Kevin Raj (ID# 11450471)
CSCE5216 – Pattern Recognition
7/18/2023

# Project Final Report: Skin Lesion Classifier

## Project Name

Skin Lesion Classifier

## Participant

Kevin Raj
Email: KevinRaj@my.unt.edu

## Google Colab Code Link

https://colab.research.google.com/drive/1w00ToyJUjhVmjCD-wqSGAsi5RhRuDTUx

## Workflow

I worked by myself for this project and spent some time each week to keep up with the project progress and final project deadline.

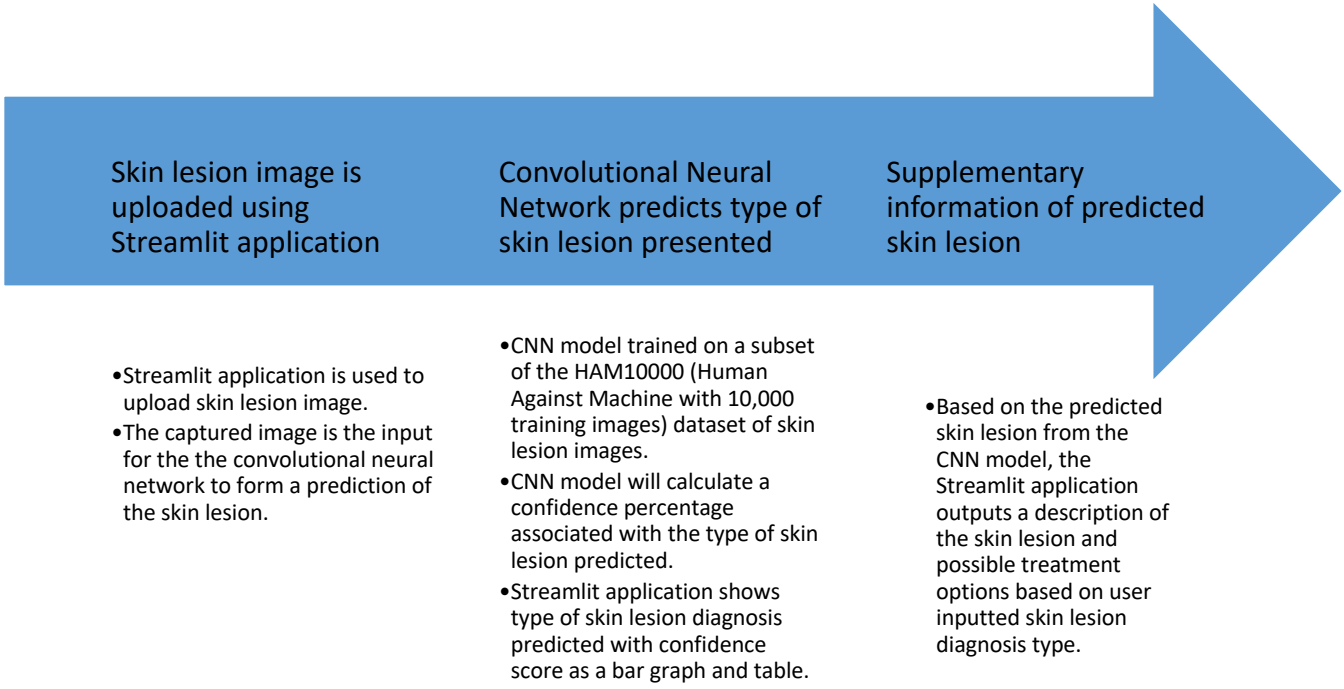
## Final Project Abstract

The objective of this project is to develop an image classification system using a convolutional neural network (CNN model) that can accurately identify skin lesion images. The classification system can be utilized to identify the skin lesion with a percentage indicating how confident the classification system is of observing a particular type of skin lesion presented. Dermatologists use their specialized knowledge and experience to identify suspicious lesions on patients during annual skin exams or patient visits. The dermatologist can monitor the lesion over time to detect any changes or opt to surgically remove the lesion if the lesion could be harmful to the patient. If the dermatologist is suspicious of the lesion, the dermatologist may remove a part of the lesion and send the sample to a pathologist for further testing to determine if the lesion is benign (non-cancerous) or malignant (cancerous). This type of system could hypothetically assist dermatologists in detecting potential skin cancer or other skin diseases, but this iteration of the project is not intended for actual medical use or treatment due to limitations with the model accuracy and model training.

Another component that this proposed solution can improve is patient education through written notes. Dermatologists provide written notes to their patients on the diagnosis, any medicines to take, and the treatment plan based on the skin lesion observed. These notes can often be time-consuming and can strain the resources of the dermatologist preventing them

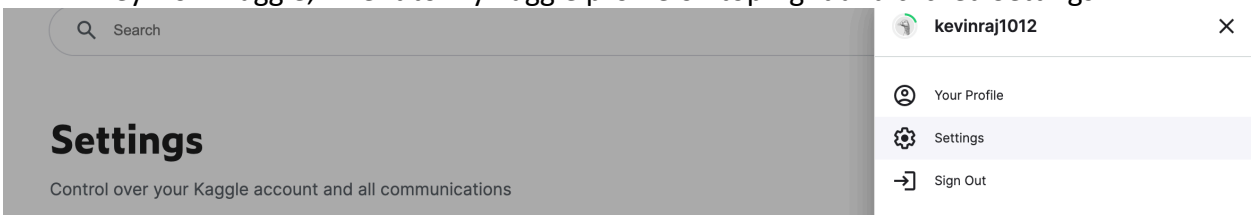
from fully attending to patients. To assist with this administrative part of the patient education, the skin lesion classifier outputs a detailed description of the predicted skin lesion type and more information on possible treatment options for the patient.

## Project Design

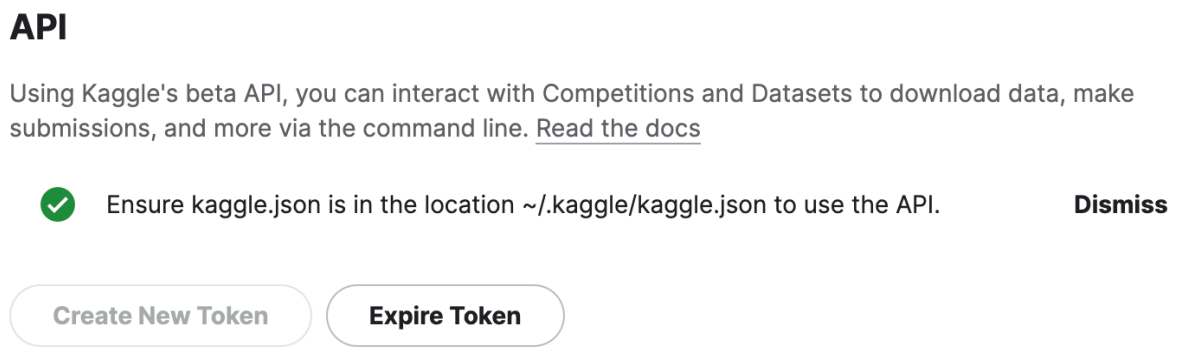



## Step 1: Load the HAM10000 Dataset

- First step, I imported the HAM10000 (Humans Against Machine with 10000 training images) dataset from Kaggle to Google Colab. To import the data, I mounted the Google Drive, generated a Kaggle API Key, uploaded the Kaggle API Key to Google Colab, downloaded the dataset from Kaggle as a zip file, and unzipped the file and stored the zip file contents in Google Drive to be accessed by Google Colab. To download the API key from Kaggle, I went to my Kaggle profile on top right and clicked Settings:



- I then clicked ‘Create New Token’ and saved the kaggle.json file on my computer and uploaded the kaggle.json file to Google Colab:




```
# Upload Kaggle API Key (code from tutorial: https://www.kaggle.com/code/kalaikumarr/google-colab-kaggle-api)

from google.colab import files
files.upload()
```

- Here is the link for the dataset: https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000, and here is the code to perform these steps to upload the HAM10000 dataset to Google Colab (code from tutorials:
  - https://colab.research.google.com/notebooks/io.ipynb
  - https://www.kaggle.com/code/kalaikumarr/google-colab-kaggle-api
  - https://www.geeksforgeeks.org/unzipping-files-in-python/ )

```
# Mount Google Drive (from tutorial: https://colab.research.google.com/notebooks/io.ipynb)

from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
# Install Kaggle python library

!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.5.15)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle) (2023.5.7)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.27.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.65.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.26.16)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.0.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.4)
```

```
# Upload Kaggle API Key (code from tutorial: https://www.kaggle.com/code/kalaikumarr/google-colab-kaggle-api)

from google.colab import files
files.upload()
```

```
# Move API Key (code from tutorial: https://www.kaggle.com/code/kalaikumarr/google-colab-kaggle-api)

!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
# Download Skin Cancer MNIST: HAM10000 dataset to Google Drive (code from tutorial: https://www.kaggle.com/code/kalaikumarr/google-colab-kaggle-api)

!kaggle datasets download -d kmader/skin-cancer-mnist-ham10000 -p /content/drive/MyDrive/Datasets/SkinCancerMNIST
```

```
Downloading skin-cancer-mnist-ham10000.zip to /content/drive/MyDrive/Datasets/SkinCancerMNIST
100% 5.19G/5.20G [01:19<00:00, 81.7MB/s]
100% 5.20G/5.20G [01:19<00:00, 70.5MB/s]
```

```
# Extract the Skin Cancer MNIST: HAM10000 dataset zip file (code from tutorial: https://www.geeksforgeeks.org/unzipping-files-in-python/)

import zipfile

# Path to the downloaded zip file of HAM10000 dataset
downloaded_path = "/content/drive/MyDrive/Datasets/SkinCancerMNIST/skin-cancer-mnist-ham10000.zip"

# Path to extract the skin lesion images in HAM10000 dataset
extracted_path = "/content/drive/MyDrive/Datasets/SkinCancerMNIST/images"

# Extract the zip file of HAM10000 dataset
with zipfile.ZipFile(downloaded_path, 'r') as zipped_files:
    zipped_files.extractall(extracted_path)
```

