collaborate with medical experts, experiment with models, attend conferences for advancements, refine algorithms based on feedback

3.2 IDEATION AND BRAINSTROMING


Brainstorm & idea prioritization
Before you collaborate
Define your problem statement
Brainstorm
Jermish Justin
Build a basic CNN Model using the given Dataset
Use data Augmentation Techniques to augment the model and then build the model
Abilash
First Augment the class data imbalance and then build the model
Use Transfer Learning Techniques to build the model
Group ideas
Building a Standard CNN model
If it does not yeild proper accuracy then,
Augment the class data imbalance and then build a model
If it does not yeild proper accuracy then,
Use data Augmentation Techniques to augment the model and then build the model
If it does not yeild proper accuracy then,
Use Transfer Learning Techniques to build the model
Prioritize
Importance
Feasibility
Building a Standard CNN model
Use data Augmentation Techniques to augment the model and then build the model
Augment the class data imbalance and then build a model
Use Transfer Learning Techniques to build the model
After you collaborate

4.REQUIREMENT ANALYSIS

4.1 Functional Requirement Analysis

- The system should accept digital images of skin lesions as input.
- Implement image preprocessing techniques to enhance features
- Extract relevant features from skin lesion images using Convolutional Neural Networks (CNNs).
- Utilize the CNN model to classify skin lesions into categories, including melanoma and non-melanoma.
- Enable real-time processing to facilitate quick diagnosis and timely intervention
- Facilitate integration with existing healthcare systems for seamless adoption in clinical workflows.

4.2 Non-Functional requirements

- Achieve a high level of accuracy in melanoma detection, minimizing false positives and false negatives.
- Design the system to handle a scalable number of input images for widespread usage.
- Ensure low response times for image processing and classification to provide timely results.
- Design the system to be robust against variations in image quality, lighting conditions, and skin types.
- Design an intuitive and user-friendly interface for healthcare professionals to interact with the system.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

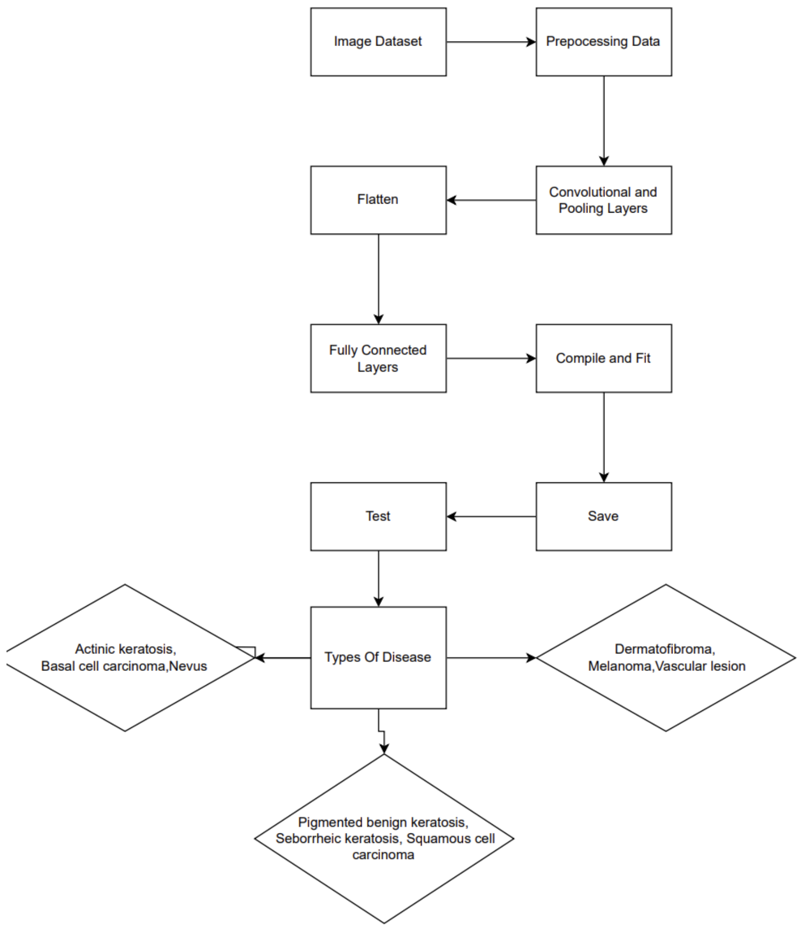
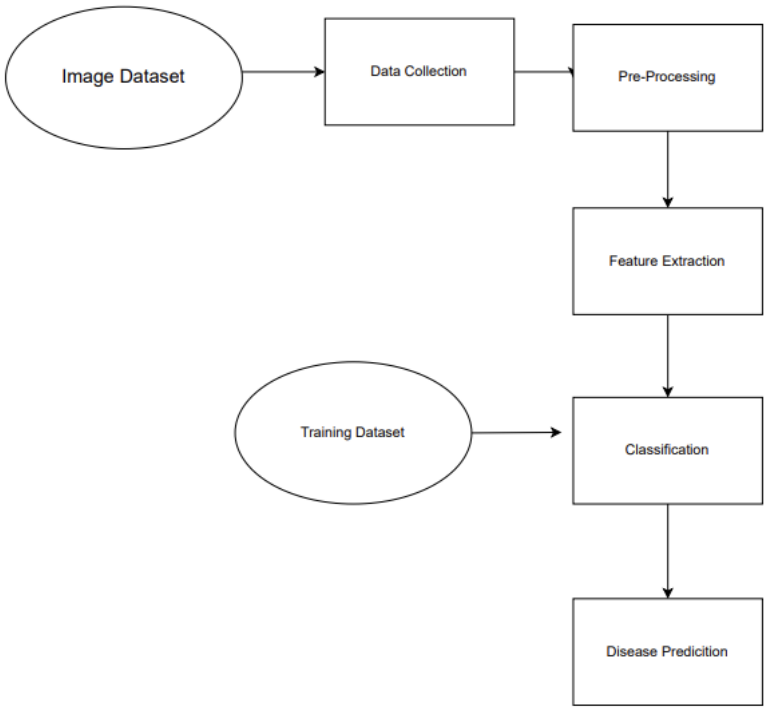


Image Dataset
Prepocessing Data
Convolutional and Pooling Layers
Flatten
Fully Connected Layers
Compile and Fit
Save
Test
Types Of Disease
Actinic keratosis, Basal cell carcinoma,Nevus
Dermatofibroma, Melanoma,Vascular lesion
Pigmented benign keratosis, Seborrheic keratosis, Squamous cell carcinoma

User Stories

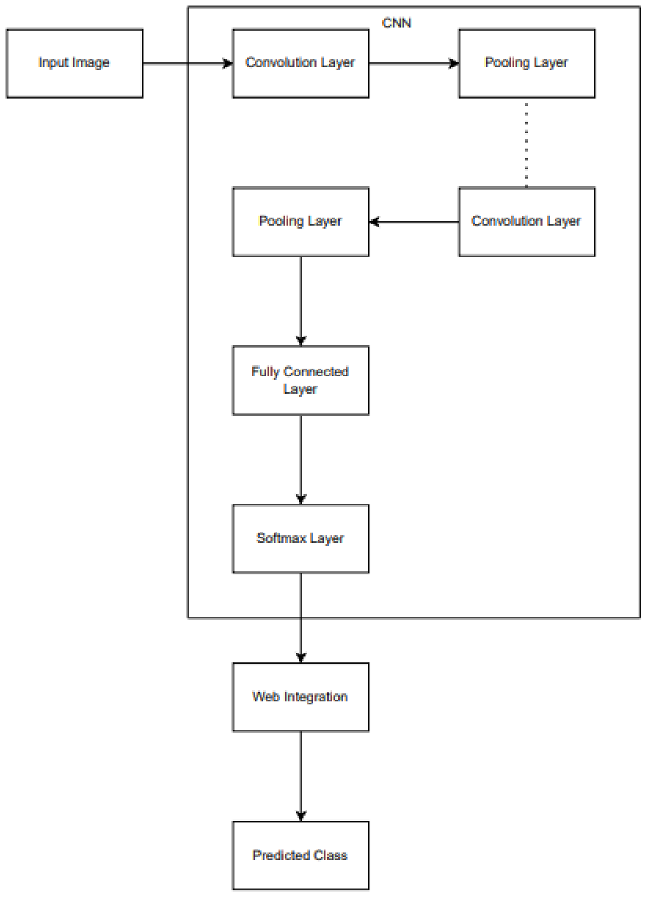
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Dermatologist (User)	Prediction	USN-1	As a dermatologist, I want the CNN model to accurately identify potential melanomas in skin images, so that I can efficiently prioritize and focus on cases that require further examination.	Model Should have high accuracy	High	Sprint-1
Patient (User)	Prediction	USN-2	As a Patient, I want the CNN model to accurately identify potential melanomas in skin images, so that I can seek medical help early .	Model Should have high accuracy	High	Sprint-1

5.2 Solution Architecture



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

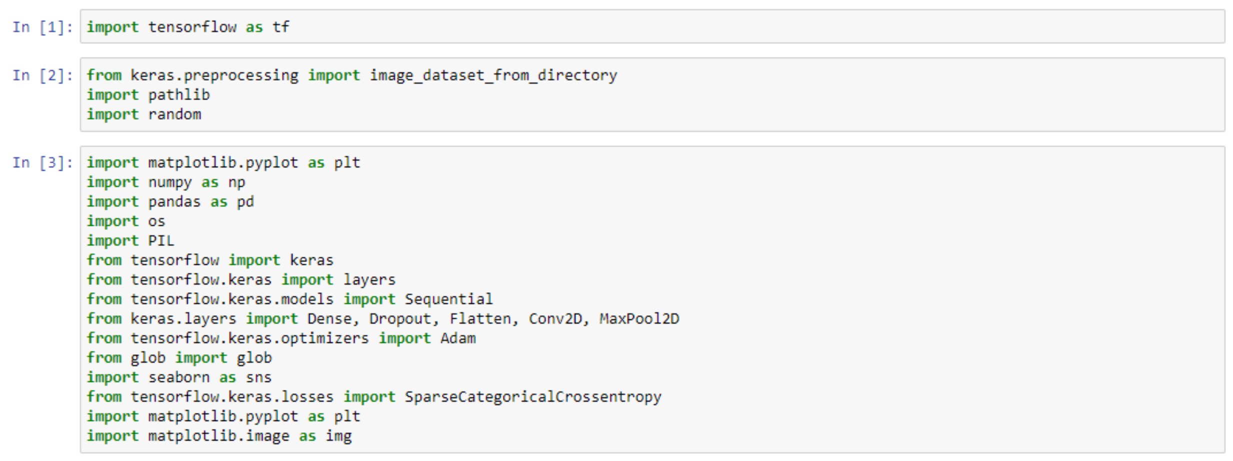
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Prediction	USN-1	As a dermatologist, I want the CNN model to accurately identify potential melanomas in skin images, so that I can efficiently prioritize and focus on cases that require further examination.	5	High	Jermish Justin, Abilash
Sprint 1	Prediction	USN-2	As a Patient, I want the CNN model to accurately identify potential melanomas in skin images, so that I can seek medical help early .	5	High	Jermish Justin, Abilash

