Installing libraries and Dependencies

Uncomment and run the following code block to install all required dependencies

In [ ]:
```
# !pip install keras
# !pip install numpy
# !pip install gradio
# !pip install tensorflow
# !pip install matplotlib
# !pip install scikit-learn
# !pip install opencv-python
```

Importing libraries

In [ ]:
```
import cv2
import os
import numpy as np

import gradio as gr

import matplotlib.pyplot as plt

import tensorflow as tf
from keras.optimizers import Adam
from keras.models import Sequential

from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout

from sklearn.metrics import classification_report
```

```
C:\ProgramData\Anaconda3\lib\site-packages\numpy\_distributor_init.py:30: UserWarning: loaded more than 1 DLL from .libs:
C:\ProgramData\Anaconda3\lib\site-packages\numpy\.libs\libopenblas.PYQHXLVVQ7VESDPUVUADXEVJOBGHJPAY.gfortran-win_amd64.dll
C:\ProgramData\Anaconda3\lib\site-packages\numpy\.libs\libopenblas.WCDJNK7YVMPZQ2ME2ZZHJJRJ3JIKNDB7.gfortran-win_amd64.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
```

Defining the `get_data` function to get data from the directory

In [ ]:
```
labels = ['red', 'black', 'geographic', 'normal', 'yellow'] # titles of subfolders
img_size = 120 # input image size

def get_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img))[...,::-1] #convert BGR to RGB format
                resized_arr = cv2.resize(img_arr, (img_size, img_size)) # Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data, dtype=object)
```

Defining training and validation directories

In [ ]:
```
train = get_data('/Dr. Tongue/data/train')
val = get_data('/Dr. Tongue/data/test/')
```

Generating the features and labels from the data and normalizing it

In [ ]:
```
x_train = []
y_train = []
x_val = []
y_val = []

for feature, label in train:
  x_train.append(feature)
  y_train.append(label)

for feature, label in val:
  x_val.append(feature)
  y_val.append(label)

# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255

x_train.reshape(-1, img_size, img_size, 1)
y_train = np.array(y_train)

x_val.reshape(-1, img_size, img_size, 1)
y_val = np.array(y_val)
```

Data Augmentation

In [ ]:
```
datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range = 30,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.2, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
        height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
        horizontal_flip = True,  # randomly flip images
        vertical_flip=False)  # randomly flip images

datagen.fit(x_train)
```

Model Definition

In [ ]:
```
model = Sequential()
model.add(Conv2D(32,3,padding="same", activation="relu", input_shape=(120,120,3)))
model.add(MaxPool2D())

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(Dropout(0.4))

model.add(Flatten())
model.add(Dense(128,activation="relu"))
model.add(Dense(5, activation="softmax"))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 120, 120, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 60, 60, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 60, 60, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 30, 30, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 15, 15, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 14400)             0
_________________________________________________________________
dense (Dense)                (None, 128)               1843328
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
Total params: 1,872,613
Trainable params: 1,872,613
Non-trainable params: 0
_________________________________________________________________
```

Model Compilation

In [ ]:
```
opt = Adam(learning_rate=0.0005)
model.compile(optimizer = opt , loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True) , metrics = ['accuracy'])
```

