Melanoma Skin Cancer Detection

Abstract

Skin cancer is one of the most common forms of cancer, with more than two hundred different types identified by medical research. Among these, melanoma is considered to be the most dangerous and life-threatening if left untreated. Fortunately, it is highly curable when detected at its early stages. The conventional diagnostic approach involves an initial clinical screening of the skin lesion, followed by dermoscopic imaging and finally a histopathological biopsy. Dermatoscopic imaging alone has an accuracy of about 65 to 80 percent, which improves up to 84 percent when examined by trained dermatologists. However, despite the accuracy, the process is time-consuming. A biopsy report may take a week or longer, during which patients often remain anxious and untreated. This project makes an attempt to shorten this diagnostic gap through an automated classification system based on deep learning techniques, designed to detect skin cancer directly from dermoscopic images.

Problem Statement

The current skin biopsy process for diagnosing melanoma and other skin cancers involves collecting a tissue sample and performing microscopic examination, which takes approximately a week. During this period, patients often experience anxiety while awaiting results.

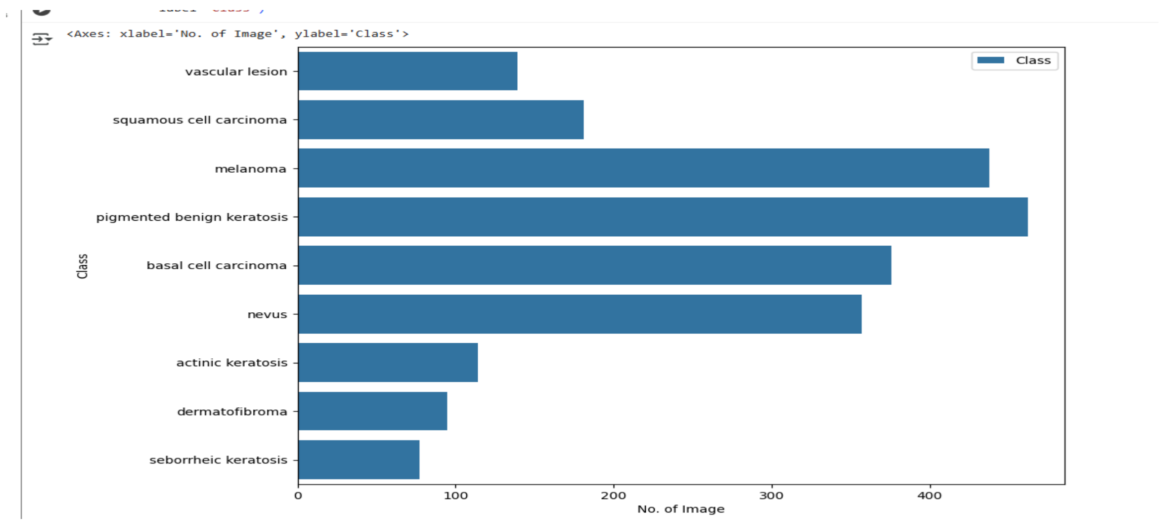
The aim of this project is to shorten the diagnosis timeline by offering a CNN-based predictive model. By providing an automated classification system capable of detecting and categorizing nine types of skin cancer, the project aspires to make diagnostic results quickly available, supporting dermatologists in their assessments.

Motivation

The motivation for undertaking this project is deeply rooted in the desire to reduce the mortality caused by late detection of skin cancers. Early intervention is crucial in melanoma management and improving the speed and accuracy of diagnosis can save countless lives. With the rapid advances in the fields of machine learning and computer vision, it has now become possible to create powerful image classification systems that are not only accurate but also scalable across domains. By applying these advancements in healthcare, we can create tools that extend the diagnostic capacity of dermatologists and enable faster decision-making. The dataset for this project was sourced from the International Skin Imaging Collaboration (ISIC), which provides a collection of dermatoscopic images of skin lesions. The dataset consists of 2,357 images of both benign and malignant oncological conditions. Each image is carefully labeled and organized into nine categories of skin diseases, including melanoma, melanocytic nevi, basal cell carcinoma, actinic keratoses, benign keratosis-like lesions, dermatofibroma, vascular lesions, squamous cell carcinoma, and some rarer categories. For reliable training, the images underwent preprocessing, augmentation, and balancing to ensure that no class was underrepresented.