6.3 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	10	4	15 November 2023	18 November 2023	10	18 November

7. CODING & SOLUTIONING

7.1 Importing necessary Libraries

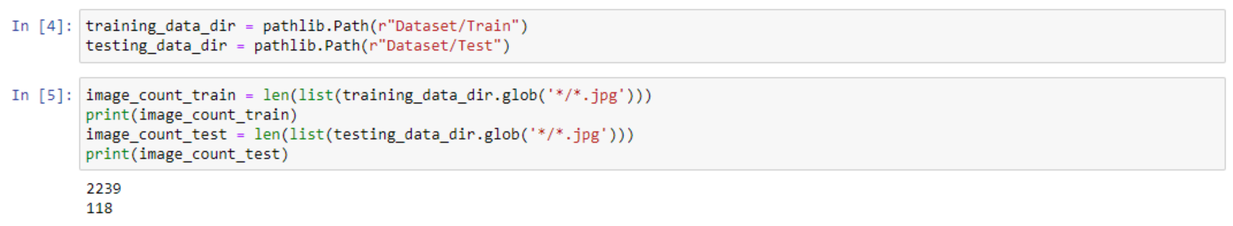


```
In [1]: import tensorflow as tf
```

```
In [2]: from keras.preprocessing import image_dataset_from_directory
        import pathlib
        import random
```

```
In [3]: import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        import os
        import PIL
        from tensorflow import keras
        from tensorflow.keras import layers
        from tensorflow.keras.models import Sequential
        from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
        from tensorflow.keras.optimizers import Adam
        from glob import glob
        import seaborn as sns
        from tensorflow.keras.losses import SparseCategoricalCrossentropy
        import matplotlib.pyplot as plt
        import matplotlib.image as img
```

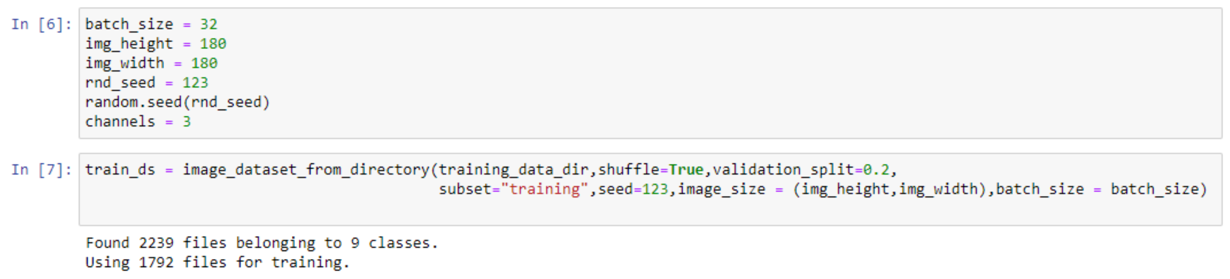
7.2 Reading Input Data



```
In [4]: training_data_dir = pathlib.Path(r"Dataset/Train")
        testing_data_dir = pathlib.Path(r"Dataset/Test")
```

```
In [5]: image_count_train = len(list(training_data_dir.glob('*/*.jpg')))
        print(image_count_train)
        image_count_test = len(list(testing_data_dir.glob('*/*.jpg')))
        print(image_count_test)

        2239
        118
```

7.3 Preparing the Dataset



```
In [6]: batch_size = 32
        img_height = 180
        img_width = 180
        rnd_seed = 123
        random.seed(rnd_seed)
        channels = 3
```

```
In [7]: train_ds = image_dataset_from_directory(training_data_dir,shuffle=True,validation_split=0.2,
                                                subset="training",seed=123,image_size = (img_height,img_width),batch_size = batch_size)

        Found 2239 files belonging to 9 classes.
        Using 1792 files for training.
```

```
In [8]: val_ds = tf.keras.preprocessing.image_dataset_from_directory(
          training_data_dir,
          validation_split=0.2,
          subset="validation",
          seed=123,
          image_size=(img_height, img_width),
          batch_size=batch_size)
```

```
Found 2239 files belonging to 9 classes.
Using 447 files for validation.
```

```
In [9]: test_ds = tf.keras.preprocessing.image_dataset_from_directory(
          testing_data_dir,
          validation_split=0.9,
          subset="validation",
          seed=123,
          image_size=(img_height, img_width),
          batch_size=batch_size)
```

```
Found 118 files belonging to 9 classes.
Using 106 files for validation.
```

7.4 Plotting the images

```
In [11]: num_classes = len(class_names)
         plt.figure(figsize=(10,10))
         for i in range(num_classes):
             plt.subplot(3,3,i+1)
             image = img.imread(str(list(training_data_dir.glob(class_names[i]+'/*.jpg'))[1]))
             plt.title(class_names[i])
             plt.imshow(image)
```

7.5 Autotuning the Model

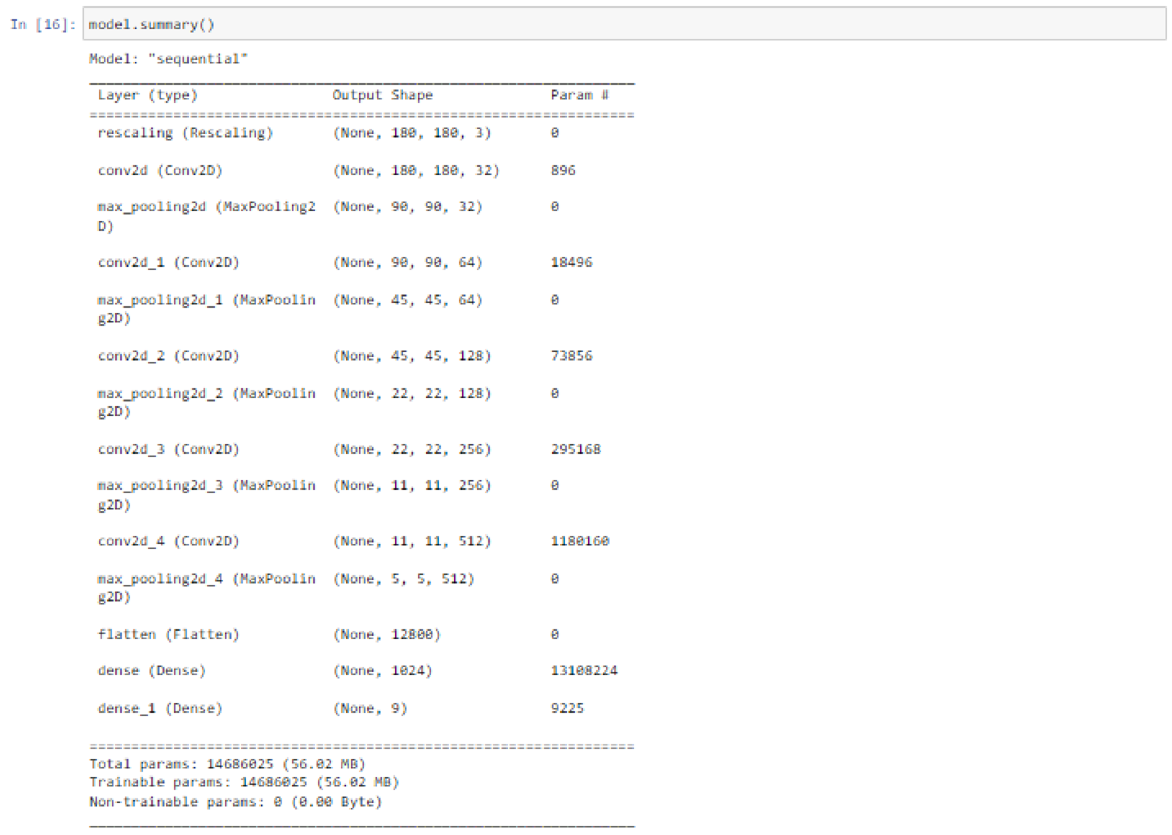
```
In [13]: AUTOTUNE = tf.data.experimental.AUTOTUNE
         train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
         val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

7.6 Building Standard model

```
In [14]: num_classes = 9
         model = Sequential([layers.experimental.preprocessing.Rescaling \
                             (1.0/255,input_shape=(img_height,img_width,3))])

         model.add(Conv2D(32, 3,padding="same",activation='relu'))
         model.add(MaxPool2D())

         model.add(Conv2D(64, 3,padding="same",activation='relu'))
         model.add(MaxPool2D())

         model.add(Conv2D(128, 3,padding="same",activation='relu'))
         model.add(MaxPool2D())

         model.add(Conv2D(256, 3,padding="same",activation='relu'))
         model.add(MaxPool2D())

         model.add(Conv2D(512, 3,padding="same",activation='relu'))
         model.add(MaxPool2D())

         model.add(Flatten())
         model.add(Dense(1024,activation="relu"))
         model.add(Dense(units=num_classes, activation= 'softmax'))
```

