DATA AUGMENTATION

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
!pip install augmentor
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting augmentor
  Downloading Augmentor-0.2.12-py2.py3-none-any.whl (38 kB)
Requirement already satisfied: Pillow>=5.2.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (8.4.0)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (1.22.4)
Requirement already satisfied: tqdm>=4.9.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (4.65.0)
Installing collected packages: augmentor
Successfully installed augmentor-0.2.12
```

```
!pip install augmentor
# Importing necessary library
import Augmentor
# Passing the path of the image directory
p = Augmentor.Pipeline('/content/gdrive/MyDrive/augmentation 8000/resized_Balanced dataset')

# Defining augmentation parameters and generating 5 samples
p.zoom(probability = 0.5, min_factor = 0.8, max_factor = 1.5)
p.flip_top_bottom(probability=0.5)
p.sample(5000)
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: augmentor in /usr/local/lib/python3.10/dist-packages (0.2.12)
Requirement already satisfied: tqdm>=4.9.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (4.65.0)
Requirement already satisfied: Pillow>=5.2.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (8.4.0)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (1.22.4)
Initialised with 3310 image(s) found.
Output directory set to /content/gdrive/MyDrive/augmentation 8000/resized_Balanced dataset/output.Processing <PIL.Image.Image image
```

```
import os

# specify the folder path
folder_path = "/content/gdrive/MyDrive/augmentation 8000/resized_Balanced dataset/output/iron"

# get a list of all the files in the folder
files = os.listdir(folder_path)

# initialize a counter variable to keep track of the number of images
num_images = 0

# loop through all the files in the folder
for file in files:
    # check if the file is an image file (you can modify this condition based on the types of images you have)
    if file.endswith(".jpg") or file.endswith(".jpeg") or file.endswith(".png") or file.endswith(".gif"):
        # increment the counter if it's an image file
        num_images += 1

# print the number of images found
print("Number of images: ", num_images)
```

```
Number of images:  851
```

```
pip install split-folders
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
```

```
import splitfolders

input_folder = r'/content/gdrive/MyDrive/augmentation 8000/resized_Balanced dataset/output'
splitfolders.ratio(input_folder, output= r'/content/gdrive/MyDrive/augmentation 8000/ausg_plit dataset_5k',
                   seed=42, ratio=(.8, .2),
                   group_prefix=None)
```

```
Copying files: 5000 files [00:48, 104.13 files/s]
```

```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.applications.xception import preprocess_input
from tensorflow.keras import layers
from tensorflow.python.keras.layers import Dense, Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam


train_path = '/content/gdrive/MyDrive/augmentation 8000/ausg_plit dataset_5k/train'
test_path = '/content/gdrive/MyDrive/augmentation 8000/ausg_plit dataset_5k/val'



from tensorflow.keras.layers import Input,Lambda,Dense,Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image


IMAGE_SIZE = [299,299]


# Use the Image Data Generator to import the images from the dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True
                                   )

test_datagen = ImageDataGenerator(rescale = 1./255,
                                  )


# Make sure you provide the same target size as initialied for the image size
train_set=train_datagen.flow_from_directory('/content/gdrive/MyDrive/augmentation 8000/ausg_plit dataset_5k/train',target_size=(299,299),
                                            batch_size=32,
                                            class_mode='categorical')
```

```
Found 3998 images belonging to 6 classes.
```

```
test_set = test_datagen.flow_from_directory('/content/gdrive/MyDrive/augmentation 8000/ausg_plit dataset_5k/val',
                                            target_size = (299, 299),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
Found 1002 images belonging to 6 classes.
```

```
class_name = train_set.class_indices
print(class_name)
```

```
{'Iodine Deficiency': 0, 'Vitamin B12': 1, 'Vitamin D': 2, 'Zinc': 3, 'healthy': 4, 'iron': 5}
```

```
xception_model = Sequential()
pretrained_model = Xception(input_shape=(299,299,3), weights='imagenet', include_top=False,
                  pooling='avg',classes=6)
for layer in pretrained_model.layers:
        layer.trainable=False

xception_model.add(pretrained_model)
xception_model.add(Flatten())
xception_model.add(Dense(512, activation='relu'))
xception_model.add(Dense(6, activation='softmax'))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_ker
83683744/83683744 [==============================] - 5s 0us/step
```