- We can connect to Google TPU's for faster model processing using this code (code from tutorial: https://colab.research.google.com/notebooks/tpu.ipynb#scrollTo=FpvUOuC3j27n)

Run on Google Colab TPU

```
# from code tutorial: https://colab.research.google.com/notebooks/tpu.ipynb#scrollTo=FpvUOuC3j27n

import tensorflow as tf
print("Tensorflow version " + tf.__version__)

try:
  tpu = tf.distribute.cluster_resolver.TPUClusterResolver()  # TPU detection
  print('Running on TPU ', tpu.cluster_spec().as_dict()['worker'])
except ValueError:
  raise BaseException('ERROR: Not connected to a TPU runtime; please see the previous cell in this notebook for instructions!')

tf.config.experimental_connect_to_cluster(tpu)
tf.tpu.experimental.initialize_tpu_system(tpu)
tpu_strategy = tf.distribute.TPUStrategy(tpu)
```

```
Tensorflow version 2.12.0
Running on TPU  ['10.104.240.10:8470']
```

- To get a better understanding of the diagnosis codes of the skin lesions in the dataset, we can use metadata csv file to create a dictionary mapping with the diagnosis code as the key and the long form of the diagnosis of the skin lesion as the value. For example, in the metadata csv file - the diagnosis code is in the field 'dx' and one type of diagnosis code is bkl representing benign keratosis as the diagnosis of the skin lesion. Here is the code to create the dictionary mapping, and we can observe the first five records in the Pandas dataframe (code from Kaggle code tutorial: https://www.kaggle.com/code/xinruizhuang/skin-lesion-classification-acc-90-pytorch):

```
# Match the image data with the skin lesion type in CSV file (code from tutorial: https://www.kaggle.com/code/xinruizhuang/skin-lesion-classification-acc-90-pytorch)

from glob import glob
import pandas as pd
import os

data_dir = '/content/drive/MyDrive/Datasets/SkinCancerMNIST/images'
all_image_path = glob(os.path.join(data_dir, '*', '*.jpg'))
imageid_path_dict = {os.path.splitext(os.path.basename(x))[0]: x for x in all_image_path}
lesion_type_dict = {
    'nv': 'melanocytic nevi',
    'mel': 'melanoma',
    'bkl': 'benign keratosis',
    'bcc': 'basal cell carcinoma',
    'akiec': 'actinic keratoses',
    'vasc': 'vascular lesions',
    'df': 'dermatofibroma'
}

# Assign lesion type to metadata csv

df_original = pd.read_csv(os.path.join(data_dir, 'HAM10000_metadata.csv'))
df_original['path'] = df_original['image_id'].map(imageid_path_dict.get)
df_original['cell_type'] = df_original['dx'].map(lesion_type_dict.get)
df_original['cell_type_idx'] = pd.Categorical(df_original['cell_type']).codes
df_original.head()
```

| | lesion_id | image_id | dx | dx_type | age | sex | localization | path | cell_type | cell_type_idx |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | HAM_0000118 | ISIC_0027419 | bkl | histo | 80.0 | male | scalp | /content/drive/MyDrive/Datasets/SkinCancerMNIS... | benign keratosis | 2 |
| 1 | HAM_0000118 | ISIC_0025030 | bkl | histo | 80.0 | male | scalp | /content/drive/MyDrive/Datasets/SkinCancerMNIS... | benign keratosis | 2 |
| 2 | HAM_0002730 | ISIC_0026769 | bkl | histo | 80.0 | male | scalp | /content/drive/MyDrive/Datasets/SkinCancerMNIS... | benign keratosis | 2 |
| 3 | HAM_0002730 | ISIC_0025661 | bkl | histo | 80.0 | male | scalp | /content/drive/MyDrive/Datasets/SkinCancerMNIS... | benign keratosis | 2 |
| 4 | HAM_0001466 | ISIC_0031633 | bkl | histo | 75.0 | male | ear | /content/drive/MyDrive/Datasets/SkinCancerMNIS... | benign keratosis | 2 |

## Step 2: Exploratory Data Analysis

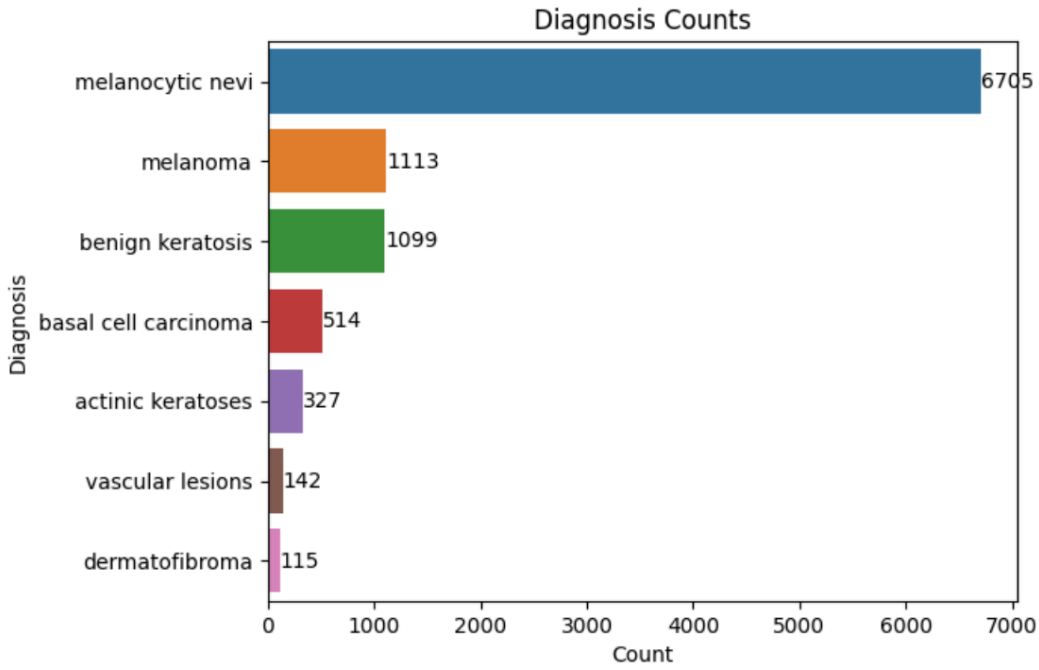
- We can then perform Exploratory Data Analysis on the dataset to observe the distribution of the diagnosis of the skin lesions to determine if there are any class imbalances in the dataset. Class imbalances will impact the accuracy of the convolutional neural network (CNN) model and could lead to less accurate results.

- We can use seaborn and matplotlib libraries in Python to observe the number of skin lesions by diagnosis type. Here is the code to use the dictionary mapping we created in the previous step to map the diagnosis code with the long form diagnosis name and observe the counts by diagnosis type. We use seaborn and matplotlib to create a horizontal bar chart and matplotlib to add a title, axis labels and display the chart (code from tutorial: https://www.statology.org/seaborn-horizontal-barplot/)

## Step 2: Exploratory Data Analysis

### Show the counts by diagnosis of skin lesion

```
# code from tutorial: https://www.statology.org/seaborn-horizontal-barplot/

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

dataset_diagnosis_counts = df_original['dx'].map(lesion_type_dict).value_counts()

# Create a bar chart using Seaborn
ax = sns.barplot(x=dataset_diagnosis_counts.values, y=dataset_diagnosis_counts.index)

# Show the bar chart title and axis labels
plt.title('Diagnosis Counts')
plt.xlabel('Count')
plt.ylabel('Diagnosis')

ax.bar_label(ax.containers[0])

# Display the bar chart
plt.show()
```



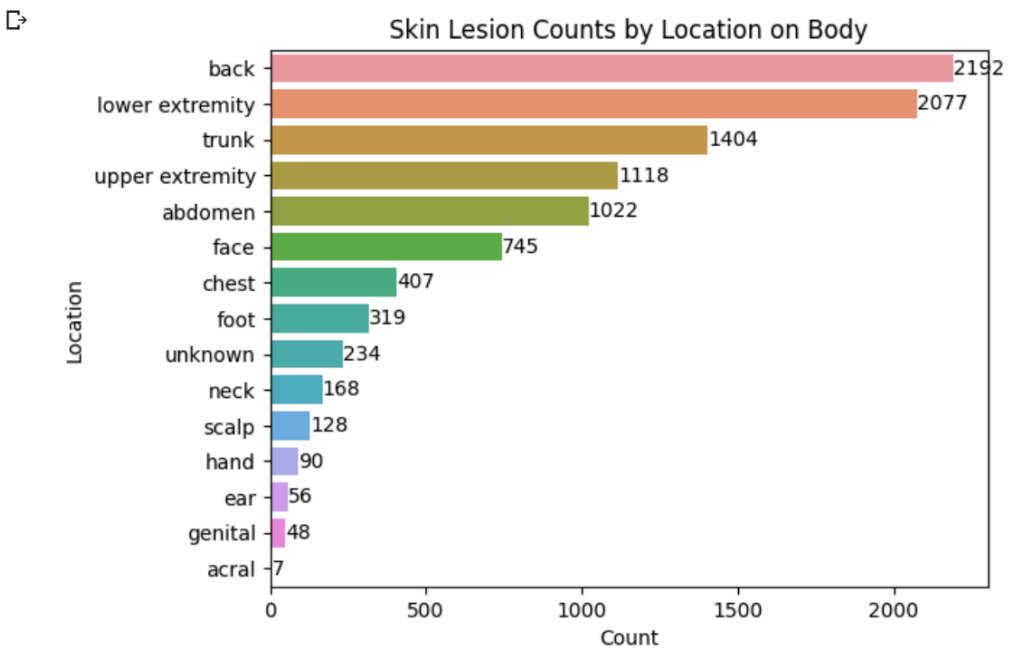

We can observe there is a big class imbalance in the dataset with the skin lesion diagnosis type of melanocytic nevi representing nearly 67% of the dataset.