```
In [15]: opt = Adam(lr=0.001)
         model.compile(optimizer= opt,
                       loss= SparseCategoricalCrossentropy(from_logits=True),
                       metrics=['accuracy'])
```

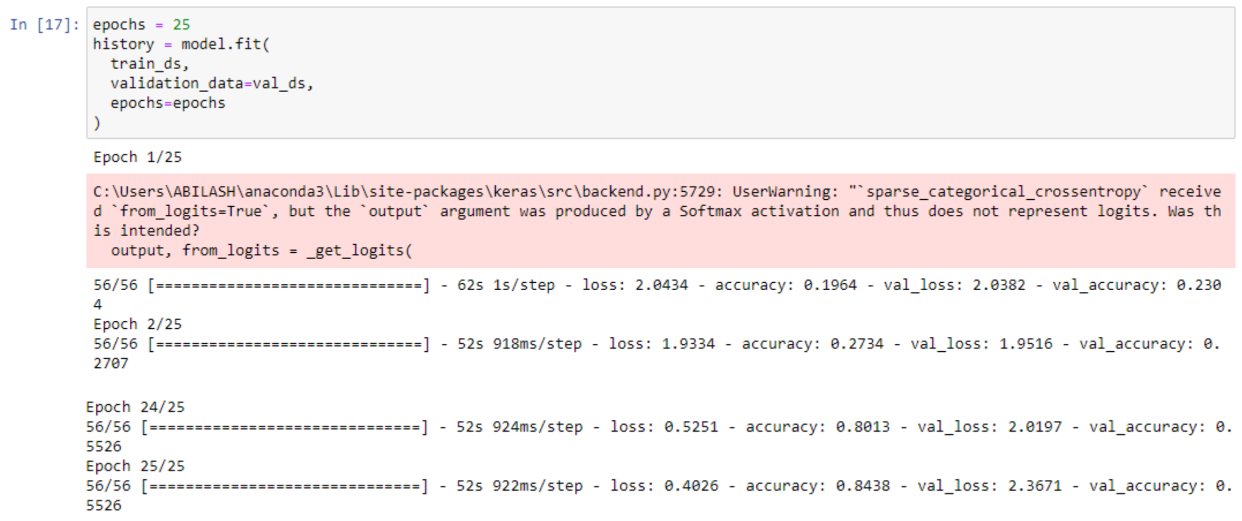
```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optim
izers.legacy.Adam.
```



```
In [16]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 180, 180, 3)       0

 conv2d (Conv2D)             (None, 180, 180, 32)      896

 max_pooling2d (MaxPooling2  (None, 90, 90, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 90, 90, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 45, 45, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 45, 45, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 22, 22, 128)       0
 g2D)

 conv2d_3 (Conv2D)           (None, 22, 22, 256)       295168

 max_pooling2d_3 (MaxPoolin  (None, 11, 11, 256)       0
 g2D)

 conv2d_4 (Conv2D)           (None, 11, 11, 512)       1180160

 max_pooling2d_4 (MaxPoolin  (None, 5, 5, 512)         0
 g2D)

 flatten (Flatten)           (None, 12800)             0

 dense (Dense)               (None, 1024)              13108224

 dense_1 (Dense)             (None, 9)                 9225

=================================================================
Total params: 14686025 (56.02 MB)
Trainable params: 14686025 (56.02 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

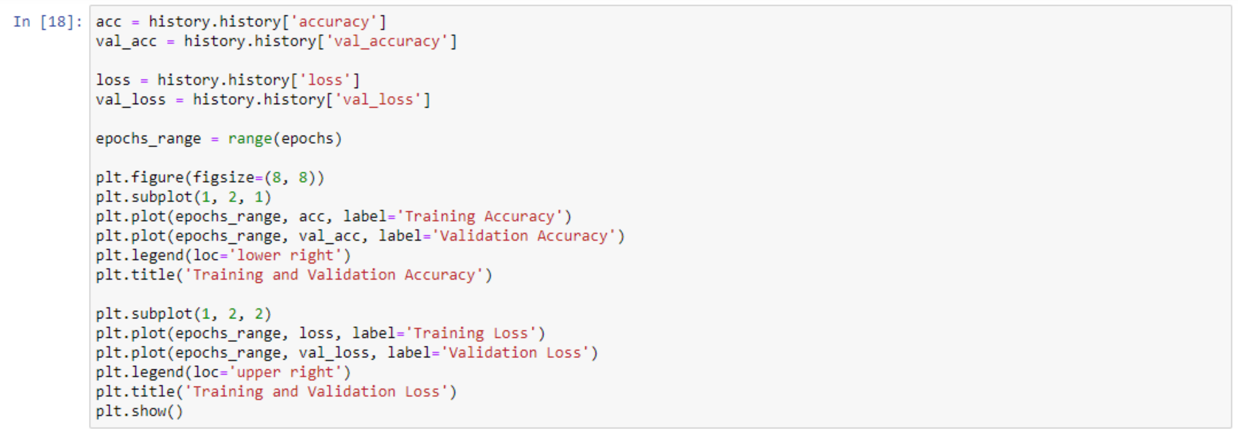
7.7 Standard model Fit



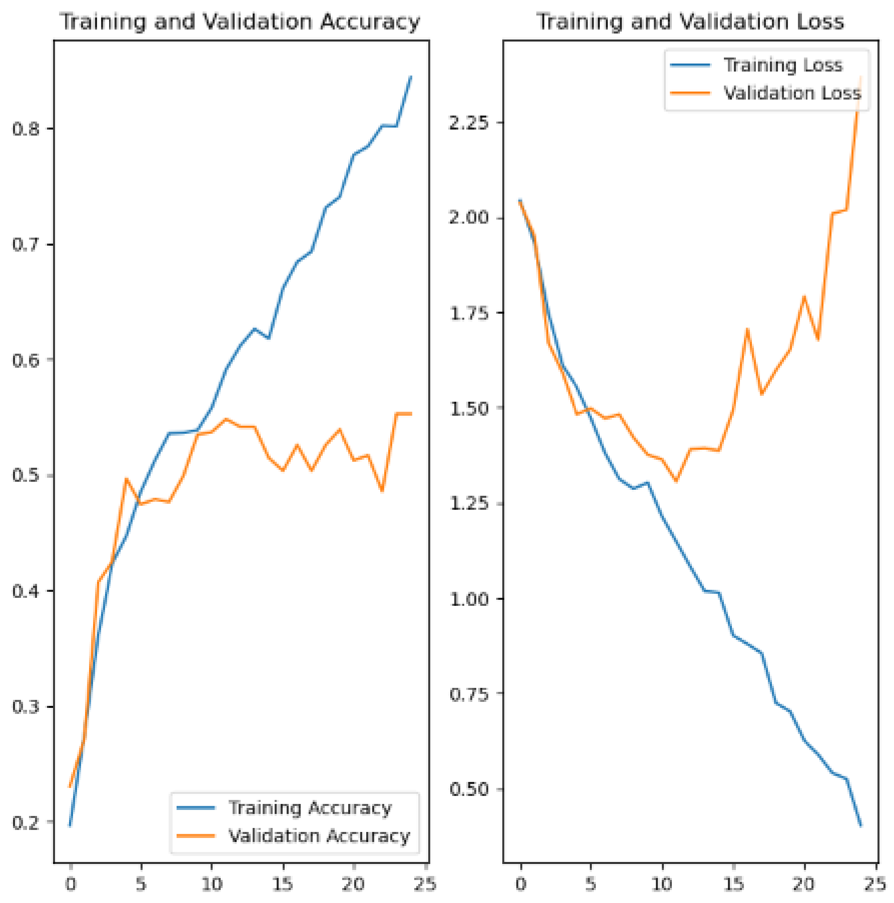
```
In [17]: epochs = 25
         history = model.fit(
           train_ds,
           validation_data=val_ds,
           epochs=epochs
         )
```

```
Epoch 1/25
C:\Users\ABILASH\anaconda3\Lib\site-packages\keras\src\backend.py:5729: UserWarning: "`sparse_categorical_crossentropy` receive
d `from_logits=True`, but the `output` argument was produced by a Softmax activation and thus does not represent logits. Was th
is intended?
  output, from_logits = _get_logits(
56/56 [==============================] - 62s 1s/step - loss: 2.0434 - accuracy: 0.1964 - val_loss: 2.0382 - val_accuracy: 0.230
4
Epoch 2/25
56/56 [==============================] - 52s 918ms/step - loss: 1.9334 - accuracy: 0.2734 - val_loss: 1.9516 - val_accuracy: 0.
2707

Epoch 24/25
56/56 [==============================] - 52s 924ms/step - loss: 0.5251 - accuracy: 0.8013 - val_loss: 2.0197 - val_accuracy: 0.
5526
Epoch 25/25
56/56 [==============================] - 52s 922ms/step - loss: 0.4026 - accuracy: 0.8438 - val_loss: 2.3671 - val_accuracy: 0.
5526
```

7.8 Plotting graph between Validation and Training Accuracy and Validation and Training Loss



```
In [18]: 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=(8, 8))
         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.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.show()
```