```
xception_model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
 inception_v3 (Functional)   (None, 2048)              21802784

 flatten_6 (Flatten)         (None, 2048)              0

 dense_14 (Dense)            (None, 512)               1049088

 dense_15 (Dense)            (None, 6)                 3078

=================================================================
Total params: 22,854,950
Trainable params: 1,052,166
Non-trainable params: 21,802,784
_________________________________________________________________
```

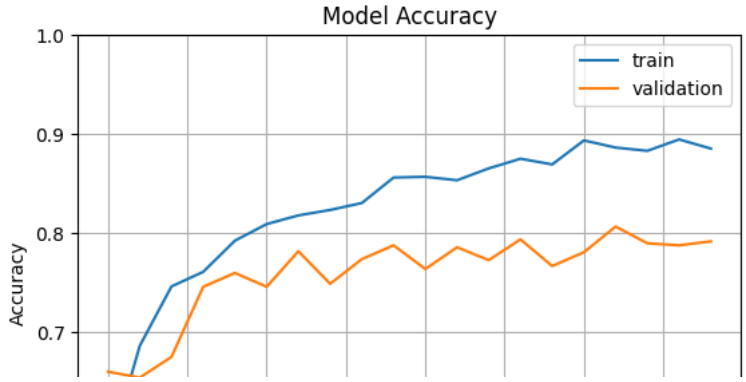
```
xception_model.compile(optimizer=Adam(learning_rate=0.01),loss='categorical_crossentropy',metrics=['accuracy'])
```