- An important characteristic to observe in skin lesions is where on the body the skin lesion was found. We can observe the distribution of where the skin lesion was found on

the body using seaborn and matplotlib. The field where the skin lesion was found on the body is represented by 'localization' in the metadata csv file. Here is the code to create the horizontal bar chart showing the localization distribution of the skin lesions (code from tutorial: https://www.statology.org/seaborn-horizontal-barplot/:

- Show the counts by location of skin lesion

```
# code from tutorial: https://www.statology.org/seaborn-horizontal-barplot/

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

dataset_skin_lesion_counts = df_original['localization'].value_counts()

# Create a horizontal bar chart using Seaborn
ax = sns.barplot(x=dataset_skin_lesion_counts.values, y=dataset_skin_lesion_counts.index)

# Show the bar chart title and axis labels
plt.title('Skin Lesion Counts by Location on Body')
plt.xlabel('Count')
plt.ylabel('Location')

ax.bar_label(ax.containers[0])

# Display the bar chart
plt.show()
```




We can observe around 43% of the skin lesions were observed either on the back or lower extremity (legs, feet, toes, etc).

## Step 3: Show Plots, Assign Classes to the Data, and Train the Model

- The next step, we will label encode the types of skin lesion diagnosis from the metadata csv file from 0-6 with each number representing a type of skin lesion diagnosis. We can use the fit() method from the LabelEncoder class from sci-kit learn library to assign a numerical value to each unique skin lesion diagnosis type from the 'dx' column in the metadata csv file. Label encoding the skin lesion diagnosis types and assigning a numerical value to each skin lesion diagnosis type can allow the convolutional neural network to train more efficiently and process the labels quicker than using text labels. Here is the code to label encode the skin lesion diagnosis types and assign a dimension size of 224 by 224 pixels for resizing the images (code from tutorial: https://www.youtube.com/watch?v=fyZ9Rxpoz2I and https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py):

```
# Import Libraries and read metadata csv to assign classes to diagnosis codes (code from tutorial: https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classificat

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
from glob import glob
import seaborn as sns
from PIL import Image

np.random.seed(42)
from sklearn.metrics import confusion_matrix

import keras
from keras.utils.np_utils import to_categorical # for one-hot encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, BatchNormalization
from sklearn.model_selection import train_test_split
from scipy import stats
from sklearn.preprocessing import LabelEncoder

skin_lesion_df = pd.read_csv('/content/drive/MyDrive/Datasets/SkinCancerMNIST/images/HAM10000_metadata.csv')

# Set the image dimension size

SIZE=224

# Label encode the diagnosis type
label_encoder = LabelEncoder()
label_encoder.fit(skin_lesion_df['dx'])
print(list(label_encoder.classes_))

skin_lesion_df['label'] = label_encoder.transform(skin_lesion_df["dx"])
print(skin_lesion_df.sample(10))
```

```
['akiec', 'bcc', 'bkl', 'df', 'mel', 'nv', 'vasc']
         lesion_id      image_id   dx    dx_type   age     sex  \
1617  HAM_0007180  ISIC_0033272  mel      histo  65.0    male
8128  HAM_0007195  ISIC_0031923   nv      histo  40.0  female
2168  HAM_0001835  ISIC_0026652  mel      histo  65.0    male
1090  HAM_0000465  ISIC_0030583  bkl  consensus  35.0  female
7754  HAM_0001720  ISIC_0034010   nv      histo  45.0    male
8071  HAM_0006333  ISIC_0024424   nv      histo  35.0    male
7423  HAM_0004548  ISIC_0032832   nv      histo  45.0  female
8984  HAM_0006526  ISIC_0026671   nv      histo  55.0    male
2310  HAM_0003102  ISIC_0032389  mel      histo  65.0    male
7256  HAM_0004260  ISIC_0025525   nv      histo  65.0    male

          localization  label
1617              face      4
8128  lower extremity      5
2168              back      4
1090             trunk      2
7754          abdomen      5
8071             trunk      5
7423  upper extremity      5
8984  lower extremity      5
2310              face      4
7256              back      5
```

- In the next step, we will balance the number of images in each skin lesion diagnosis type. We observed the HAM10000 dataset has around 67% of skin lesions with diagnosis type of melanocytic nevi. We can rebalance the classes of each skin lesion diagnosis type to help the model reduce bias in favoring the majority class (melanocytic nevi) in making its predictions. This can help the model in training and allow it to make more accurate predictions on new skin lesion images. We can use the resample function from sci-kit learn utilities to assist with the rebalancing, and we can assign 700 images to each of the 7 skin lesion diagnosis types. We will create separate Pandas dataframes for each of the 7 skin lesion types with each skin lesion diagnosis type containing 700 images. We will then concatenate the 7 Pandas dataframes into a single Pandas dataframe and observe the distribution of the number of images from the single, concatenated dataframe to ensure each skin lesion diagnosis type has 700 images each. Here is the code to rebalance the skin lesion diagnosis types with 700 images, concatenate the 7 dataframes into a single dataframe, and observe the single dataframe to ensure each skin lesion diagnosis type has 700 images each (code from tutorial: https://www.youtube.com/watch?v=fyZ9Rxpoz2I and https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py):

```
# Code from tutorial: https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py) and (https://www.youtube.com/watch?v=fyZ9Rxpoz2I)

from sklearn.utils import resample

# Balance the skin lesion diagnosis types

df_0 = skin_lesion_df[skin_lesion_df['label'] == 0]
df_1 = skin_lesion_df[skin_lesion_df['label'] == 1]
df_2 = skin_lesion_df[skin_lesion_df['label'] == 2]
df_3 = skin_lesion_df[skin_lesion_df['label'] == 3]
df_4 = skin_lesion_df[skin_lesion_df['label'] == 4]
df_5 = skin_lesion_df[skin_lesion_df['label'] == 5]
df_6 = skin_lesion_df[skin_lesion_df['label'] == 6]

# Set number of samples for each skin lesion diagnosis type

n_samples = 700

df_0_balanced = resample(df_0, replace=True, n_samples=n_samples, random_state=42)
df_1_balanced = resample(df_1, replace=True, n_samples=n_samples, random_state=42)
df_2_balanced = resample(df_2, replace=True, n_samples=n_samples, random_state=42)
df_3_balanced = resample(df_3, replace=True, n_samples=n_samples, random_state=42)
df_4_balanced = resample(df_4, replace=True, n_samples=n_samples, random_state=42)
df_5_balanced = resample(df_5, replace=True, n_samples=n_samples, random_state=42)
df_6_balanced = resample(df_6, replace=True, n_samples=n_samples, random_state=42)

# Combine the skin lesion diagnosis type dataframes and observe the distribution of each skin lesion diangosis type

skin_lesion_df_balanced = pd.concat([df_0_balanced, df_1_balanced,
                           df_2_balanced, df_3_balanced,
                           df_4_balanced, df_5_balanced, df_6_balanced])

# Display the distribution of skin lesion diagnosis counts

print(skin_lesion_df_balanced['label'].value_counts())
```

```
0    700
1    700
2    700
3    700
4    700
5    700
6    700
Name: label, dtype: int64
```

- We now need to preprocess the skin lesion images to prepare the data for model training (code from tutorial code: https://www.youtube.com/watch?v=fyZ9Rxpoz2I and

https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py):

- We create a variable called 'data_dir' to store the Google Drive path where the skin lesion images are stored:

```
data_dir = '/content/drive/MyDrive/Datasets/SkinCancerMNIST/images'
```

- We then create a dictionary called 'all_image_path' and use the 'glob' function to find all skin lesion image files with the file extension type jpg (representing jpeg images) in each subdirectory of the main directory of where the skin lesion images are stored on Google Drive.

```
all_image_path = {os.path.splitext(os.path.basename(x))[0]: x
                  for x in glob(os.path.join(data_dir, '*', '*.jpg'))}
```

- We then add a column called 'path' to the balanced Pandas dataframe (skin_lesion_df_balanced) containing 700 skin lesion images of each skin diagnosis type. The path column contains the file path directory of each skin lesion image in the skin_lesion_df_balanced dataframe. We create a lambda function to map each image path with the corresponding image array. We use 'Image.open' from the PIL library to open each image from the image paths, resize the image to 224x224 pixels using the resize() method from PIL library, and convert each image to a numpy array using np.asarray():