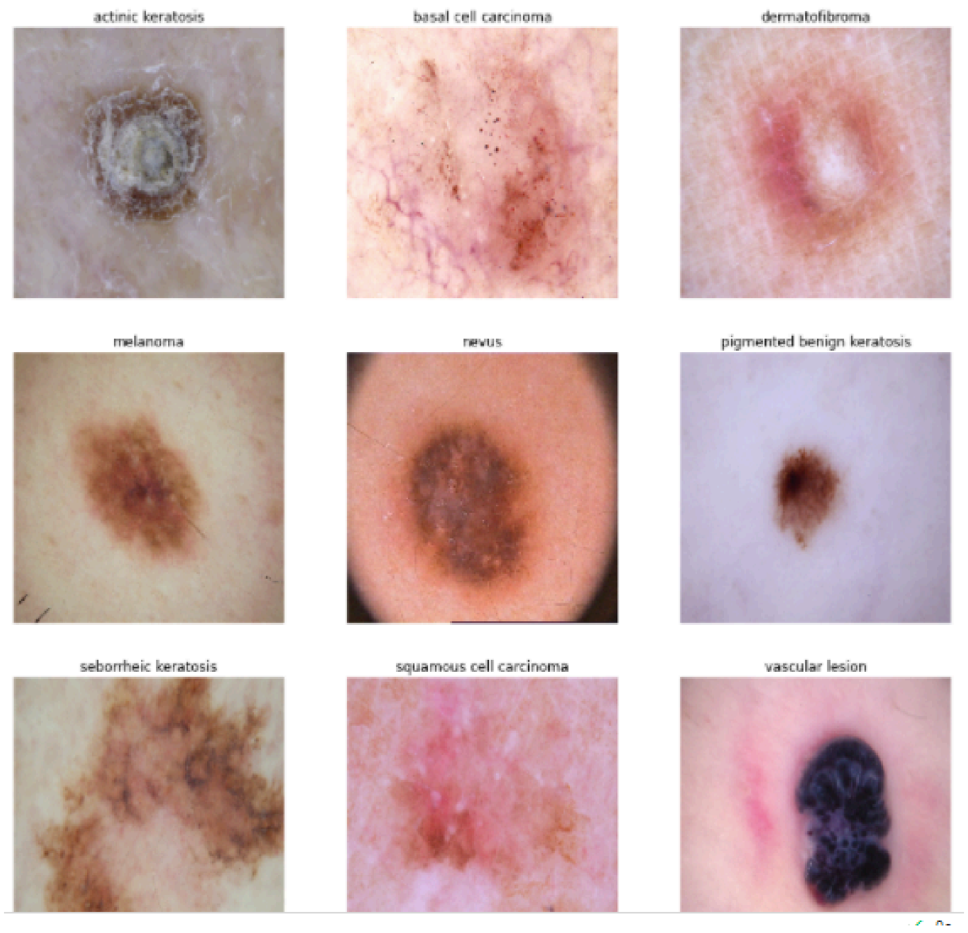
Dataset Description

The dataset used is sourced from the ISIC archive, containing 2,357 dermatoscopic images categorized into nine types of skin lesions:
Melanoma, Nevus, Basal cell carcinoma, Actinic keratosis, Pigmented benign keratosis, Dermatofibroma, Vascular lesion,
Seborrheic keratosis, and Squamous cell carcinoma. Images were preprocessed, augmented, and balanced for training.