```
epochs=20
history = xception_model.fit(train_set,validation_data=test_set,epochs=epochs)
```

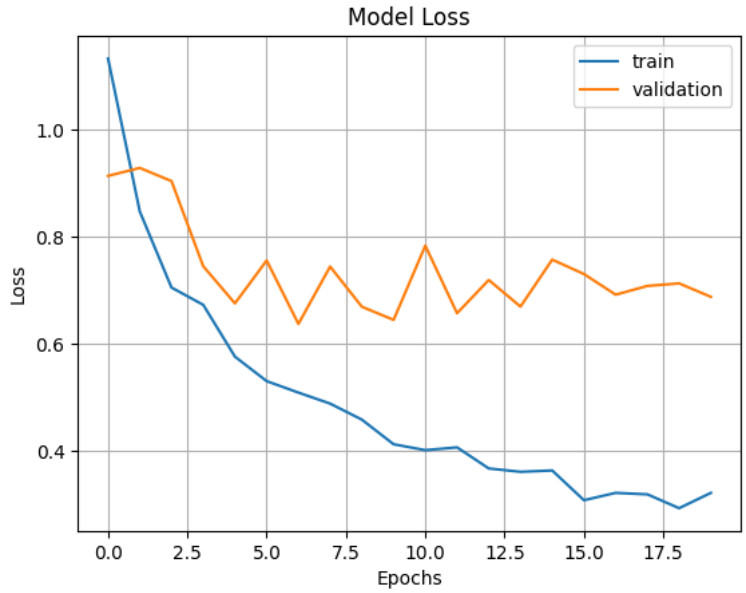
```
Epoch 1/20
125/125 [==============================] - 3243s 26s/step - loss: 1.1341 - accuracy: 0.5743 - val_loss: 0.9142 - val_accuracy: 0.65
Epoch 2/20
125/125 [==============================] - 111s 884ms/step - loss: 0.8477 - accuracy: 0.6858 - val_loss: 0.9292 - val_accuracy: 0.6
Epoch 3/20
125/125 [==============================] - 110s 876ms/step - loss: 0.7052 - accuracy: 0.7459 - val_loss: 0.9045 - val_accuracy: 0.6
Epoch 4/20
125/125 [==============================] - 109s 867ms/step - loss: 0.6728 - accuracy: 0.7606 - val_loss: 0.7451 - val_accuracy: 0.7
Epoch 5/20
125/125 [==============================] - 107s 858ms/step - loss: 0.5758 - accuracy: 0.7921 - val_loss: 0.6756 - val_accuracy: 0.7
Epoch 6/20
125/125 [==============================] - 108s 863ms/step - loss: 0.5300 - accuracy: 0.8089 - val_loss: 0.7558 - val_accuracy: 0.7
Epoch 7/20
125/125 [==============================] - 112s 892ms/step - loss: 0.5085 - accuracy: 0.8177 - val_loss: 0.6371 - val_accuracy: 0.7
Epoch 8/20
125/125 [==============================] - 108s 862ms/step - loss: 0.4879 - accuracy: 0.8232 - val_loss: 0.7444 - val_accuracy: 0.7
Epoch 9/20
125/125 [==============================] - 108s 860ms/step - loss: 0.4579 - accuracy: 0.8302 - val_loss: 0.6693 - val_accuracy: 0.7
Epoch 10/20
125/125 [==============================] - 108s 858ms/step - loss: 0.4118 - accuracy: 0.8559 - val_loss: 0.6446 - val_accuracy: 0.7
Epoch 11/20
125/125 [==============================] - 107s 850ms/step - loss: 0.4007 - accuracy: 0.8567 - val_loss: 0.7836 - val_accuracy: 0.7
Epoch 12/20
125/125 [==============================] - 105s 841ms/step - loss: 0.4059 - accuracy: 0.8532 - val_loss: 0.6571 - val_accuracy: 0.7
Epoch 13/20
125/125 [==============================] - 106s 845ms/step - loss: 0.3664 - accuracy: 0.8652 - val_loss: 0.7193 - val_accuracy: 0.7
Epoch 14/20
125/125 [==============================] - 105s 837ms/step - loss: 0.3602 - accuracy: 0.8749 - val_loss: 0.6695 - val_accuracy: 0.7
Epoch 15/20
125/125 [==============================] - 149s 1s/step - loss: 0.3627 - accuracy: 0.8692 - val_loss: 0.7575 - val_accuracy: 0.7665
Epoch 16/20
125/125 [==============================] - 106s 849ms/step - loss: 0.3070 - accuracy: 0.8934 - val_loss: 0.7305 - val_accuracy: 0.7
Epoch 17/20
125/125 [==============================] - 106s 849ms/step - loss: 0.3207 - accuracy: 0.8862 - val_loss: 0.6918 - val_accuracy: 0.8
Epoch 18/20
125/125 [==============================] - 106s 847ms/step - loss: 0.3180 - accuracy: 0.8829 - val_loss: 0.7082 - val_accuracy: 0.7
Epoch 19/20
125/125 [==============================] - 106s 846ms/step - loss: 0.2920 - accuracy: 0.8944 - val_loss: 0.7130 - val_accuracy: 0.7
Epoch 20/20
125/125 [==============================] - 104s 835ms/step - loss: 0.3207 - accuracy: 0.8852 - val_loss: 0.6878 - val_accuracy: 0.7
```

```
xception_model.save("/content/gdrive/MyDrive/augmentation 8000/xception.h5")
```

```
fig1 = plt.gcf()
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.axis(ymin=0.5,ymax=1)
plt.grid()
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.show()
```




```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.grid()
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.show()
```

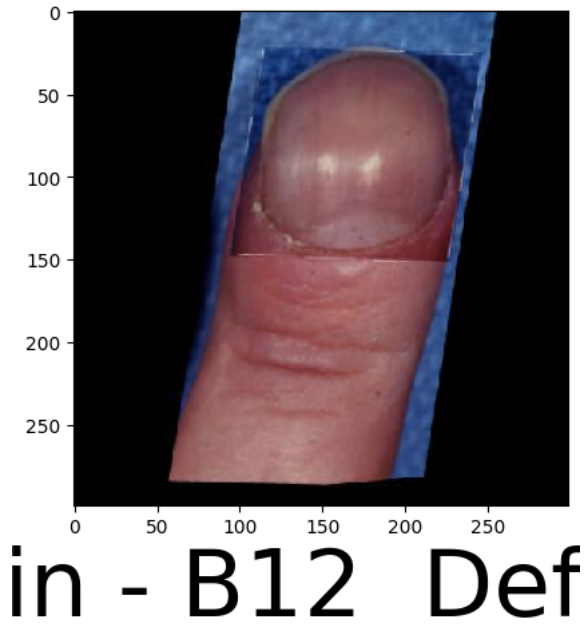