Read and preprocess the images for model training

```
# Code from tutorial: https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py) and (https://www.youtube.com/watch?v=fyZ9Rxpoz2I)

data_dir = '/content/drive/MyDrive/Datasets/SkinCancerMNIST/images'
all_image_path = {os.path.splitext(os.path.basename(x))[0]: x
                  for x in glob(os.path.join(data_dir, '*', '*.jpg'))}

# Add path as a new column in the dataframe
skin_lesion_df_balanced['path'] = skin_lesion_df['image_id'].map(all_image_path.get)
# Read the images from the path
skin_lesion_df_balanced['image'] = skin_lesion_df_balanced['path'].map(lambda x: np.asarray(Image.open(x).resize((SIZE,SIZE))))
```

- We can observe the first five records of the modified Pandas dataframe (skin_lesion_df_balanced) to see if the path and image columns are added and write the skin_lesion_df_balanced dataframe to a csv using the Pandas function to_csv():

```
[ ] # Show first 5 records of skin_lesion_df_balanced to show new image column
    skin_lesion_df_balanced.head()
```

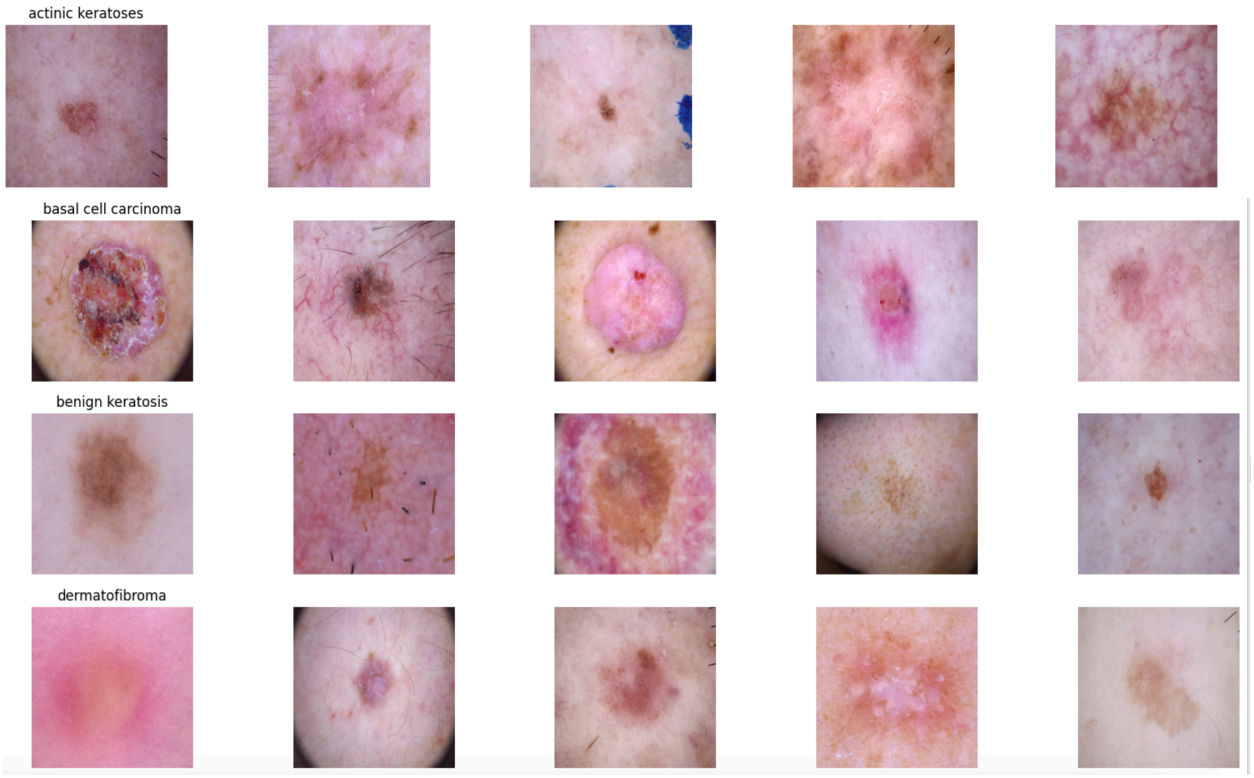
| | lesion_id | image_id | dx | dx_type | age | sex | localization | label | path | image |
|---|---|---|---|---|---|---|---|---|---|---|
| **9789** | HAM_0003136 | ISIC_0026645 | akiec | histo | 65.0 | male | back | 0 | /content/drive/MyDrive/Datasets/SkinCancerMNIS... | [[[149, 106, 88], [148, 108, 94], [153, 117, 1... |
| **9957** | HAM_0006587 | ISIC_0025780 | akiec | histo | 70.0 | male | face | 0 | /content/drive/MyDrive/Datasets/SkinCancerMNIS... | [[[135, 75, 83], [137, 77, 84], [141, 81, 86],... |
| **9793** | HAM_0005505 | ISIC_0024450 | akiec | histo | 50.0 | male | upper extremity | 0 | /content/drive/MyDrive/Datasets/SkinCancerMNIS... | [[[69, 36, 38], [79, 46, 50], [91, 59, 61], [1... |
| **9758** | HAM_0003455 | ISIC_0027896 | akiec | histo | 75.0 | male | hand | 0 | /content/drive/MyDrive/Datasets/SkinCancerMNIS... | [[[166, 115, 152], [181, 128, 163], [184, 128,... |
| **9875** | HAM_0005459 | ISIC_0029268 | akiec | histo | 85.0 | male | upper extremity | 0 | /content/drive/MyDrive/Datasets/SkinCancerMNIS... | [[[195, 171, 195], [194, 171, 193], [198, 176,... |

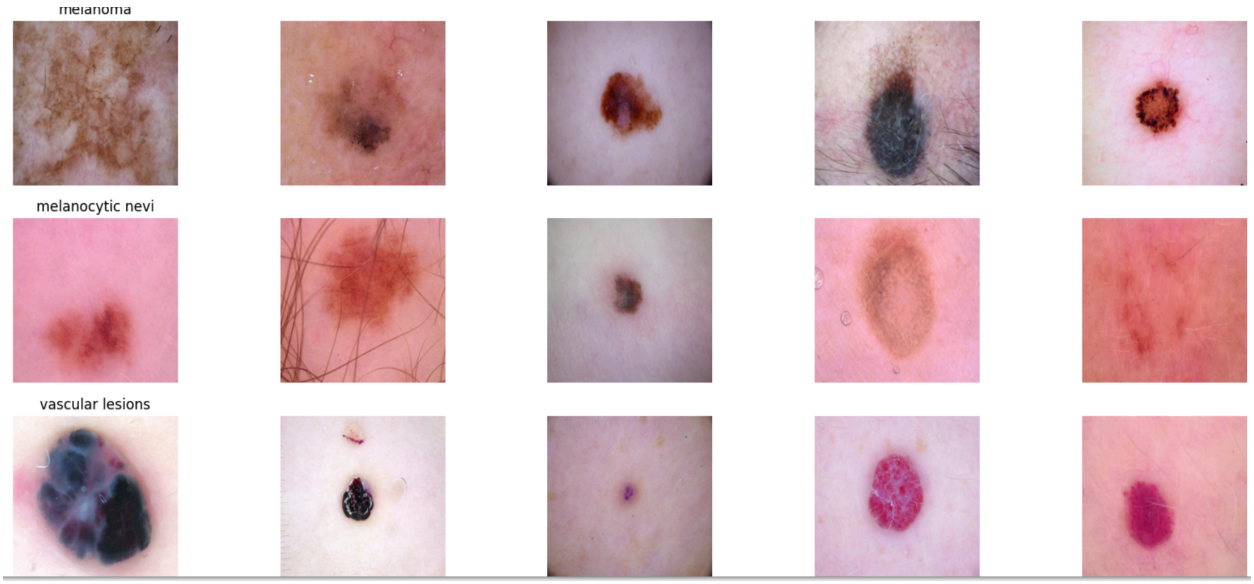
```
[ ] # Write balanced class dataframe to CSV
    skin_lesion_df_balanced.to_csv('/content/drive/MyDrive/Datasets/SkinCancerMNIST/skin_lesion_df_balanced_version_2_.csv', index=False)
```

- From our modified Pandas dataframe (skin_lesion_df_balanced), we can show a sample of 5 resized images (224x224 pixels) from each skin diagnosis type. We can use a dictionary with the diagnosis type abbreviation as the key and the long form of the skin

lesion diagnosis as the value. The diagnosis type abbreviation is represented by the column name ‘dx’ in the skin_lesion_df_balanced Pandas dataframe, and we use this abbreviation to retrieve the long form of the skin lesion diagnosis. We can show 5 resized images by iterating over the 7 diagnosis types and displaying 5 random images from each diagnosis type using matplotlib (code from tutorials: https://www.youtube.com/watch?v=fyZ9Rxpoz2I and https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py) and https://www.kaggle.com/code/smitisinghal/skin-disease-classification):

```
# Code from tutorial: https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py) and (https://www.youtube.com/watch?v=fyZ9Rxpoz2I)
# and from tutorial: https://www.kaggle.com/code/smitisinghal/skin-disease-classification

# Set number of samples for each skin lesion diagnosis type
n_samples = 5


# Create a dictionary of the diagnosis codes and skin lesion types to show the skin lesion type on the figures
diagnosis_dict = {
    'bkl': 'benign keratosis',
    'mel': 'melanoma',
    'nv': 'melanocytic nevi',
    'vasc': 'vascular lesions',
    'akiec': 'actinic keratoses',
    'bcc': 'basal cell carcinoma',
    'df': 'dermatofibroma'
}

# Set the number of samples to show and image dimensions
fig, m_axs = plt.subplots(7, n_samples, figsize=(4*n_samples, 3*7))


for n_axs, (type_name, type_rows) in zip(m_axs, skin_lesion_df_balanced.sort_values(['dx']).groupby('dx')):
    n_axs[0].set_title(diagnosis_dict[type_name])  # Use the long form of the skin diagnosis type from the dictionary
    for c_ax, (_, c_row) in zip(n_axs, type_rows.sample(n_samples, random_state=1234).iterrows()):
        c_ax.imshow(c_row['image'])
        c_ax.axis('off')
```