Model fitting

In [ ]:
```
history = model.fit(x_train,y_train, epochs = 25, validation_data = (x_val, y_val))
```

```
Epoch 1/25
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\keras\backend.py:4929: UserWarning: "`sparse_categorical_crossentropy
` received `from_logits=True`, but the `output` argument was produced by a sigmoid or softmax activation and thus does not represe
nt logits. Was this intended?"
  warnings.warn(
4/4 [==============================] - 37s 1s/step - loss: 1.6725 - accuracy: 0.1871 - val_loss: 1.5874 - val_accuracy: 0.2500
Epoch 2/25
4/4 [==============================] - 3s 779ms/step - loss: 1.5801 - accuracy: 0.3243 - val_loss: 1.5712 - val_accuracy: 0.4545
Epoch 3/25
4/4 [==============================] - 3s 741ms/step - loss: 1.5641 - accuracy: 0.4384 - val_loss: 1.5112 - val_accuracy: 0.5227
Epoch 4/25
4/4 [==============================] - 3s 750ms/step - loss: 1.4882 - accuracy: 0.5846 - val_loss: 1.4405 - val_accuracy: 0.5909
Epoch 5/25
4/4 [==============================] - 3s 743ms/step - loss: 1.3829 - accuracy: 0.7074 - val_loss: 1.2838 - val_accuracy: 0.6591
Epoch 6/25
4/4 [==============================] - 3s 735ms/step - loss: 1.2158 - accuracy: 0.6744 - val_loss: 1.1375 - val_accuracy: 0.5909
Epoch 7/25
4/4 [==============================] - 3s 749ms/step - loss: 1.0038 - accuracy: 0.7296 - val_loss: 0.9511 - val_accuracy: 0.7500
```

```
Epoch 8/25
4/4 [==============================] - 3s 741ms/step - loss: 0.8086 - accuracy: 0.7574 - val_loss: 0.8892 - val_accuracy: 0.6364
Epoch 9/25
4/4 [==============================] - 3s 790ms/step - loss: 0.6509 - accuracy: 0.8023 - val_loss: 0.9254 - val_accuracy: 0.5909
Epoch 10/25
4/4 [==============================] - 3s 896ms/step - loss: 0.5857 - accuracy: 0.8005 - val_loss: 0.8257 - val_accuracy: 0.7045
Epoch 11/25
4/4 [==============================] - 3s 676ms/step - loss: 0.5047 - accuracy: 0.8148 - val_loss: 1.1355 - val_accuracy: 0.5455
Epoch 12/25
4/4 [==============================] - 3s 682ms/step - loss: 0.4461 - accuracy: 0.8398 - val_loss: 1.0214 - val_accuracy: 0.6818
Epoch 13/25
4/4 [==============================] - 3s 693ms/step - loss: 0.5980 - accuracy: 0.8367 - val_loss: 0.9263 - val_accuracy: 0.5682
Epoch 14/25
4/4 [==============================] - 3s 690ms/step - loss: 0.3708 - accuracy: 0.8848 - val_loss: 0.7373 - val_accuracy: 0.7500
Epoch 15/25
4/4 [==============================] - 3s 690ms/step - loss: 0.3317 - accuracy: 0.8662 - val_loss: 0.7091 - val_accuracy: 0.8182
Epoch 16/25
4/4 [==============================] - 3s 805ms/step - loss: 0.3165 - accuracy: 0.9136 - val_loss: 0.7127 - val_accuracy: 0.6818
Epoch 17/25
4/4 [==============================] - 3s 732ms/step - loss: 0.2472 - accuracy: 0.9362 - val_loss: 0.7563 - val_accuracy: 0.7955
Epoch 18/25
4/4 [==============================] - 3s 696ms/step - loss: 0.2302 - accuracy: 0.9386 - val_loss: 0.7990 - val_accuracy: 0.6818
Epoch 19/25
4/4 [==============================] - 3s 709ms/step - loss: 0.1796 - accuracy: 0.9828 - val_loss: 0.5977 - val_accuracy: 0.7727
Epoch 20/25
4/4 [==============================] - 3s 707ms/step - loss: 0.1619 - accuracy: 0.9533 - val_loss: 0.7217 - val_accuracy: 0.7727
Epoch 21/25
4/4 [==============================] - 3s 677ms/step - loss: 0.1100 - accuracy: 0.9967 - val_loss: 0.7703 - val_accuracy: 0.7955
Epoch 22/25
4/4 [==============================] - 3s 703ms/step - loss: 0.0978 - accuracy: 0.9861 - val_loss: 0.6461 - val_accuracy: 0.7727
Epoch 23/25
4/4 [==============================] - 3s 677ms/step - loss: 0.0903 - accuracy: 0.9915 - val_loss: 0.7164 - val_accuracy: 0.7727
Epoch 24/25
4/4 [==============================] - 3s 685ms/step - loss: 0.0655 - accuracy: 1.0000 - val_loss: 0.7716 - val_accuracy: 0.7955
Epoch 25/25
4/4 [==============================] - 3s 672ms/step - loss: 0.0451 - accuracy: 1.0000 - val_loss: 0.6817 - val_accuracy: 0.7955
```