```
from keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model("/content/gdrive/MyDrive/augmentation 8000/xception.h5")
```

```
import numpy as np
def predictImage(filename,model):
  img1=image.load_img(filename,target_size=(299,299))
  plt.imshow(img1)
  Y=image.img_to_array(img1)
  X=np.expand_dims(Y,axis=0)
  pred=model.predict(X/255)
  print(pred)
  pred = np.array(pred)
  val = np.argmax(pred)
  print(val)
  if (val==0).all():
    plt.xlabel("Iodine Deficiency",fontsize=50)
  elif (val==1).all():
    plt.xlabel("Iron Deficiency",fontsize=50)
  elif (val==2).all():
    plt.xlabel("Vitamin - B12  Deficiency",fontsize=50)
  elif (val==3).all():
    plt.xlabel("Vitamin D - Deficiency",fontsize=25)
  elif (val==4).all():
    plt.xlabel("Zinc Deficiency",fontsize=50)
  elif (val==5).all():
    plt.xlabel("healthy",fontsize=25)
```

```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Iodine Deficiency/Iodine Deficiency_original_Screen-Shot-2021-10-26-at-10-55-01-A
```

```
1/1 [==============================] - 3s 3s/step
[[1.4412683e-01 1.7236900e-02 7.4900180e-01 1.7844034e-02 7.1769923e-02
  2.0544147e-05]]
2
```

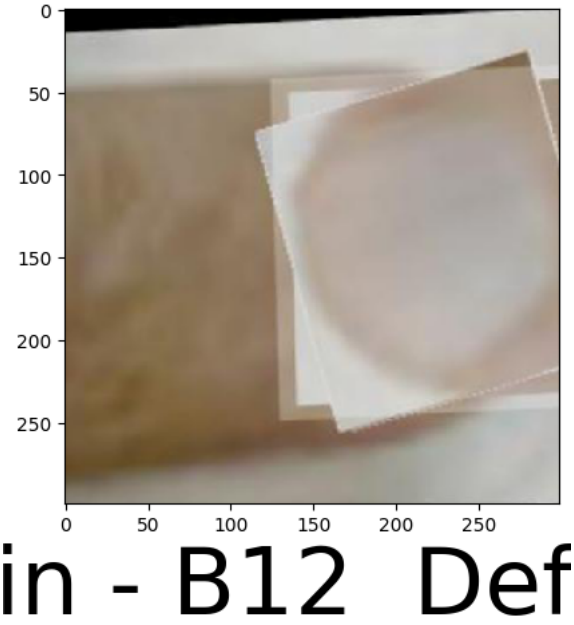



```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Iron Deficiency/Iron Deficiency_original_112_JPG.rf.4a61af6ceaa29030b5965935eae3c
```

```
1/1 [==============================] - 0s 47ms/step
[[1.2578721e-01 7.1040863e-01 2.5478872e-02 8.6294105e-03 1.2921669e-01
  4.7909268e-04]]
1
```

```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Vitamin - B12  Deficiency/Vitamin - B12  Deficiency_original_Screen-Shot-2021-10-
```

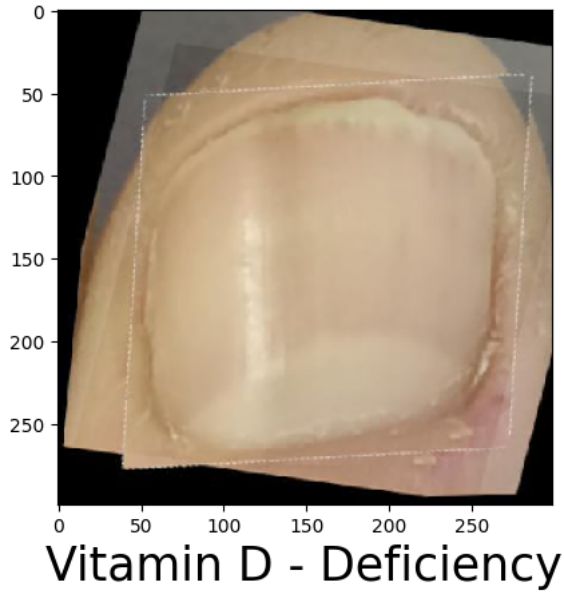
```
1/1 [==============================] - 0s 45ms/step
[[3.1044530e-02 3.8101595e-02 9.2412591e-01 1.9836647e-04 8.3671941e-04
  5.6927935e-03]]
2
```



in - B12 Def