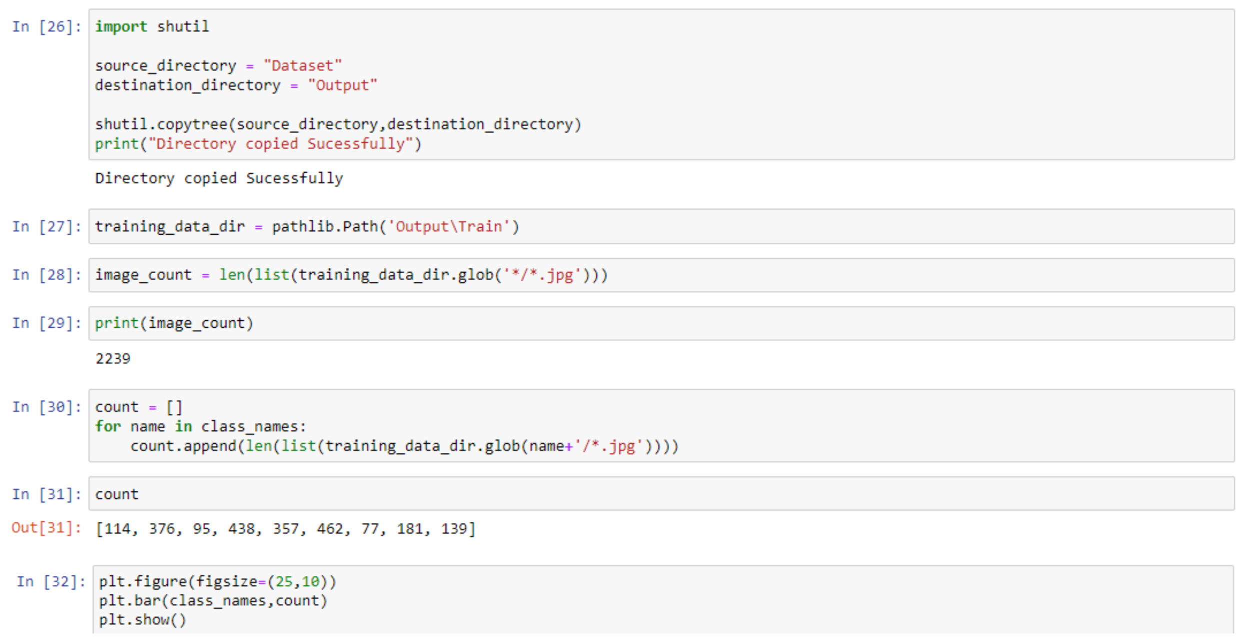


- The model is overfitting because we can see the difference in accuracy in training data & accuracy in the validation data that is almost 30%.
- The training accuracy is just around 80-85% with 25 epochos and the model is yet to

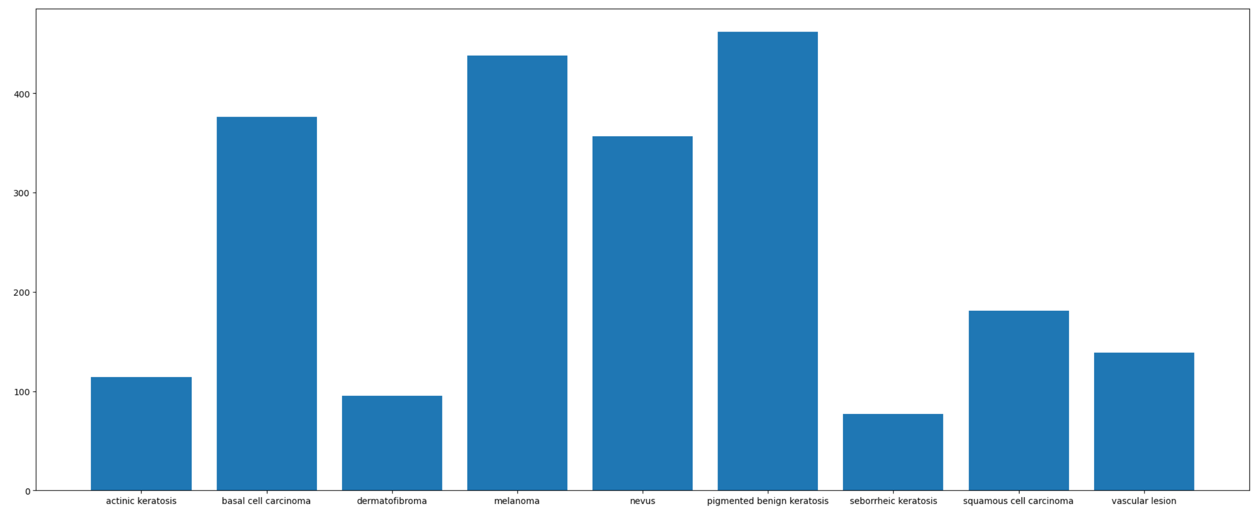
learn the many features.

- This bias in the model can be due to data imbalance.

7.9 Analysing class data imbalance



```
In [26]: import shutil

         source_directory = "Dataset"
         destination_directory = "Output"

         shutil.copytree(source_directory,destination_directory)
         print("Directory copied Sucessfully")

         Directory copied Sucessfully

In [27]: training_data_dir = pathlib.Path('Output\Train')

In [28]: image_count = len(list(training_data_dir.glob('*/*.jpg')))

In [29]: print(image_count)

         2239

In [30]: count = []
         for name in class_names:
             count.append(len(list(training_data_dir.glob(name+'/*.jpg'))))

In [31]: count

Out[31]: [114, 376, 95, 438, 357, 462, 77, 181, 139]

In [32]: plt.figure(figsize=(25,10))
         plt.bar(class_names,count)
         plt.show()
```



7.10 Fixing class data imbalance

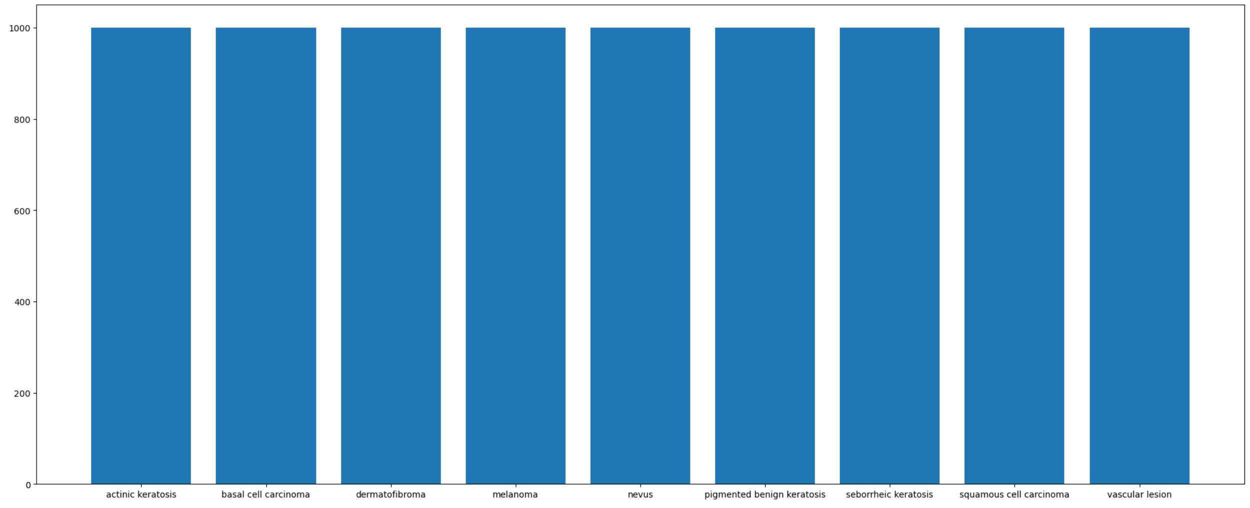
```
In [33]: import Augmentor

         path_to_training_dataset = str(training_data_dir) + '/'

         for i in class_names:
             p = Augmentor.Pipeline(path_to_training_dataset + i)
             p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation = 10)
             p.sample(1000)
```

Adding 1000 samples per class, to make sure that none of the classes are sparse

```
In [35]: image_count_train = len(list(training_data_dir.glob('*/output/*.jpg')))
         print(image_count_train)

         9000
```

Each class has 1000 images

```
In [37]: count = []
         for name in class_names:
             count.append(len(list(training_data_dir.glob(name+'*/output/*.jpg'))))
         plt.figure(figsize=(25,10))
         plt.bar(class_names,count)
```



All the classess are balanced

7.11 Creating a dataframe which includes “Path” and “Label” of every image in the dataset

```
In [38]: import os
         from glob import glob

         path_list_new  = [x for x in glob(os.path.join(training_data_dir, "*","output","*.jpg"))]
         lesion_list_new = [os.path.basename(os.path.dirname(os.path.dirname(y)))
                            for y in glob(os.path.join(training_data_dir,"*","output","*.jpg"))]
         df_dict_new = dict(zip(path_list_new,lesion_list_new))
```

```
In [39]: path_list = []
         lesion_list = []
         for name in class_names:
             for file in training_data_dir.glob(name+'/*.jpg'):
                 path_list.append(str(file))
                 lesion_list.append(name)
         df_dict_original = dict(zip(path_list,lesion_list))
         original_df = pd.DataFrame(list(df_dict_original.items()),columns=['Path','Label'])
```

7.12 Prepocessing data to build a model with class balance data

```
In [40]: train_ds = tf.keras.preprocessing.image_dataset_from_directory(
           training_data_dir,
           seed=123,
           validation_split = 0.2,
           subset = 'training',
           image_size=(img_height, img_width),
           batch_size=batch_size)
```

```
Found 11239 files belonging to 9 classes.
Using 8992 files for training.
```

```
In [41]: val_ds = tf.keras.preprocessing.image_dataset_from_directory(
           training_data_dir,
           seed=123,
           validation_split = 0.2,
           subset = 'validation',
           image_size=(img_height, img_width),
           batch_size=batch_size)
```

```
Found 11239 files belonging to 9 classes.
Using 2247 files for validation.
```