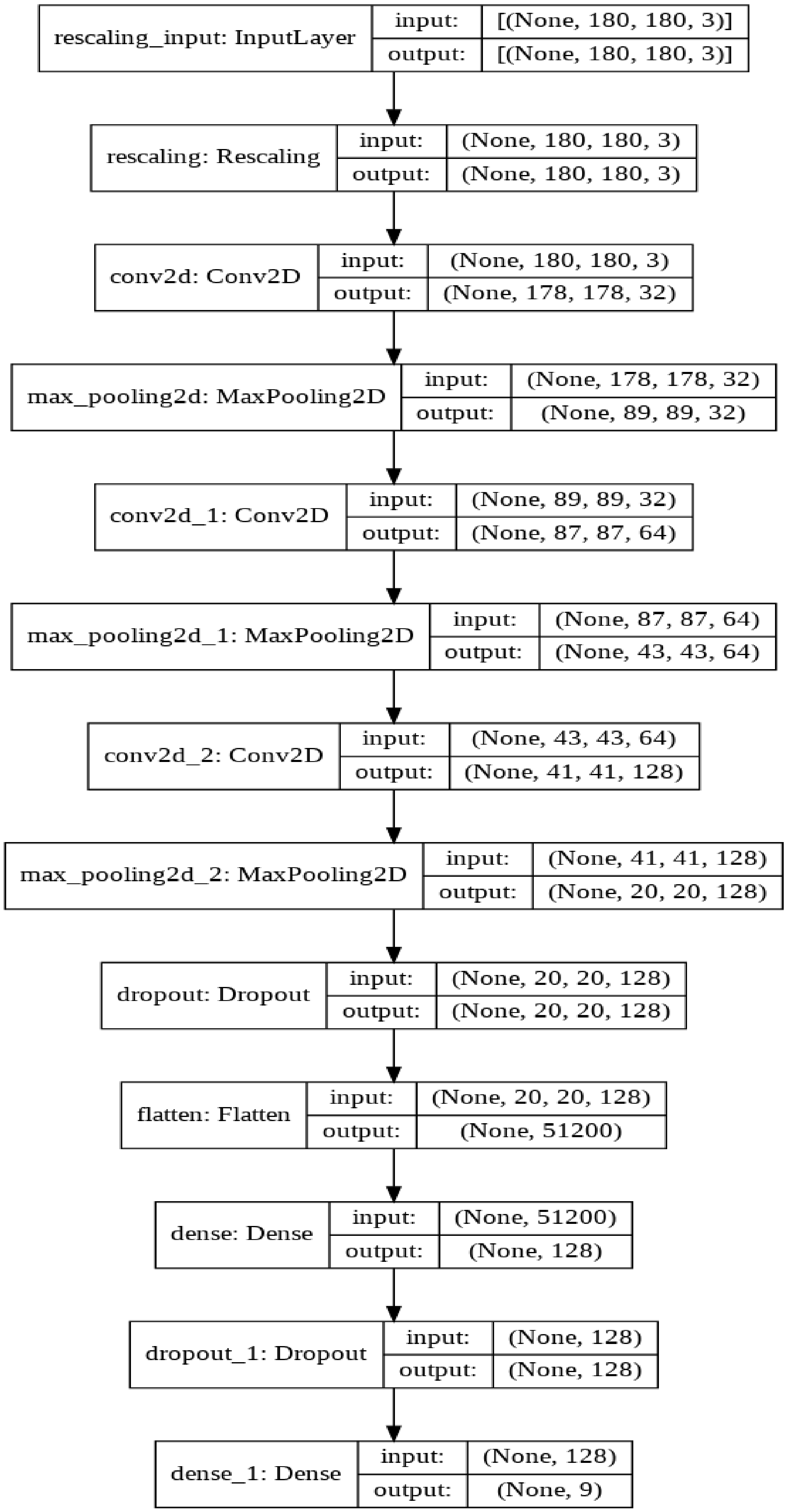
```
<Axes: xlabel='No. of Image', ylabel='Class'>
```