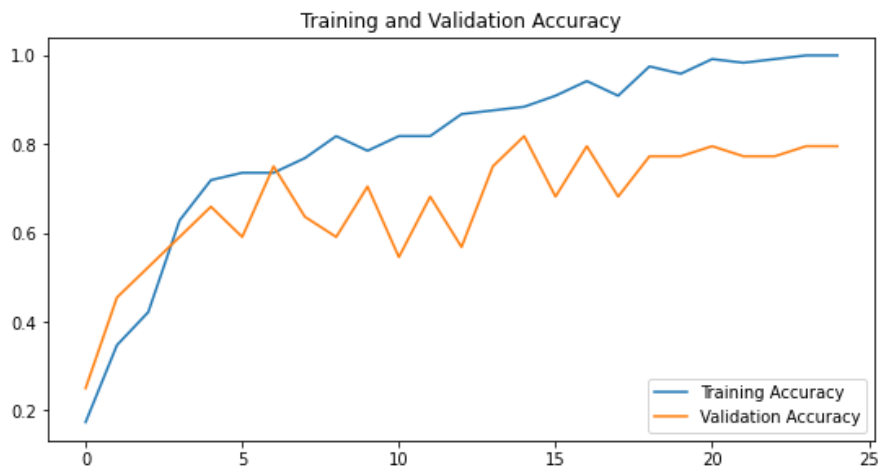
Plotting Model's accuracy and loss w.r.t. training and validation set

In [ ]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(25)

plt.figure(figsize=(20, 10))
plt.subplot(2, 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(2, 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()
```



Making predictions

In [ ]:
```
predictions = model.predict_classes(x_val)
predictions = predictions.reshape(1,-1)[0]
print(classification_report(y_val, predictions, target_names = ['red (Class 0)','black (Class 1)', 'geographic (Class 2)', 'normal
```

```
C:\ProgramData\Anaconda3\lib\site-packages\keras\engine\sequential.py:450: UserWarning: `model.predict_classes()` is deprecated an
d will be removed after 2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class
classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model
does binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
                    precision    recall  f1-score   support