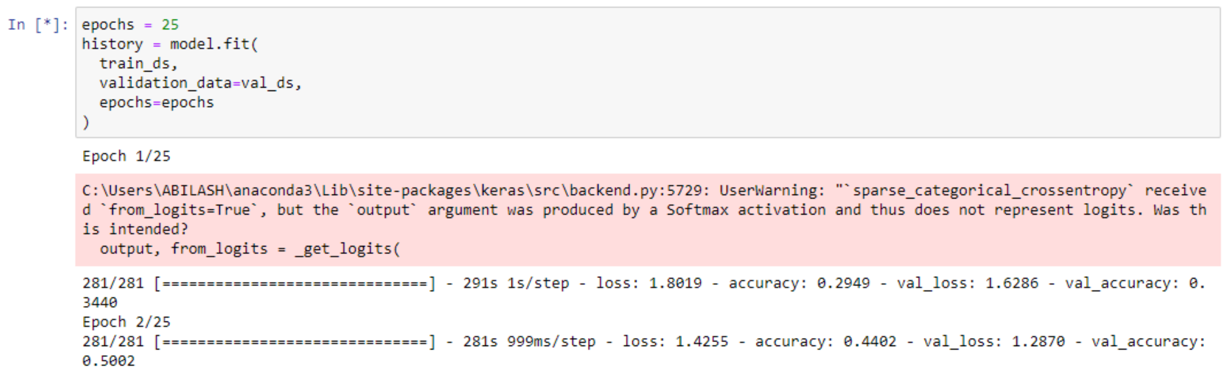
7.13 Building a model with class balance data

```
In [43]: num_classes = 9
         model = Sequential([layers.experimental.preprocessing.Rescaling(1.0/255,input_shape=(img_height,img_width,3))])

         model.add(Conv2D(32, 3,padding="same",activation='relu'))
         model.add(MaxPool2D())

         model.add(Conv2D(64, 3,padding="same",activation='relu'))
         model.add(MaxPool2D())

         model.add(Conv2D(128, 3,padding="same",activation='relu'))
         model.add(MaxPool2D())
         model.add(Dropout(0.15))

         model.add(Conv2D(256, 3,padding="same",activation='relu'))
         model.add(MaxPool2D())
         model.add(Dropout(0.20))

         model.add(Conv2D(512, 3,padding="same",activation='relu'))
         model.add(MaxPool2D())
         model.add(Dropout(0.25))

         model.add(Flatten())
         model.add(Dense(1024,activation="relu"))
         model.add(Dense(units=num_classes, activation= 'softmax'))
```

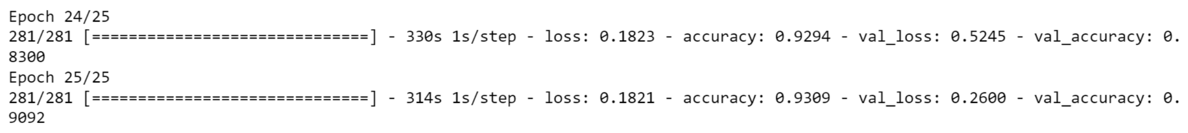
```
In [44]: opt = Adam(lr=0.001)
         model.compile(optimizer= opt,
                       loss = SparseCategoricalCrossentropy(from_logits=True),
                       metrics=['accuracy'])
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optim
izers.legacy.Adam.
```

7.14 Class balance model fit

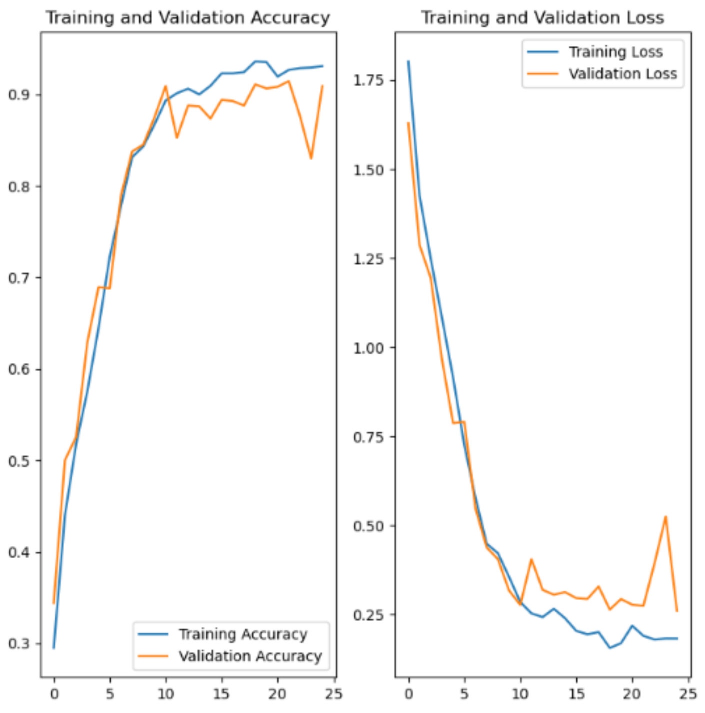


```
In [*]: epochs = 25
        history = model.fit(
          train_ds,
          validation_data=val_ds,
          epochs=epochs
        )

        Epoch 1/25

        C:\Users\ABILASH\anaconda3\Lib\site-packages\keras\src\backend.py:5729: UserWarning: "`sparse_categorical_crossentropy` receive
        d `from_logits=True`, but the `output` argument was produced by a Softmax activation and thus does not represent logits. Was th
        is intended?
          output, from_logits = _get_logits(

        281/281 [==============================] - 291s 1s/step - loss: 1.8019 - accuracy: 0.2949 - val_loss: 1.6286 - val_accuracy: 0.
        3440
        Epoch 2/25
        281/281 [==============================] - 281s 999ms/step - loss: 1.4255 - accuracy: 0.4402 - val_loss: 1.2870 - val_accuracy:
        0.5002
```



```
Epoch 24/25
281/281 [==============================] - 330s 1s/step - loss: 0.1823 - accuracy: 0.9294 - val_loss: 0.5245 - val_accuracy: 0.
8300
Epoch 25/25
281/281 [==============================] - 314s 1s/step - loss: 0.1821 - accuracy: 0.9309 - val_loss: 0.2600 - val_accuracy: 0.
9092
```

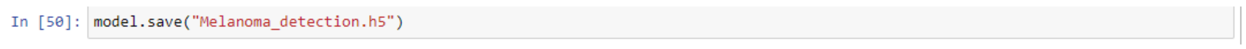
7.15 Plotting graph between Validation and Training Accuracy and Validation and Training Loss



- After resampling of the data, the accuray of the model increased to nearly 90%.
- Problem of overfitting is solved and the difference between train and validation dataset is

nearly 2-3%

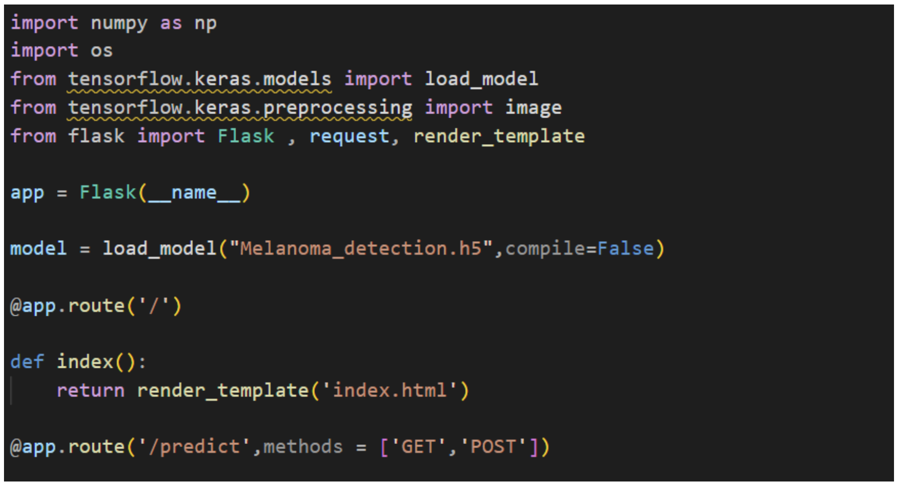
7.16 Saving the Model



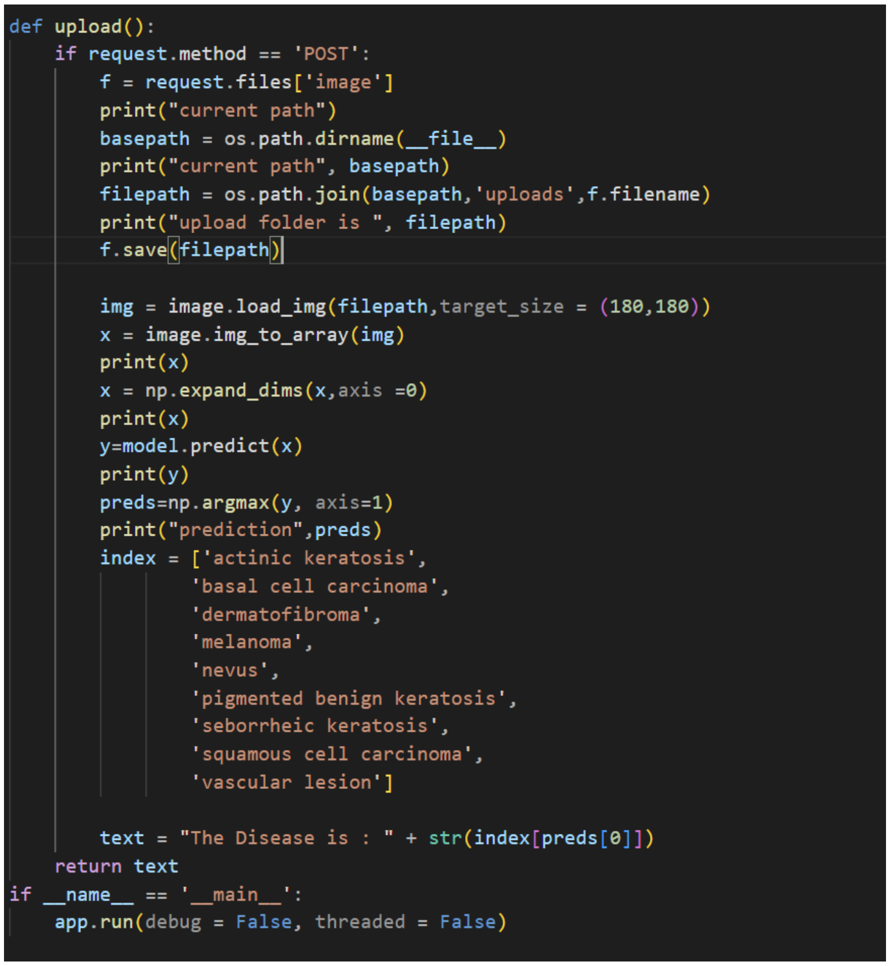
```
In [50]: model.save("Melanoma_detection.h5")
```