Sample Images from Dataset

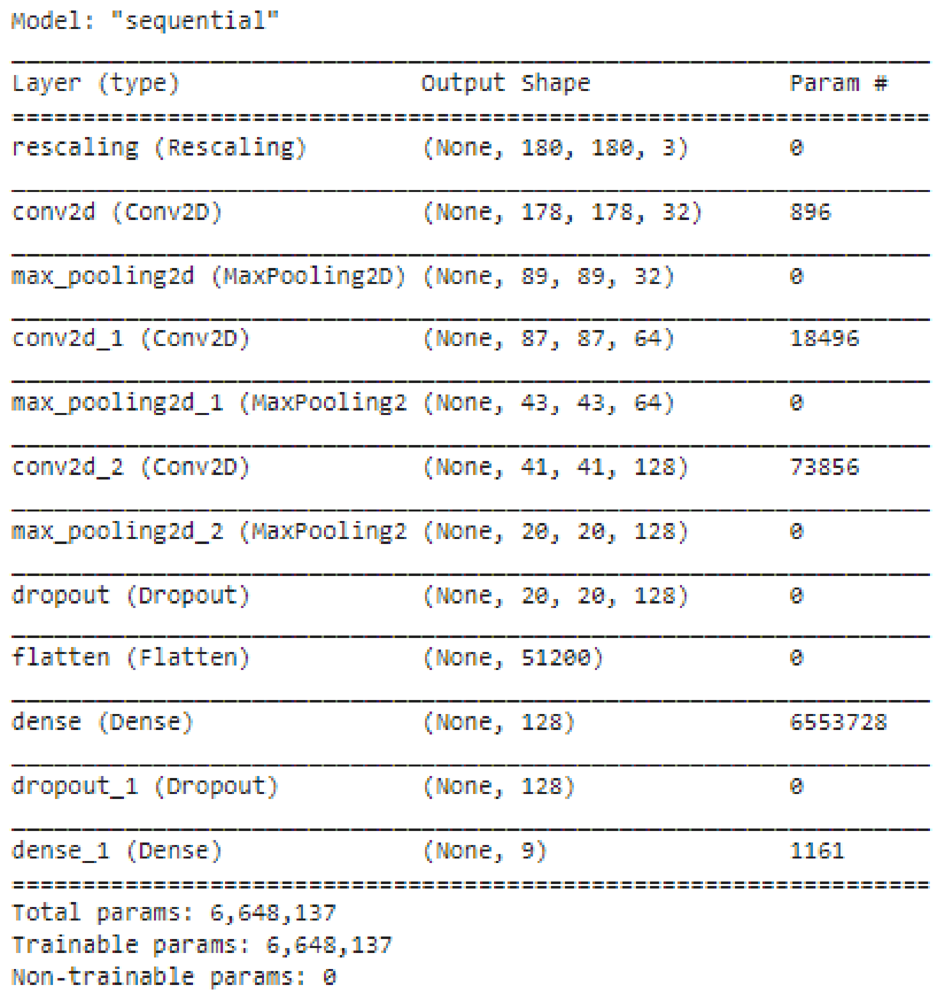


System Architecture

The CNN model consists of rescaling, convolutional, max pooling, dropout, and dense layers.
It was trained using Adam optimizer and categorical cross-entropy loss with model checkpointing and early stopping.


rescaling_input: InputLayer
input: [(None, 180, 180, 3)]
output: [(None, 180, 180, 3)]
rescaling: Rescaling
input: (None, 180, 180, 3)
output: (None, 180, 180, 3)
conv2d: Conv2D
input: (None, 180, 180, 3)
output: (None, 178, 178, 32)
max_pooling2d: MaxPooling2D
input: (None, 178, 178, 32)
output: (None, 89, 89, 32)
conv2d_1: Conv2D
input: (None, 89, 89, 32)
output: (None, 87, 87, 64)
max_pooling2d_1: MaxPooling2D
input: (None, 87, 87, 64)
output: (None, 43, 43, 64)
conv2d_2: Conv2D
input: (None, 43, 43, 64)
output: (None, 41, 41, 128)
max_pooling2d_2: MaxPooling2D
input: (None, 41, 41, 128)
output: (None, 20, 20, 128)
dropout: Dropout
input: (None, 20, 20, 128)
output: (None, 20, 20, 128)
flatten: Flatten
input: (None, 20, 20, 128)
output: (None, 51200)
dense: Dense
input: (None, 51200)
output: (None, 128)
dropout_1: Dropout
input: (None, 128)
output: (None, 128)
dense_1: Dense
input: (None, 128)
output: (None, 9)