- Our next step, we will convert the pixel values stored in the 'image' column in the skin_lesion_df_balanced Pandas dataframe to numpy arrays and assign to variable X. We then scale the pixel values of the images stored in variable X by dividing by 255 which allows all the images to be rescaled proportionally to a value between 0 and 1. We then assign the skin lesion diagnosis types to the variable Y and convert the skin lesion types to categorical to one-hot encode the labels for training a multi-class classification model (7 classes). We then use train_test_split function to split the data to 20% used for testing and 80% used for training (code from tutorial: https://www.youtube.com/watch?v=fyZ9Rxpoz2I and https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py):

Convert dataframe of images into a numpy array and split the data to train and test 20% of the data

```
# Code from tutorial: https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py) and (https://www.youtube.com/watch?v=fyZ9Rxpoz2I)

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Convert image attribute to numpy array
X = np.asarray(skin_lesion_df_balanced['image'].tolist())
X = X/255.  # Scale values to 0-1

# Assign the labels to Y
Y = skin_lesion_df_balanced['label']

# Convert to categorical for multiclass classification
Y_cat = to_categorical(Y, num_classes=7)

# Split into training and testing datasets (test 20% of the data)
x_train, x_test, y_train, y_test = train_test_split(X, Y_cat, test_size=0.20, random_state=42)
```

- Before training the convolutional neural network model, we can create a callback to stop training the model if it the validation accuracy reaches 87%. This can help in capturing the highest validation accuracy for the model since training the model would reach percentages in the 80s when training (code from StackOverflow post: https://stackoverflow.com/questions/59563085/how-to-stop-training-when-it-hits-a-specific-validation-accuracy)

```
# Stop training if model reaches 87% validation accuracy (code from StackOverflow: https://stackoverflow.com/questions/59563085/how-to-stop-training-when-it-hits-a
import tensorflow as tf


class ValidationThreshold(tf.keras.callbacks.Callback):
    def __init__(self, threshold):
        super(ValidationThreshold, self).__init__()
        self.threshold = threshold

    def on_epoch_end(self, epoch, logs=None):
        val_acc = logs["val_acc"]
        if val_acc >= self.threshold:
            self.model.stop_training = True

validation_threshold = ValidationThreshold(threshold=0.87)
```

- Our next step will be to train the convolutional neural network model to make predictions of the skin lesion diagnosis type. We can use a pretrained model called 'MobileNetV2' to assist with model training. We add the following modules and layers to the model architecture (code from tutorial: https://www.youtube.com/watch?v=fyZ9Rxpoz2I and https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py):
    - EarlyStopping callback which can be used to prevent overfitting by stopping training if the model performance does not improve after a certain number of epochs (for training the model, the number of epochs is set with the parameter 'patience' at 20 epochs to monitor if the model performance does not improve after running 20 epochs – source: https://keras.io/api/callbacks/early_stopping/)
    - We load the MobileNetV2 model with pretrained weights trained on the ImageNet dataset, which is a large dataset used by researchers for image classification tasks (source: https://www.mathworks.com/help/deeplearning/ref/mobilenetv2.html)
    - We initialize a sequential model where subsequent layers will be added. We then add the MobileNetV2 model, GlobalAveragePooling2D layer and Batch Normalization layers to the model.
    - We then add a Dense layer (fully connected layer) with 64 neurons to assist the model in recognizing patterns in the data (source: https://keras.io/api/layers/core_layers/dense/ )
    - We add a Dropout layer with a dropout rate of 0.5. During each epoch of training, 50% of the input neurons will be set to 0. This can help prevent overfitting and improve the generalization ability of the model to predict skin lesion images it has not seen before (source: https://keras.io/api/layers/regularization_layers/dropout/ )
    - We compile the model and use the 'categorical_crossentropy' loss function, Adam optimizer, and assess the performance of the model with accuracy metrics.
    - We train the model on 50 epochs and save the model. We then plot the training and validation loss, training and validation accuracy, and the confusion matrix. We then plot a bar graph showing the fraction of incorrect predictions by each class to observe which skin lesion diagnosis types the model predicted less accurate than others.

```
# Code from tutorial: https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py) and (https://www.youtube.com/watch?v=fyZ9Rxpoz2I)

from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import GlobalAveragePooling2D, Dropout, Dense
from tensorflow.keras.layers import BatchNormalization

# Load the MobileNetV2 model with pre-trained weights
mobilenet_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(SIZE, SIZE, 3))

for layer in mobilenet_model.layers:
    layer.trainable = False

num_classes = 7

model = Sequential()
model.add(mobilenet_model)

model.add(GlobalAveragePooling2D())
model.add(BatchNormalization())

model.add(Dense(64))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['acc'])


# Define early stopping criteria
early_stopping = EarlyStopping(monitor='val_loss', patience=20, verbose=1)


batch_size = 64
epochs = 50
```

```
history = model.fit(
    x_train, y_train,
    epochs=epochs,
    batch_size = batch_size,
    callbacks=[early_stopping, validation_threshold],
    validation_data=(x_test, y_test),
    verbose=2)

score = model.evaluate(x_test, y_test)
print('Test accuracy:', score[1])
# Save the model
model.save('/content/drive/MyDrive/Datasets/SkinCancerMNIST/model_version_2.h5')



# Plot training and validation accuracy after running each epoch
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()


acc = history.history['acc']
val_acc = history.history['val_acc']
plt.plot(epochs, acc, 'y', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
# Predict on test data
y_pred = model.predict(x_test)
# Convert the predictions to one hot vectors
y_pred_classes = np.argmax(y_pred, axis = 1)
# Convert the test data to one hot vecotrs
y_true = np.argmax(y_test, axis = 1)

# Define the confusion matrix
cm = confusion_matrix(y_true, y_pred_classes)

# Create figure with subplots
fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(6, 12))

# Label the confusion matrix
sns.set(font_scale=1.6)
sns.heatmap(cm, annot=True, linewidths=.5, ax=ax1)
ax1.set_title('Confusion Matrix')

# Plot the fractional incorrect misclassifications bar chart
incorr_fraction = 1 - np.diag(cm) / np.sum(cm, axis=1)
ax2.bar(np.arange(7), incorr_fraction)
ax2.set_xlabel('True Skin Lesion Class')
ax2.set_ylabel('Fraction of Incorrect Predictions')
ax2.set_title('Fraction of Incorrect Predictions by Class')

# Adjust spacing between subplots
plt.tight_layout()

# Show the plots
plt.show()
```

---

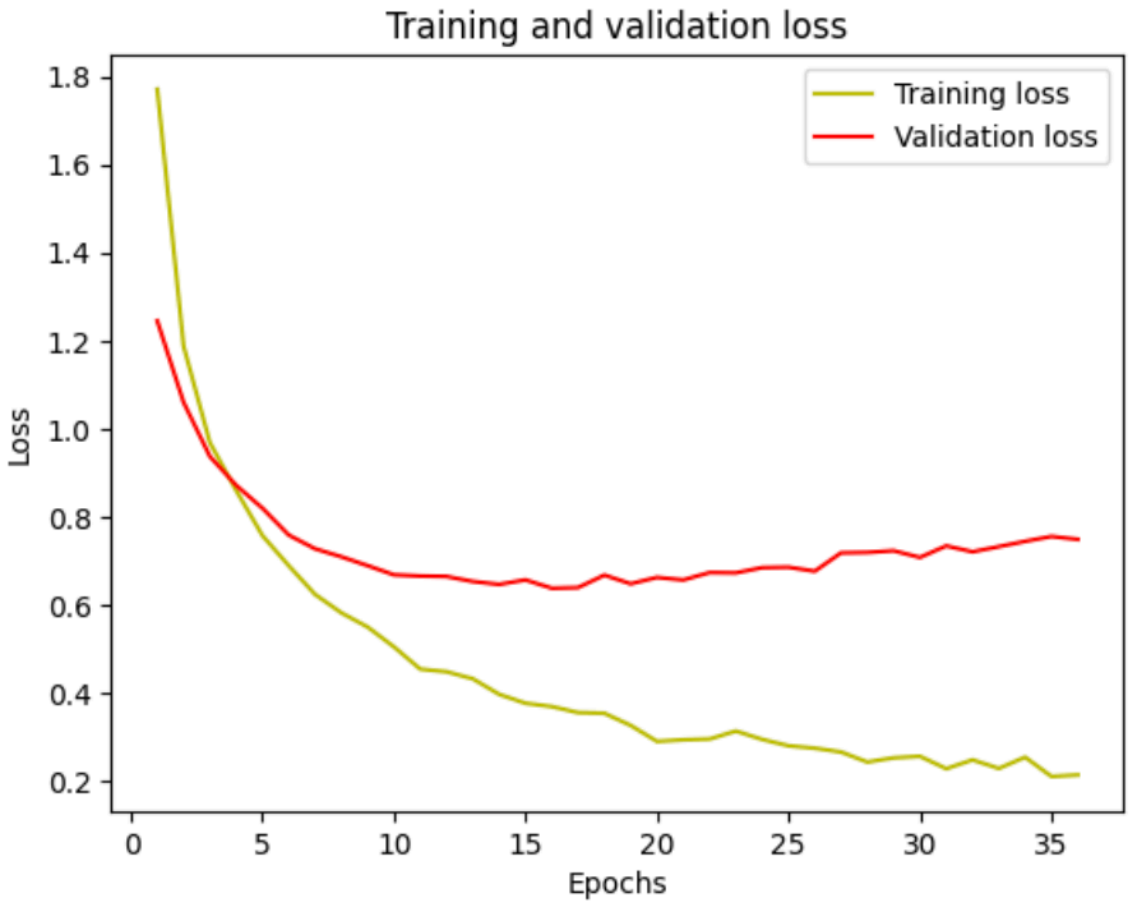
**Model Results**

- After training the model on 36 epochs, the model activated Early Stopping as the validation accuracy was not improving for 20 epochs and the validation accuracy was **82.3%**