       red (Class 0)       0.80      1.00      0.89         8
     black (Class 1)       0.91      0.91      0.91        11
geographic (Class 2)       1.00      0.44      0.62         9
    normal (Class 3)       0.55      0.75      0.63         8
    yellow (Class 4)       0.88      0.88      0.88         8
```

accuracy			0.80	44
macro avg	0.83	0.80	0.78	44
weighted avg	0.84	0.80	0.79	44

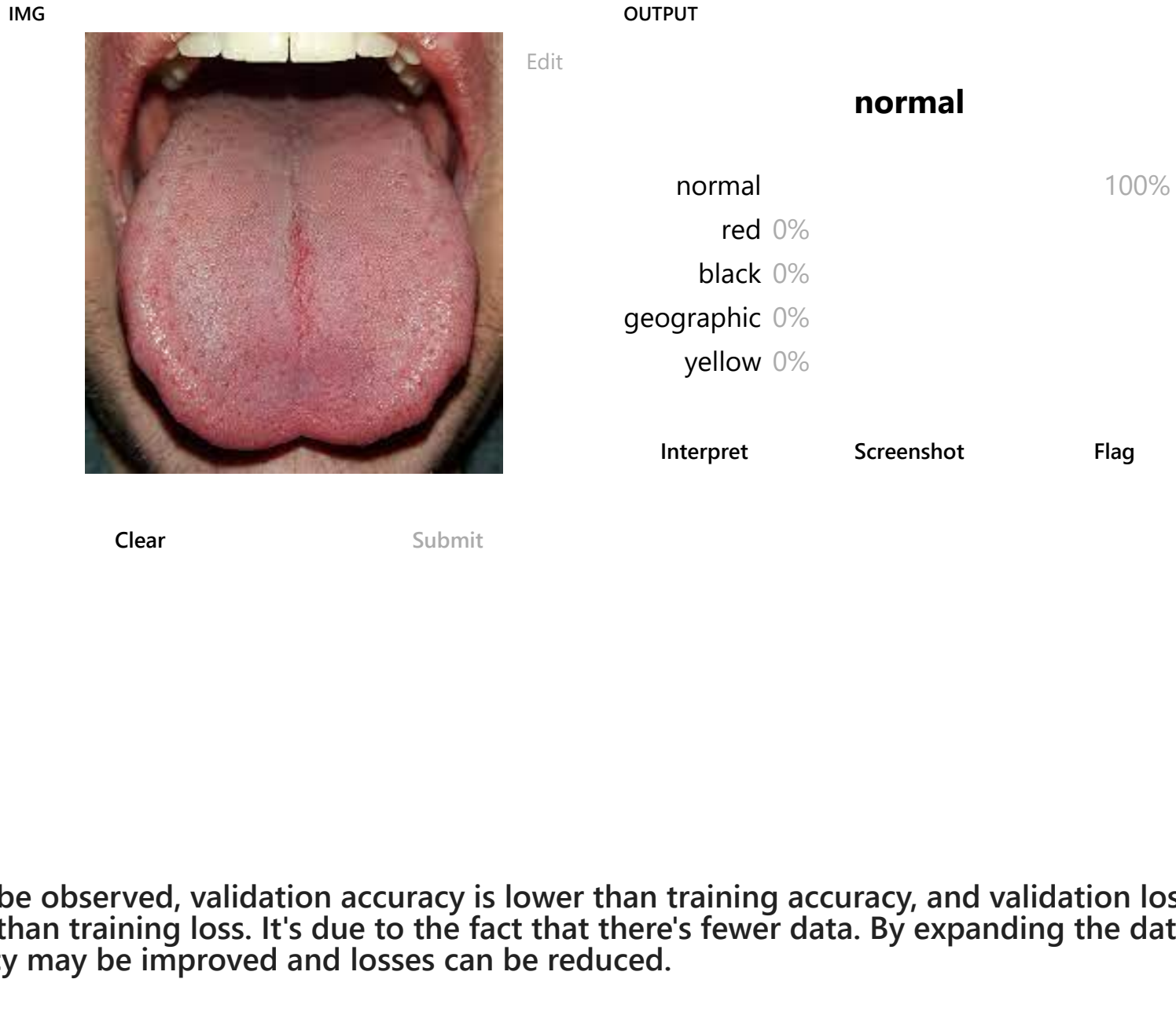
Predicting using local web interface at `http://127.0.0.1:7860/`

In [ ]:
```
def predict_image(img):
  img_4d=img.reshape(-1,120,120,3)
  prediction=model.predict(img_4d)[0]
  return {labels[i]: float(prediction[i]) for i in range(5)}

image = gr.inputs.Image(shape=(120,120))
label = gr.outputs.Label(num_top_classes=5)

gr.Interface(fn=predict_image, inputs=image, outputs=label,interpretation='default').launch(debug='True')
```

```
Running locally at: http://127.0.0.1:7860/
To create a public link, set `share=True` in `launch()`.
Interface loading below...
```



Note:

As can be observed, validation accuracy is lower than training accuracy, and validation loss is likewise higher than training loss. It's due to the fact that there's fewer data. By expanding the data set, accuracy may be improved and losses can be reduced.