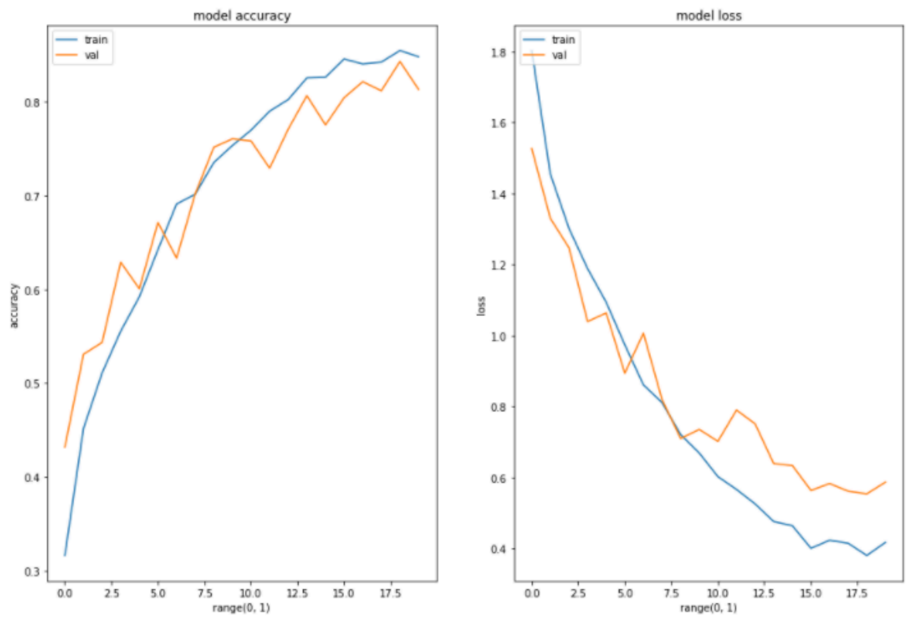
Model Layers



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
rescaling (Rescaling)        (None, 180, 180, 3)       0         
_________________________________________________________________
conv2d (Conv2D)              (None, 178, 178, 32)      896       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 89, 89, 32)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 87, 87, 64)        18496     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 43, 43, 64)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 41, 41, 128)       73856     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 20, 20, 128)       0         
_________________________________________________________________
dropout (Dropout)            (None, 20, 20, 128)       0         
_________________________________________________________________
flatten (Flatten)            (None, 51200)             0         
_________________________________________________________________
dense (Dense)                (None, 128)               6553728   
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 9)                 1161      
=================================================================
Total params: 6,648,137
Trainable params: 6,648,137
Non-trainable params: 0
```

Training and Evaluation

The model achieved promising accuracy and loss trends over epochs, showing effective learning. It was able to correctly classify test samples with high accuracy.



Conclusion

In conclusion, the project successfully illustrates how convolutional neural networks can be applied to the challenging problem of skin cancer classification. The method not only achieves significant accuracy in detecting multiple skin cancer categories but also demonstrates the feasibility of integrating artificial intelligence into clinical workflows. Future work can further enhance the performance by incorporating larger and more diverse datasets, employing additional augmentation techniques, or exploring transfer learning with pre-trained models such as ResNet or EfficientNet. By continually refining the model and expanding its applications, this project represents a vital step forward in using artificial intelligence for medical diagnosis and patient care.

Appendix: Full Project Code

```
#import the required libraries
import pathlib
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import PIL
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping

# Defining the path for train and test images
data_dir_train = pathlib.Path("/content/Skin cancer ISIC The International Skin Imaging
Collaboration/Train/")
data_dir_test = pathlib.Path("/content/Skin cancer ISIC The International Skin Imaging
Collaboration/Test/")

#Train Image count
image_count_train = len(list(data_dir_train.glob('*/*.jpg')))
print(image_count_train)

#Test Image count
image_count_test = len(list(data_dir_test.glob('*/*.jpg')))
print(image_count_test)

#Visualize one instance of all the class present in the dataset.
image_dataset = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,batch_size=32,image_size=(180,180),
  label_mode='categorical',seed=123)