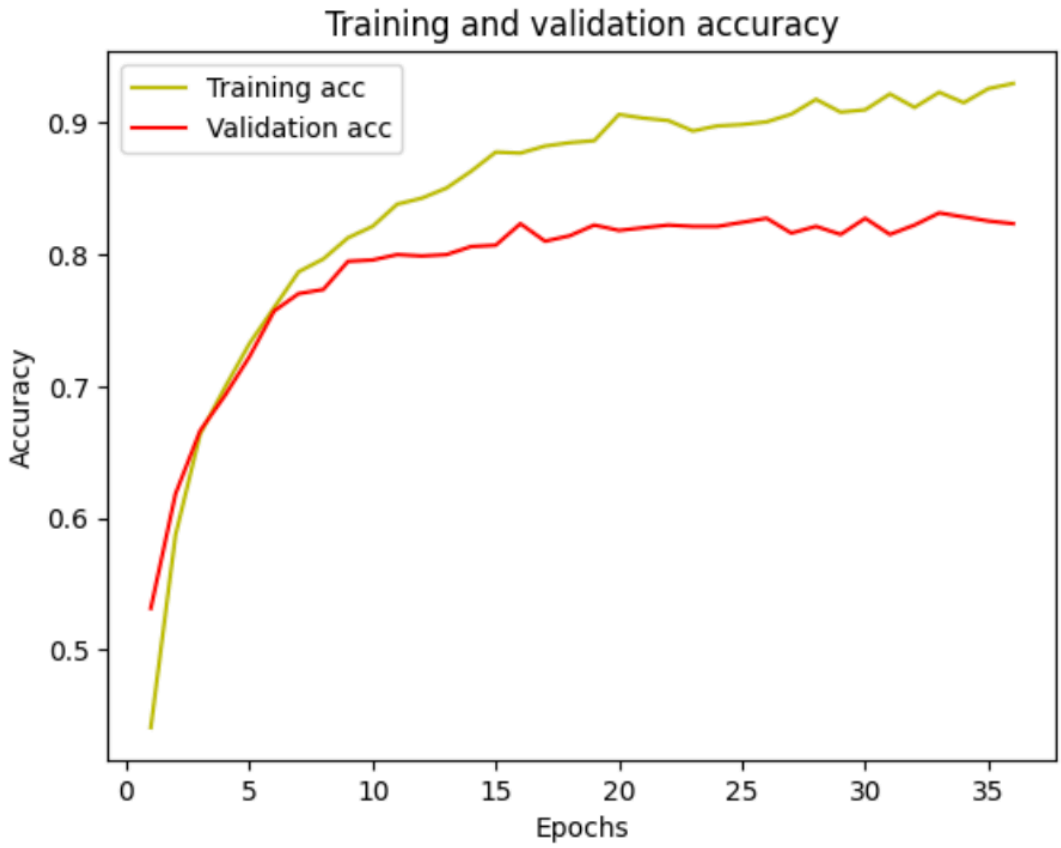
```
Epoch 36: early stopping
31/31 [==============================] - 14s 458ms/step - loss: 0.7493 - acc: 0.8235
Test accuracy: 0.8234694004058838
```

---

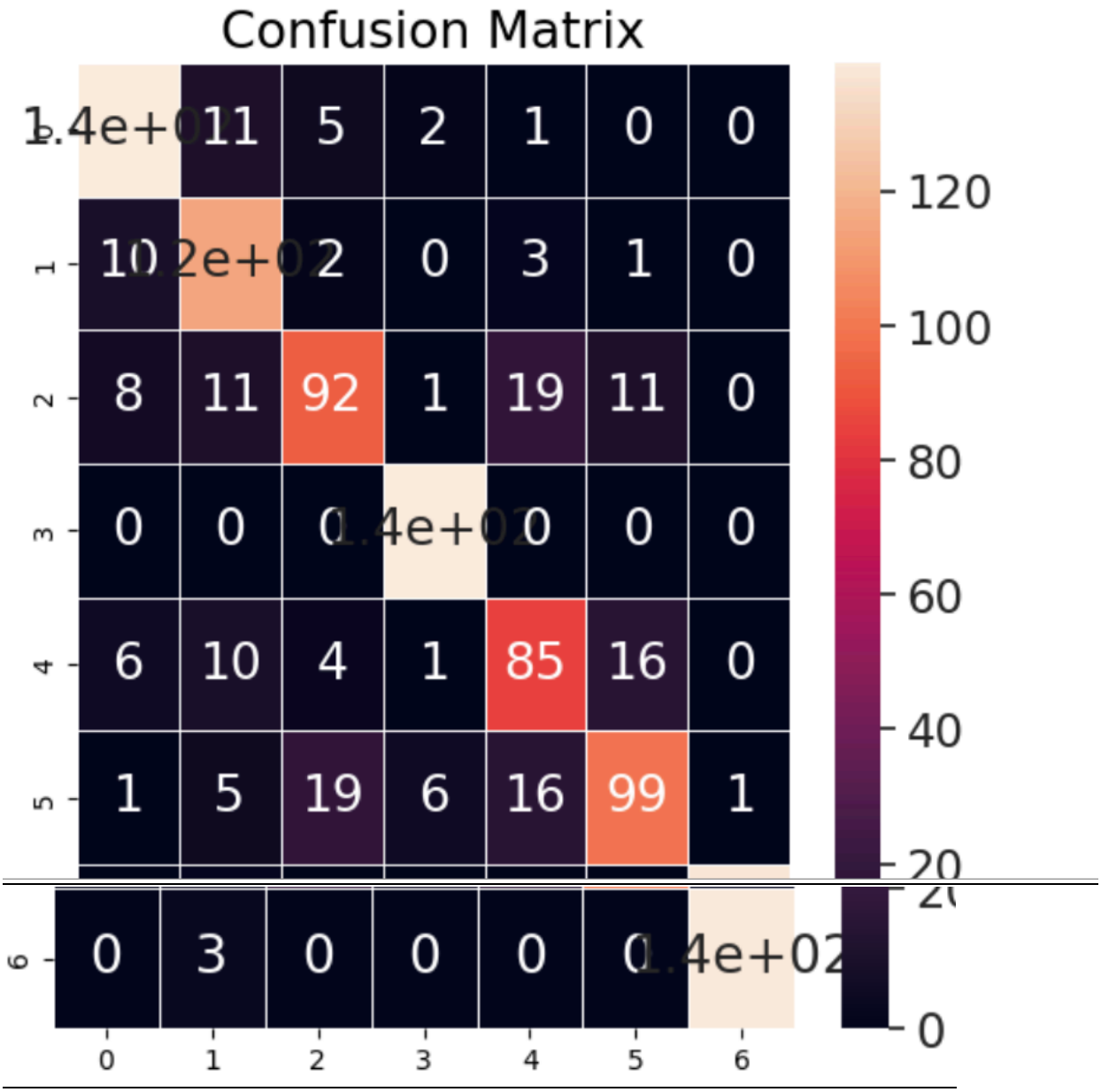
- Here is the plot showing the training and validation loss over the 36 epochs of training the model:



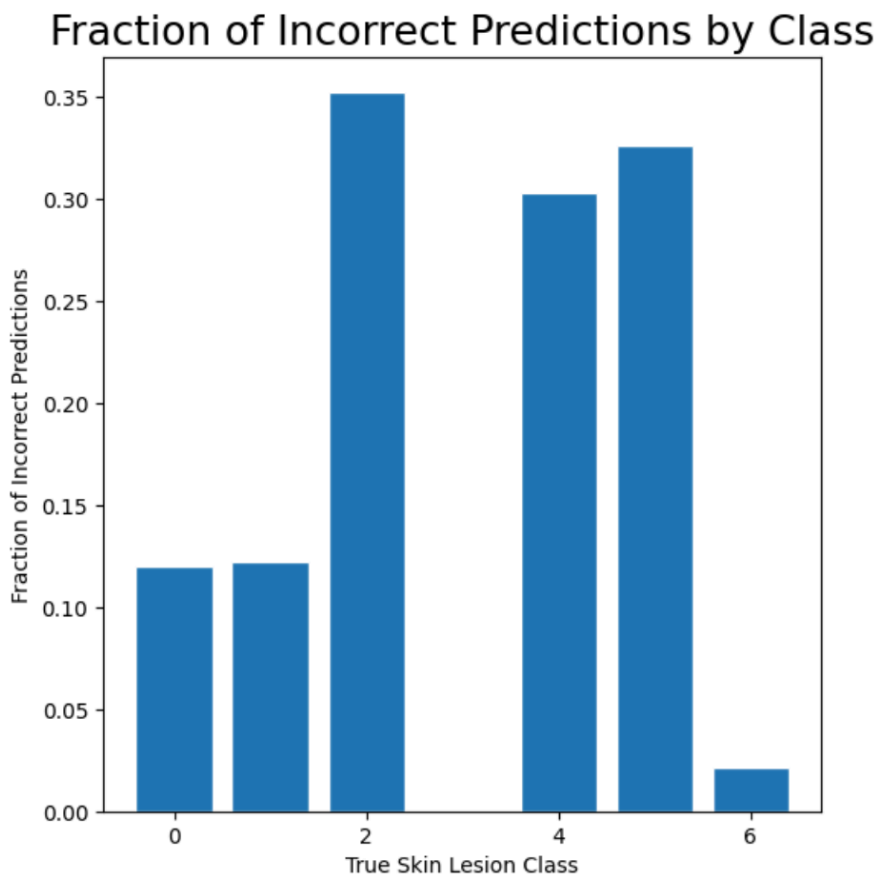

- Here is the plot showing the training and validation accuracy over 36 epochs. We can observe between epochs 0 and 7, the model reaches significant validation accuracy improvement and then begins to taper off around epoch 10 to around 80-83% validation accuracy from epochs 10-36:




- We can observe from the confusion matrix, the lower the diagonal values the better the model performed in predicting the correct skin lesion diagnosis types. We can observe from the confusion matrix, the model performed well in classifying class 0 (actinic keratoses), class 1 (basal cell carcinoma), class 3 (dermatofibroma), and class 6 (vascular lesions).




- Observing the bar chart with the fraction of incorrect predictions by class, we can see class 2 (benign keratosis) had the highest percentage of incorrect predictions at around 35%. Class 4 (melanoma) and class 5 (melanocytic nevi) also did not perform well in classifying the correct skin lesion diagnosis type with percentages of incorrect predictions around 30-33%.




---

**Implementation**

After training the model, we can use the Streamlit Python library to create an interface for the user to upload a skin lesion image and have the model predict the skin lesion diagnosis type with a percentage of how the confident the model is in its prediction. We can also use Streamlit to output information regarding the predicted skin lesion diagnosis type to the hypothetical patient.

- Code from YouTube tutorial: https://www.youtube.com/watch?v=_Syn5SpWgZ0&feature=youtu.be
- Code from Towards Datascience Tutorial: https://towardsdatascience.com/Streamlit-from-scratch-presenting-data-d5b0c77f9622
- Code from Streamlit documentation:
- https://docs.Streamlit.io/library/api-reference/widgets/st.text_input
- https://docs.Streamlit.io/library/api-reference/charts/st.altair_chart
- https://docs.Streamlit.io/library/api-reference/media/st.image

The Streamlit application can be launched using a LocalTunnel (code from tutorial: https://discuss.Streamlit.io/t/how-to-launch-Streamlit-app-from-google-colab-notebook/42399)

The Streamlit application code has the following components:

- Allows user to upload a jpeg file of a skin lesion image.
- Converts uploaded jpeg file to a PIL image for image processing.
- Resizes the uploaded skin lesion image to a 224 x 224 pixels to match the input shape of the model.
- Converts the resized skin lesion image to a NumPy array for the model prediction.
- Loads the model from the saved model file after training the model.
- Makes a prediction of the skin lesion image based on the NumPy array.
- Retrieves the skin lesion diagnosis class with the highest confidence score percentage (probability) using np.argmax and references the dictionary mapping to output the skin lesion diagnosis type.
- Retrieves the confidence score percentages of all the skin lesion types for the predicted skin lesion image. This can be useful if the model was not very accurate in its prediction and having the 2nd or 3rd highest confidence score can assist in making a diagnosis.
- Creates a Pandas dataframe with the skin lesion diagnosis names and confidence score percentages. This dataframe is used as the input for the bar chart that is outputted when predicting the skin lesion image. The bar chart uses the Altair visualization Python library and shows all the skin lesion types and the confidence score percentages associated with them after the user uploads a skin lesion image. The Pandas dataframe is also outputted as a table to show the confidence score percentage associated with each skin lesion type for the user uploaded skin lesion image.
- A text input box to have the user input the predicted skin lesion type.
- Once the user inputs the skin lesion type, the Streamlit application will output information and treatment regarding the predicted skin lesion type (information on skin lesion types and treatment options referenced from the following sites:
    - https://www.mayoclinic.org/diseases-conditions
    - https://www.skincancer.org/
    - https://dermnetnz.org/
    - https://www.mayoralderm.com/vascular-lesions/

**Implementation Output**

- Launch the Streamlit application by running the skin_lesion_classifier.py file using this code and copy the outputted url:

```
# from Streamlit forums: https://discuss.streamlit.io/t/how-to-launch-streamlit-app-from-google-colab-notebook/42399

!streamlit run skin_lesion_classifier.py &>/content/logs.txt & npx localtunnel --port 8501 & curl ipv4.icanhazip.com
```

```
34.32.231.170
npx: installed 22 in 1.493s
your url is: https://polite-bees-push.loca.lt
```

---

- Enter the public IP from the output (ex: 34.32.231.170) into the ‘Endpoint IP’ and click on ‘Click to Submit’. The Streamlit application will now run in the browser

## polite-bees-push.loca.lt

### Friendly Reminder

This website is served via a localtunnel. This is just a reminder to always check the website address you're giving personal, financial, or login details to is actually the real/official website.

Phishing pages often look similar to pages of known banks, social networks, email portals or other trusted institutions in order to acquire personal information such as usernames, passwords or credit card details.

Please proceed with caution.

### To access the website, please confirm the tunnel creator's public IP below.

If you don't know what it is, please ask whoever you got this link from.

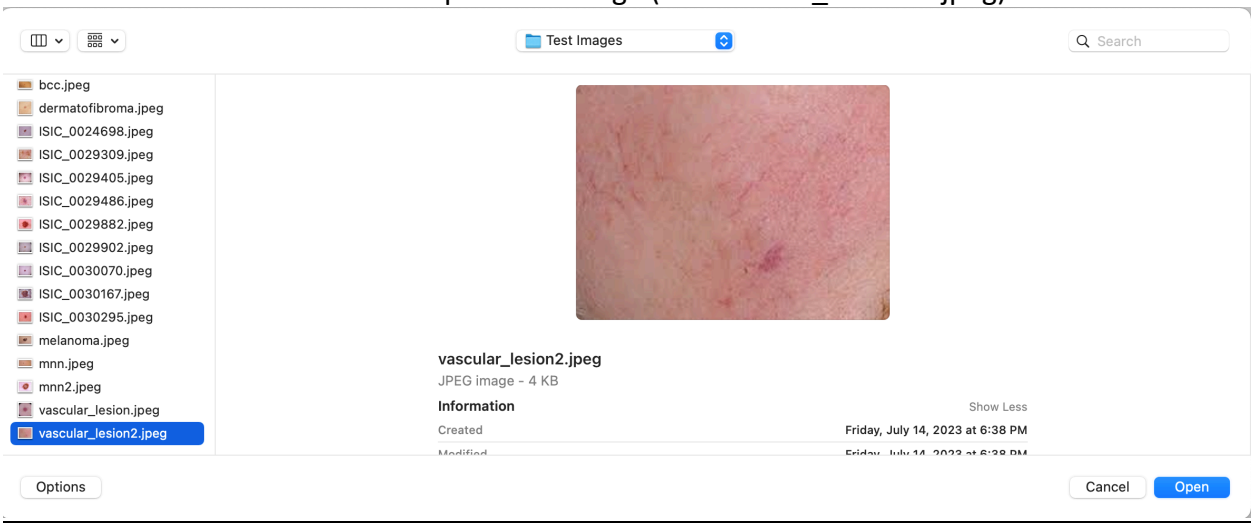
This password-like gate is now sadly required since too many phishing portals are being hosted via localtunnel and I'm getting bombarded with abuse notices.

Endpoint IP: 34.32.231.170

Click to Submit

---

- Click on Browse files and upload an image (ex: vascular_lesions2.jpeg)




# Skin Lesion Image Classifier

Predict the classification of the skin lesion diagnosis type based on user uploaded image