7.17 FLASK INTEGRATION

Path: Flask/app1.py

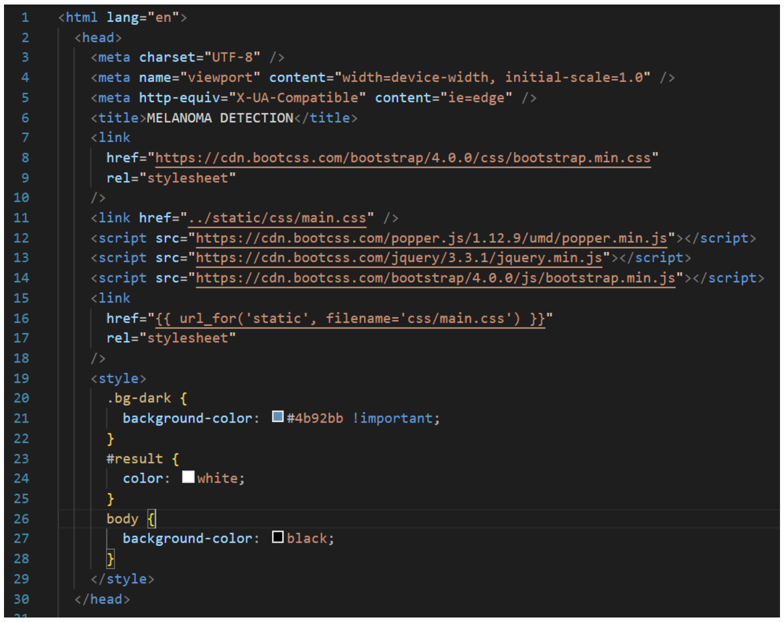


```
import numpy as np
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask , request, render_template

app = Flask(__name__)

model = load_model("Melanoma_detection.h5",compile=False)

@app.route('/')

def index():
    return render_template('index.html')

@app.route('/predict',methods = ['GET','POST'])
```

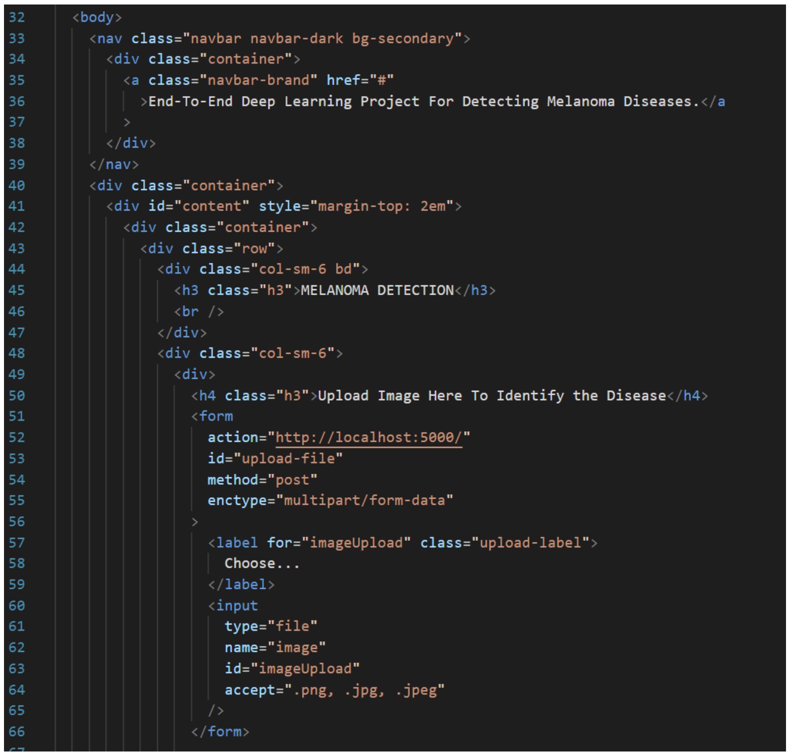


```python
def upload():
    if request.method == 'POST':
        f = request.files['image']
        print("current path")
        basepath = os.path.dirname(__file__)
        print("current path", basepath)
        filepath = os.path.join(basepath,'uploads',f.filename)
        print("upload folder is ", filepath)
        f.save(filepath)

        img = image.load_img(filepath,target_size = (180,180))
        x = image.img_to_array(img)
        print(x)
        x = np.expand_dims(x,axis =0)
        print(x)
        y=model.predict(x)
        print(y)
        preds=np.argmax(y, axis=1)
        print("prediction",preds)
        index = ['actinic keratosis',
                'basal cell carcinoma',
                'dermatofibroma',
                'melanoma',
                'nevus',
                'pigmented benign keratosis',
                'seborrheic keratosis',
                'squamous cell carcinoma',
                'vascular lesion']

        text = "The Disease is : " + str(index[preds[0]])
    return text
if __name__ == '__main__':
    app.run(debug = False, threaded = False)
```

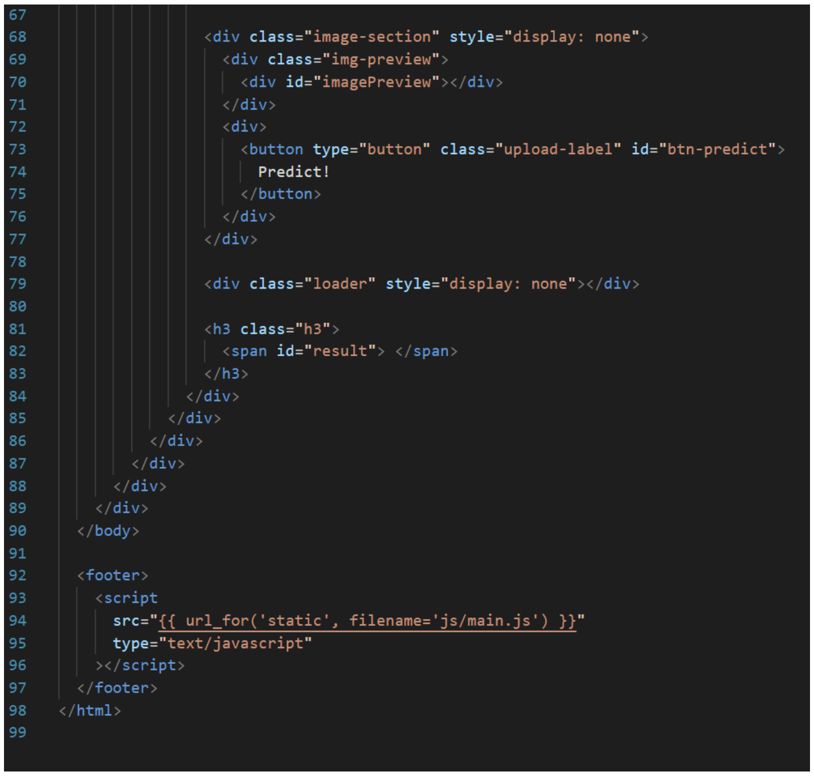
Path: Flask/templates/index.html



```html
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>MELANOMA DETECTION</title>
    <link
      href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css"
      rel="stylesheet"
    />
    <link href="../static/css/main.css" />
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <link
      href="{{ url_for('static', filename='css/main.css') }}"
      rel="stylesheet"
    />
    <style>
      .bg-dark {
        background-color: #4b92bb !important;
      }
      #result {
        color: white;
      }
      body {
        background-color: black;
      }
    </style>
  </head>
```

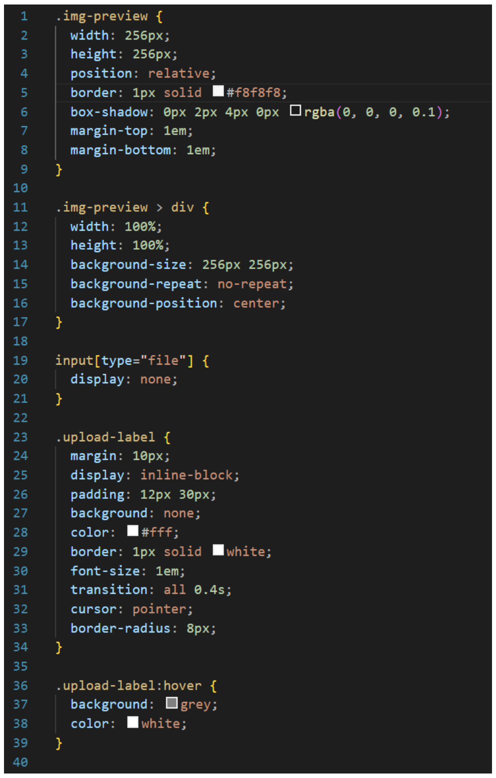


```html
  <body>
    <nav class="navbar navbar-dark bg-secondary">
      <div class="container">
        <a class="navbar-brand" href="#"
          >End-To-End Deep Learning Project For Detecting Melanoma Diseases.</a
        >
      </div>
    </nav>
    <div class="container">
      <div id="content" style="margin-top: 2em">
        <div class="container">
          <div class="row">
            <div class="col-sm-6 bd">
              <h3 class="h3">MELANOMA DETECTION</h3>
              <br />
            </div>
            <div class="col-sm-6">
              <div>
                <h4 class="h3">Upload Image Here To Identify the Disease</h4>
                <form
                  action="http://localhost:5000/"
                  id="upload-file"
                  method="post"
                  enctype="multipart/form-data"
                >
                  <label for="imageUpload" class="upload-label">
                    Choose...
                  </label>
                  <input
                    type="file"
                    name="image"
                    id="imageUpload"
                    accept=".png, .jpg, .jpeg"
                  />
                </form>
```