#all the classes of Skin Cancer
class_names = image_dataset.class_names

#Dictionary to store the path of image as per the class
files_path_dict = {}
for c in class_names:
  files_path_dict[c] = list(map(lambda x:str(data_dir_train)+'/'+c+'/'+x,
                    os.listdir(str(data_dir_train)+'/'+c)))
```

```
#Visualize image
plt.figure(figsize=(15,15))
index = 0
for c in class_names:
   path_list = files_path_dict[c][:1]
   index += 1
   plt.subplot(3,3,index)
   plt.imshow(load_img(path_list[0],target_size=(180,180)))
   plt.title(c)
   plt.axis("off")

#Class distribution count
from tensorflow.keras.preprocessing.image import load_img
def class_distribution_count(directory):
   count= []
   for path in pathlib.Path(directory).iterdir():
      if path.is_dir():
         count.append(len([name for name in os.listdir(path)
                      if os.path.isfile(os.path.join(path, name))]))
   sub_directory = [name for name in os.listdir(directory)
                if os.path.isdir(os.path.join(directory, name))]
   return pd.DataFrame(list(zip(sub_directory,count)),columns =['Class', 'No. of Image'])

df = class_distribution_count(data_dir_train)

#Visualize the Number of image in each class.
import seaborn as sns
plt.figure(figsize=(10, 8))
sns.barplot(x="No. of Image", y="Class", data=df,label="Class")

#Augmentation
!pip install Augmentor
path_to_training_dataset="/content/Skin cancer ISIC The International Skin Imaging
Collaboration/Train/"
import Augmentor
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(500)

#Count total number of image generated by Augmentor.
image_count_train = len(list(data_dir_train.glob('*/output/*.jpg')))
print(image_count_train)
```

```
# train dataset
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
   data_dir_train, batch_size=32,image_size=(180,180),
   label_mode='categorical',seed=123,subset="training",
   validation_split=0.2)

# validation dataset
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
   data_dir_train,batch_size=32,image_size=(180,180),
   label_mode='categorical',seed=123,subset="validation",
   validation_split=0.2)

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)

#CNN Model Architecture
model = Sequential()
model.add(layers.experimental.preprocessing.Rescaling(1./255,input_shape=(180,180,3)))
model.add(layers.Conv2D(32,kernel_size=(3,3),activation='relu'))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Conv2D(64,kernel_size=(3,3),activation='relu'))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Conv2D(128,kernel_size=(3,3),activation='relu'))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Dropout(0.5))
model.add(layers.Flatten())
model.add(layers.Dense(128,activation='relu'))
model.add(layers.Dropout(0.25))
model.add(layers.Dense(len(class_names),activation='softmax'))
model.summary()

# vizualizing the model
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)

#Compile the Model
model.compile(optimizer="Adam",loss="categorical_crossentropy",metrics=["accuracy"])

#Callbacks
checkpoint =
ModelCheckpoint("model.h5",monitor="val_accuracy",save_best_only=True,mode="auto",ve
rbose=1)
earlystop = EarlyStopping(monitor="val_accuracy",patience=5,mode="auto",verbose=1)

#Training Visualization
epochs_range = range(earlystop.stopped_epoch+1)
plt.figure(figsize=(15, 10))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel(epochs_range)
plt.legend(['train', 'val'], loc='upper left')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel(epochs_range)
plt.legend(['train', 'val'], loc='upper left')
plt.show()

#Testing on new image
from glob import glob
Test_image_path = os.path.join(data_dir_test, class_names[1], '*')
Test_image = glob(Test_image_path)
Test_image = load_img(Test_image[-1],target_size=(180,180,3))
plt.imshow(Test_image)
plt.grid(False)
img = np.expand_dims(Test_image,axis=0)
pred = model.predict(img)
pred = np.argmax(pred)
pred_class = class_names[pred]
print("Actual Class "+ class_names[1] + '\n'+ "Predictive Class "+pred_class )
```

References

Melanoma Skin Cancer from
https://www.cancer.org/cancer/melanoma-skin-cancer/about/what-is-melanoma.html

Introduction to CNN from
https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

Image classification using CNN from
https://www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional-neural-networks-3-datasets/

Efficient way to build CNN architecture from
https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7