Please upload skin lesion image

Drag and drop file here

Limit 200MB per file • JPG, JPEG

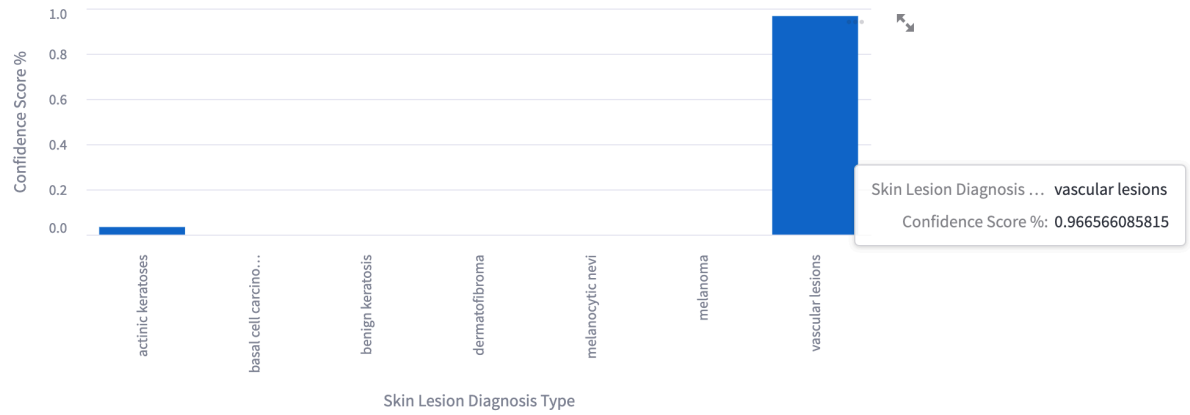
Browse files

vascular_lesion2.jpeg 3.6KB

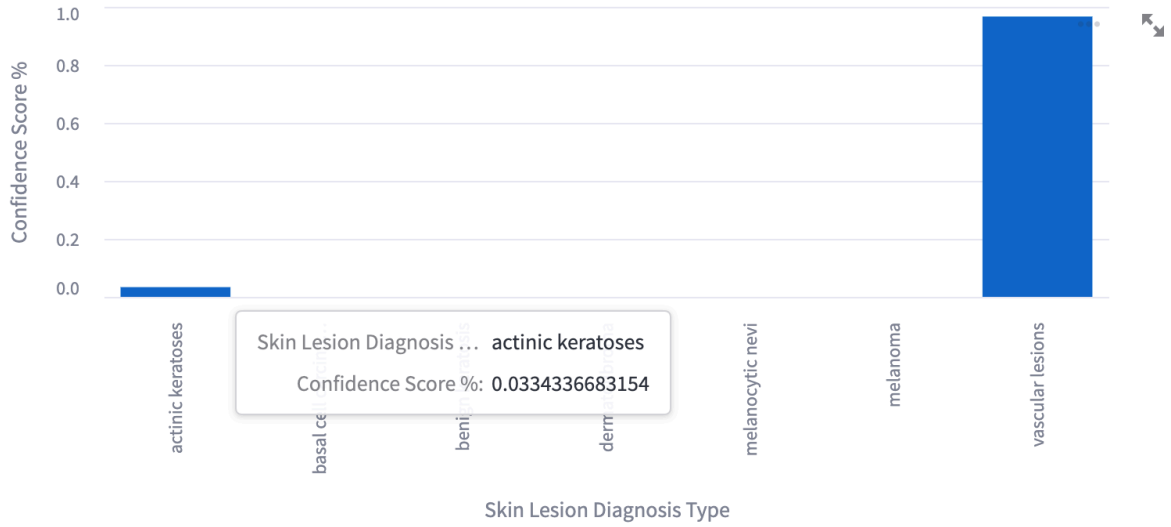
---

Resized Skin Lesion Image

The outputted bar graph shows the model predicted the uploaded skin lesion diagnosis type as vascular lesions with 96.66% confidence (this skin lesion diagnosis type had the highest probability):




The other skin lesion diagnosis type (actinic keratoses) had confidence score of 3.34%:




The output also shows a table representing the skin lesion diagnosis types and confidence score percentages associated with the user uploaded skin lesion image:

| | Skin Lesion Diagnosis Type | Confidence Score % |
|---|---|---|
| 0 | actinic keratoses | 0.0334 |
| 1 | basal cell carcinoma | 0 |
| 2 | benign keratosis | 0 |
| 3 | dermatofibroma | 0 |
| 4 | melanoma | 0 |
| 5 | melanocytic nevi | 0 |
| 6 | vascular lesions | 0.9666 |

Enter the name of the predicted skin lesion diagnosis type (ex: vascular lesions) into the text input box. The output will show more information regarding vascular lesions.

Please enter the name of the skin lesion diagnosis type for more information

vascular lesions

Here is more information on: vascular lesions

Vascular lesions are due to ruptures underneath the blood vessels and can be reddish or purplish. These types of lesions can be congenital (birthmarks) or acquired later in life. Treatment options include Intense Pulsed Light Therapy which applies a series of light beams over the lesions that penetrate the skin layers and allow absorption which will reduce or eliminate the lesion. The dermatologist will discuss all treatment options with you regarding this type of lesion.

---

## Project Dataset Information

- The dataset used to train the CNN model is a subset of 4,900 images from the HAM10000 dataset ('Human Against Machine with 10,000 training images). The link for the dataset is from Kaggle: https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000
- The HAM10000 dataset contains 10,015 images of skin lesions. The dataset also contains a metadata csv file called 'HAM10000_metadata.csv'. The metadata csv contains the following attributes:
    - lesion_id: identifier used for the skin lesion (source: https://medium.com/@nahmed3536/exploring-the-ham10000-dataset-355a9c79116b)
    - image_id: 7-digit identifier in form of ISIC_X to label each image in the HAM10000 dataset (source: https://medium.com/@nahmed3536/exploring-the-ham10000-dataset-355a9c79116b).
    - dx: refers to the diagnosis. The diagnosis types are represented by 7 different diagnosis codes (source: https://medium.com/@nahmed3536/exploring-the-ham10000-dataset-355a9c79116b)
        - akiec: actinic keratosis
        - bcc: basal cell carcinoma
        - bkl: benign keratosis
        - df: dermatofibroma
        - nv: melanocytic nevi
        - mel: melanoma
        - vasc: vascular skin lesions
    - dx_type: field representing how the diagnosis of the skin lesion was verified. The different methods are histopathology (verified by a dermopathologist), confocal (verified through microscopes), follow-up visits to track any changes with the lesion, and consensus by panel of experts (source:

https://medium.com/@nahmed3536/exploring-the-ham10000-dataset-355a9c79116b)
    - age: age of the patient with the skin lesion (source: https://medium.com/@nahmed3536/exploring-the-ham10000-dataset-355a9c79116b)
    - sex: biological sex of the patient (male, female or unknown) (source: https://medium.com/@nahmed3536/exploring-the-ham10000-dataset-355a9c79116b)
    - localization: where the skin lesion was found on the patient's body (face, back, etc) (source: https://medium.com/@nahmed3536/exploring-the-ham10000-dataset-355a9c79116b)

**Conclusion/Future Work**

Utilizing the pretrained model MobileNetV2 resulted in validation accuracy of around 82.3%. This validation accuracy is ok for training purposes but is not suitable for medical use or production ready systems. There is a large class imbalance in the HAM10000 dataset with nearly 67% of the data representing skin lesion diagnosis type of melanocytic nevi. To improve on similar types of skin lesion classification models, additional images of skin lesions should be gathered to further balance the classes and feed the training of the model. The Streamlit web application is one option for web-based applications to allow users to upload a skin lesion image and retrieve information regarding the predicted skin lesion diagnosis type. Other options to consider for web-based applications include Flask and Dash by Plotly.

**Reference Material**


- Using Google Drive in Google Colab Tutorial:
    - https://colab.research.google.com/notebooks/io.ipynb
- Kaggle API Key Code Tutorial:
    - https://www.kaggle.com/code/kalaikumarr/google-colab-kaggle-api
- Kaggle Dataset for Skin Cancer MNIST: HAM10000
    - https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000
- Code Tutorial for Unzipping Files in Python:
    - https://www.geeksforgeeks.org/unzipping-files-in-python/
- Code Tutorial for running TPU in Google Colab:
    - https://colab.research.google.com/notebooks/tpu.ipynb#scrollTo=FpvUOuC3j27n
- Kaggle Code Tutorial for Creating Skin Diagnosis Mapping and Assigning Mapping to Pandas Dataframe
    - https://www.kaggle.com/code/xinruizhuang/skin-lesion-classification-acc-90-pytorch
- Code Tutorial for Plotting Horizontal Bar Charts in Seaborn
    - https://www.statology.org/seaborn-horizontal-barplot/

- Code Tutorial for Preprocessing and Building A Skin Lesion Classifier Convolutional Neural Network
    - https://github.com/bnsreenu/python_for_microscopists/blob/master/203b_skin_cancer_lesion_classification_V4.0.py
    - https://www.youtube.com/watch?v=fyZ9Rxpoz2I
- Kaggle Code Tutorial for Showing Skin Lesion Image Samples
    - https://www.kaggle.com/code/smitisinghal/skin-disease-classification
- Stack Overflow Post for Stopping Training if Model Reaches Validation Accuracy Threshold
    - https://stackoverflow.com/questions/59563085/how-to-stop-training-when-it-hits-a-specific-validation-accuracy
- Scientific Data Paper Describing Skin Lesion Classification Algorithm
    - https://arxiv.org/pdf/1803.10417.pdf
- Machine Learning Apps with Streamlit Tutorial
    - https://www.youtube.com/watch?v=_Syn5SpWgZ0
- Streamlit Documentation
    - https://docs.streamlit.io/library/api-reference/widgets/st.text_input
    - https://docs.streamlit.io/library/api-reference/charts/st.altair_chart
    - https://docs.streamlit.io/library/api-reference/media/st.image
- Towards Data Science Tutorial on Presenting Data in Streamlit
    - https://towardsdatascience.com/streamlit-from-scratch-presenting-data-d5b0c77f9622
- Reference Material on Skin Lesions for Streamlit Output on Skin Lesion Information:
    - https://www.mayoclinic.org/diseases-conditions
    - https://www.skincancer.org/
    - https://dermnetnz.org/
    - https://www.mayoralderm.com/vascular-lesions/
- Information Regarding Features in HAM10000 Dataset
    - https://medium.com/@nahmed3536/exploring-the-ham10000-dataset-355a9c79116b
- Keras Documentation for Model Layer Explanations
    - https://keras.io/api/callbacks/early_stopping/
    - https://keras.io/api/layers/core_layers/dense/
    - https://keras.io/api/layers/regularization_layers/dropout/
- MobileNetV2 Model Reference
    - https://www.mathworks.com/help/deeplearning/ref/mobilenetv2.html