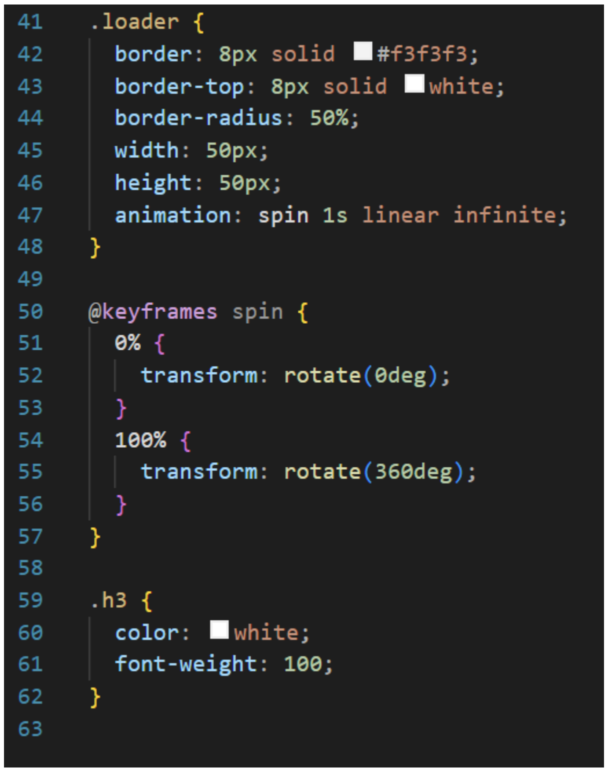


```html

                <div class="image-section" style="display: none">
                  <div class="img-preview">
                    <div id="imagePreview"></div>
                  </div>
                  <div>
                    <button type="button" class="upload-label" id="btn-predict">
                      Predict!
                    </button>
                  </div>
                </div>

                <div class="loader" style="display: none"></div>

                <h3 class="h3">
                  <span id="result"> </span>
                </h3>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </body>

  <footer>
    <script
      src="{{ url_for('static', filename='js/main.js') }}"
      type="text/javascript"
    ></script>
  </footer>
</html>

```

Path: Flask/static/css/main.css

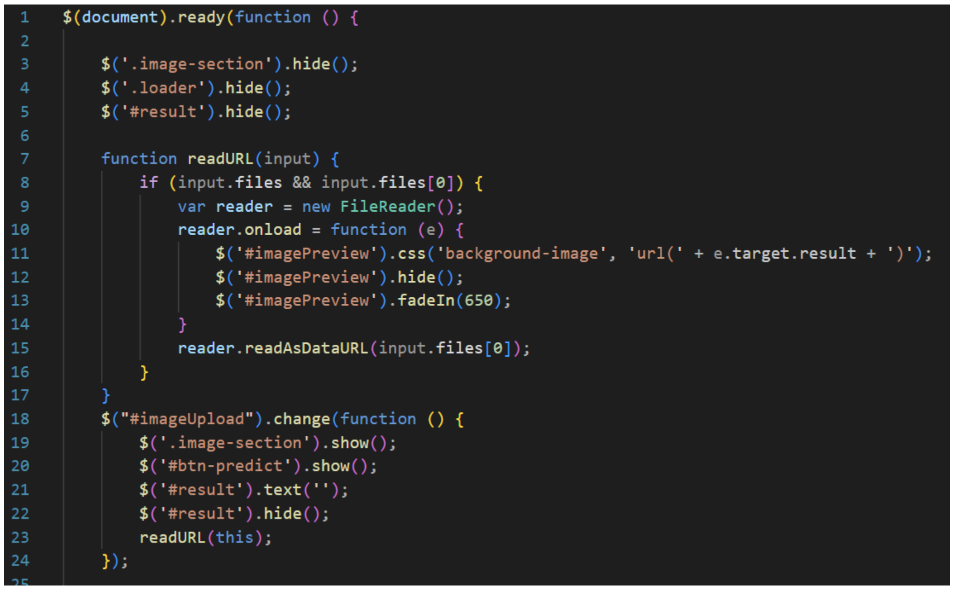


```css
.img-preview {
  width: 256px;
  height: 256px;
  position: relative;
  border: 1px solid #f8f8f8;
  box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
  margin-top: 1em;
  margin-bottom: 1em;
}

.img-preview > div {
  width: 100%;
  height: 100%;
  background-size: 256px 256px;
  background-repeat: no-repeat;
  background-position: center;
}

input[type="file"] {
  display: none;
}

.upload-label {
  margin: 10px;
  display: inline-block;
  padding: 12px 30px;
  background: none;
  color: #fff;
  border: 1px solid white;
  font-size: 1em;
  transition: all 0.4s;
  cursor: pointer;
  border-radius: 8px;
}

.upload-label:hover {
  background: grey;
  color: white;
}

```

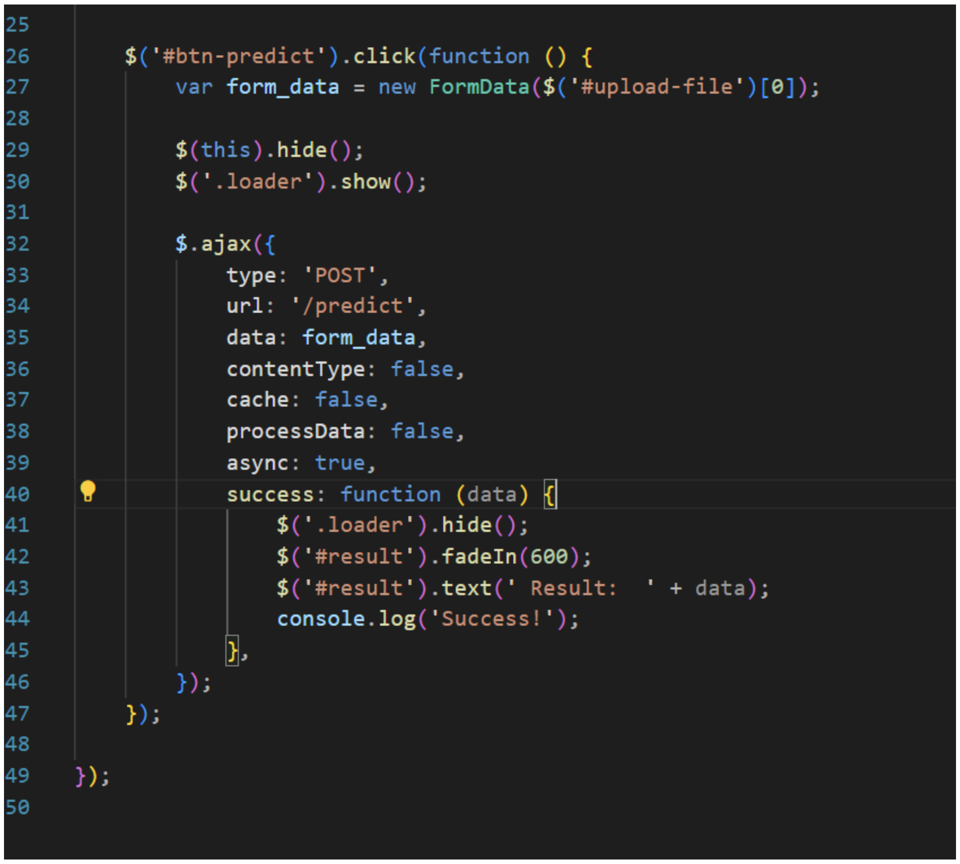


```css
.loader {
  border: 8px solid #f3f3f3;
  border-top: 8px solid white;
  border-radius: 50%;
  width: 50px;
  height: 50px;
  animation: spin 1s linear infinite;
}

@keyframes spin {
  0% {
    transform: rotate(0deg);
  }
  100% {
    transform: rotate(360deg);
  }
}

.h3 {
  color: white;
  font-weight: 100;
}

```

Path: Flask/static/js/main.js



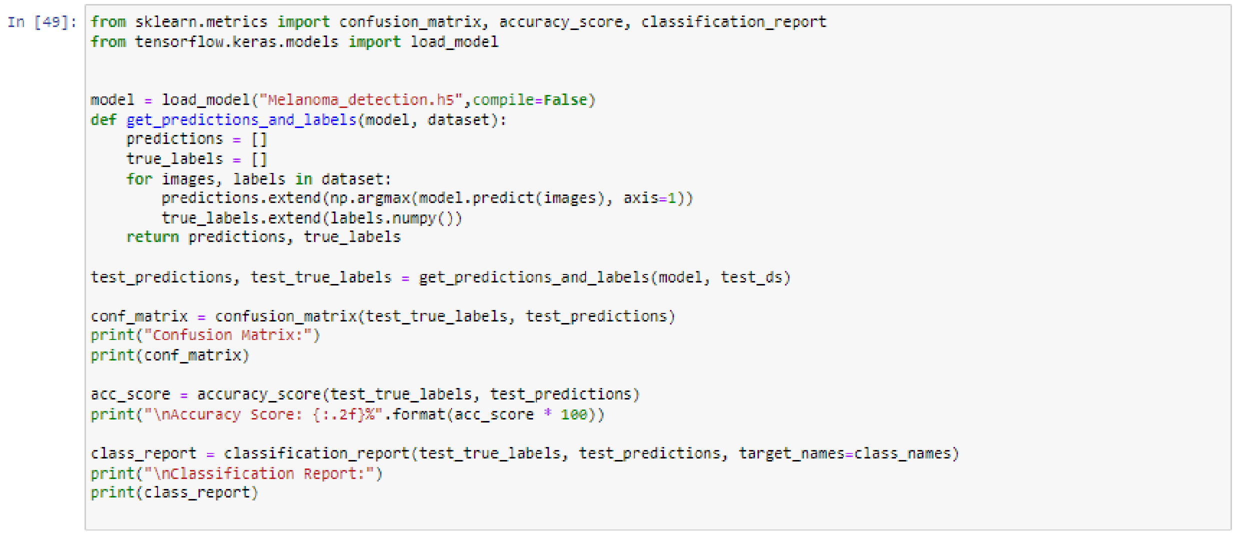
```javascript
$(document).ready(function () {

    $('.image-section').hide();
    $('.loader').hide();
    $('#result').hide();

    function readURL(input) {
        if (input.files && input.files[0]) {
            var reader = new FileReader();
            reader.onload = function (e) {
                $('#imagePreview').css('background-image', 'url(' + e.target.result + ')');
                $('#imagePreview').hide();
                $('#imagePreview').fadeIn(650);
            }
            reader.readAsDataURL(input.files[0]);
        }
    }
    $("#imageUpload").change(function () {
        $('.image-section').show();
        $('#btn-predict').show();
        $('#result').text('');
        $('#result').hide();
        readURL(this);
    });
```



```
    $('#btn-predict').click(function () {
        var form_data = new FormData($('#upload-file')[0]);

        $(this).hide();
        $('.loader').show();

        $.ajax({
            type: 'POST',
            url: '/predict',
            data: form_data,
            contentType: false,
            cache: false,
            processData: false,
            async: true,
            success: function (data) {
                $('.loader').hide();
                $('#result').fadeIn(600);
                $('#result').text(' Result:  ' + data);
                console.log('Success!');
            },
        });
    });

});

```

8. PERFORMANCE TESTING

8.1 Performance Metrics

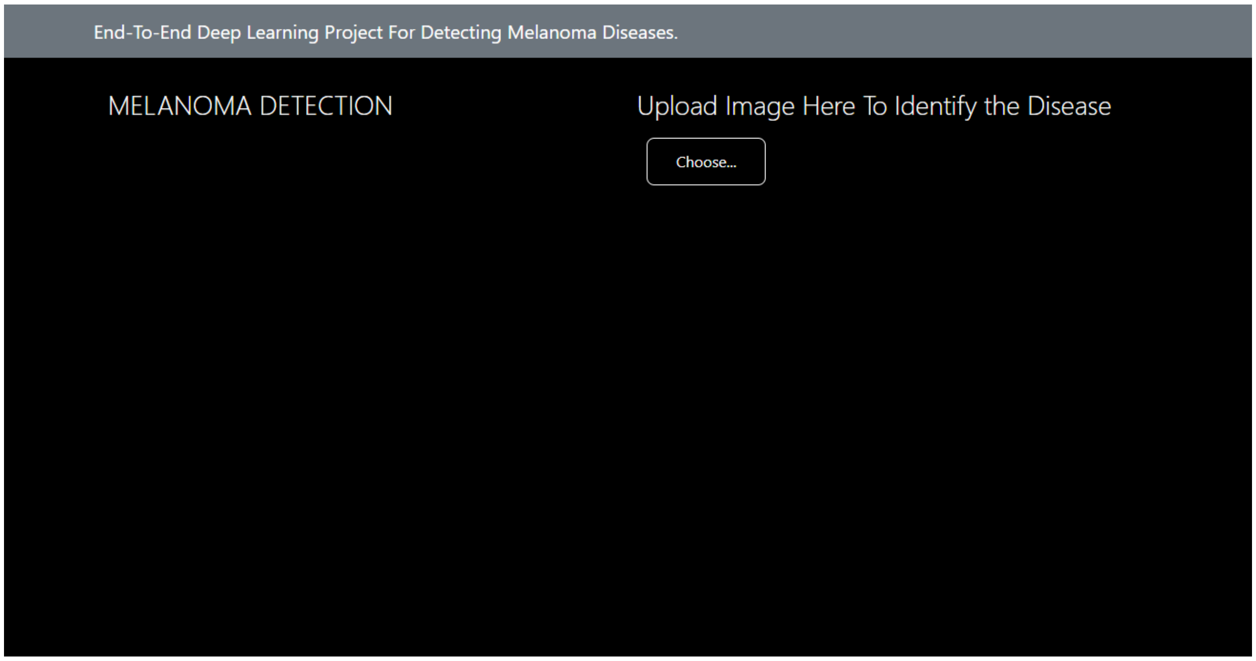


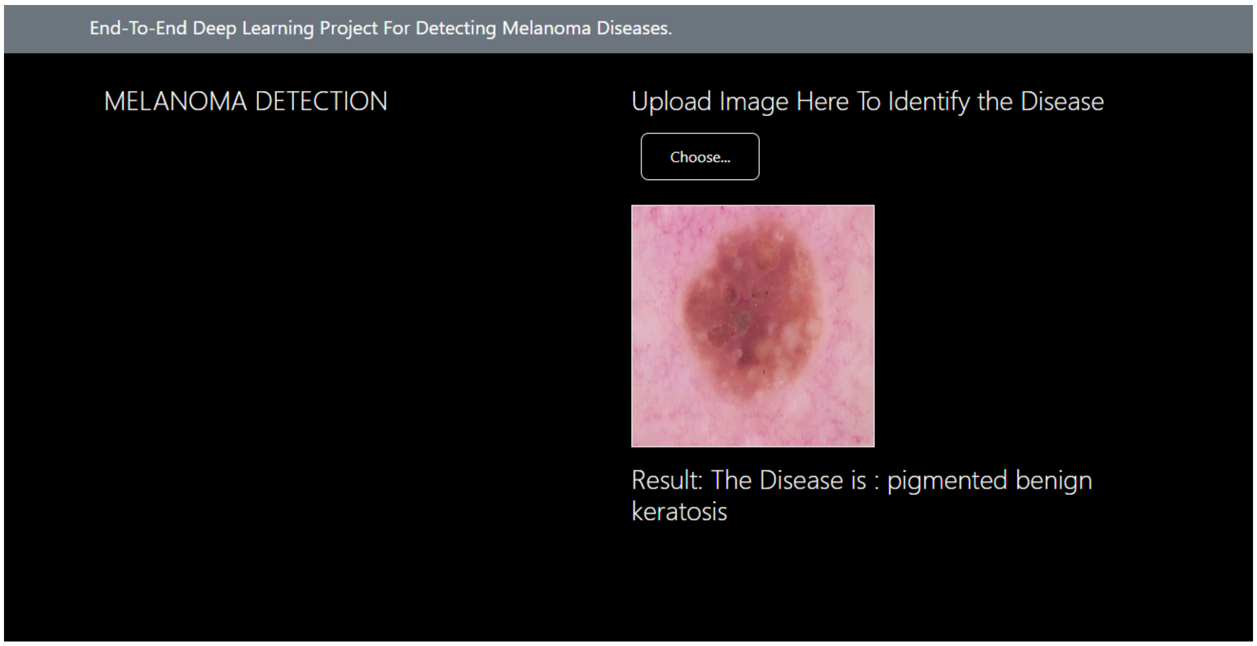
```
In [49]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
         from tensorflow.keras.models import load_model


         model = load_model("Melanoma_detection.h5",compile=False)
         def get_predictions_and_labels(model, dataset):
             predictions = []
             true_labels = []
             for images, labels in dataset:
                 predictions.extend(np.argmax(model.predict(images), axis=1))
                 true_labels.extend(labels.numpy())
             return predictions, true_labels

         test_predictions, test_true_labels = get_predictions_and_labels(model, test_ds)

         conf_matrix = confusion_matrix(test_true_labels, test_predictions)
         print("Confusion Matrix:")
         print(conf_matrix)

         acc_score = accuracy_score(test_true_labels, test_predictions)
         print("\nAccuracy Score: {:.2f}%".format(acc_score * 100))

         class_report = classification_report(test_true_labels, test_predictions, target_names=class_names)
         print("\nClassification Report:")
         print(class_report)
```

```
Confusion Matrix:
[[ 1  0  0  0 12  1  0  0  0]
 [ 3  9  0  0  0  2  0  1  0]
 [ 0  5  3  3  1  2  0  1  0]
 [ 0  0  0  2  8  3  0  0  0]
 [ 0  0  0  1 13  2  0  0  0]
 [ 0  2  2  1  1  5  0  1  0]
 [ 0  0  0  3  0  0  0  0  0]
 [ 0  2  2  3  1  4  0  4  0]
 [ 0  0  0  0  0  0  0  0  2]]

Accuracy Score: 36.79%

Classification Report:
                            precision    recall  f1-score   support

        actinic keratosis       0.25      0.07      0.11        14
     basal cell carcinoma       0.50      0.60      0.55        15
           dermatofibroma       0.43      0.20      0.27        15
                 melanoma       0.15      0.15      0.15        13
                    nevus       0.36      0.81      0.50        16
pigmented benign keratosis       0.26      0.42      0.32        12
     seborrheic keratosis       0.00      0.00      0.00         3
  squamous cell carcinoma       0.57      0.25      0.35        16
          vascular lesion       1.00      1.00      1.00         2

                 accuracy                           0.37       106
                macro avg       0.39      0.39      0.36       106
             weighted avg       0.37      0.37      0.33       106
```

9. RESULTS

9.1 Output Screenshots


End-To-End Deep Learning Project For Detecting Melanoma Diseases.
MELANOMA DETECTION
Upload Image Here To Identify the Disease
Choose...


End-To-End Deep Learning Project For Detecting Melanoma Diseases.
MELANOMA DETECTION
Upload Image Here To Identify the Disease
Choose...

Result: The Disease is : pigmented benign keratosis

10. ADVANTAGES & DISADVANTAGES

10.1 Advantages

- CNNs automatically learn hierarchical features from the input images, allowing them to capture intricate patterns and representations useful for melanoma detection.
- CNNs maintain the spatial hierarchy of features, recognizing local patterns and gradually combining them to understand global structures.
- Pre-trained CNN models on large datasets can be fine-tuned for melanoma detection with smaller datasets.
- CNNs can be trained with augmented data (rotated, flipped, zoomed) to improve model generalization.

10.2 Disadvantages

- CNNs require large amounts of labeled data for effective training.
- Training CNNs can be computationally expensive and time-consuming.
- CNNs are often considered as "black box" models, making it challenging to interpret their decision-making process.

- CNNs may produce false positives or false negatives in certain cases.
- Explaining why the CNN made a specific decision is challenging.
- High-performance GPUs are often required for efficient training and inference.

11. CONCLUSION

In this end-to-end deep learning project aimed at detecting melanoma diseases, we have successfully developed and implemented a robust system leveraging Convolutional Neural Networks (CNNs). The project involved various stages, including data collection, preprocessing, model development, training, and evaluation. While the results are encouraging, it is essential to acknowledge certain challenges, including the need for extensive and diverse datasets for optimal model training.

12. APPENDIX

12.1 Github Link:

https://github.com/smartinternz02/SI-GuidedProject-615389-1700490341

12.2 Demo Link:

https://drive.google.com/file/d/1_ceviG8uYmcY9N6GwOaRg__-L_G4KQ58/view?usp=sharing