```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Vitamin D - Deficiency/Vitamin D - Deficiency_original_Screen-Shot-2021-11-20-at-
```

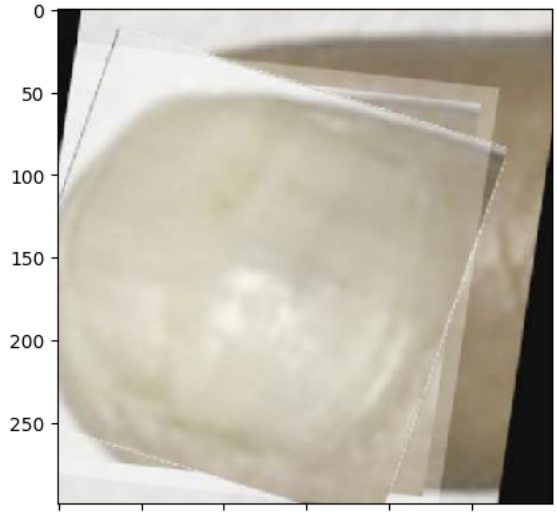
```
1/1 [==============================] - 0s 28ms/step
[[0.00152076 0.29042125 0.03032159 0.5030235  0.1166429  0.05807005]]
3
```



Vitamin D - Deficiency

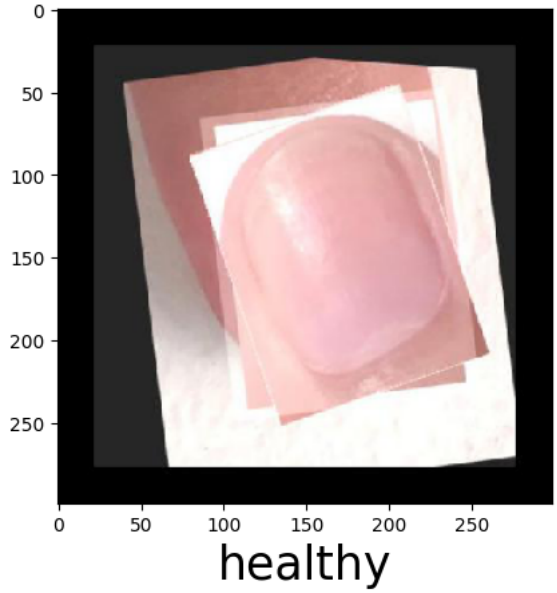
```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Zinc Deficiency/Zinc Deficiency_original_Screen-Shot-2021-11-22-at-1-01-12-PM_png
```

```
1/1 [==============================] - 0s 27ms/step
[[0.4563299  0.04907864 0.33956146 0.0005035  0.15108983 0.00343669]]
0
```



```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/healthy/healthy_original_Screen-Shot-2021-11-15-at-12-52-17-PM_png.rf.b5628687bd9
```

```
1/1 [==============================] - 0s 97ms/step
[[0.00148466 0.02746078 0.01187978 0.00133189 0.2960549  0.6617879 ]]
5
```




```
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import numpy as np


saved_model = load_model("/content/gdrive/MyDrive/augmentation 8000/xception.h5")
all_y_pred = []
all_y_true = []

for i in range(len(test_set)):
    x, y = test_set[i]
    y_pred = saved_model.predict(x)

    all_y_pred.append(y_pred)
    all_y_true.append(y)

all_y_pred = np.concatenate(all_y_pred, axis=0)
all_y_true = np.concatenate(all_y_true, axis=0)
```

```
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 52ms/step
```

```
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 71ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 75ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 1s 715ms/step
```

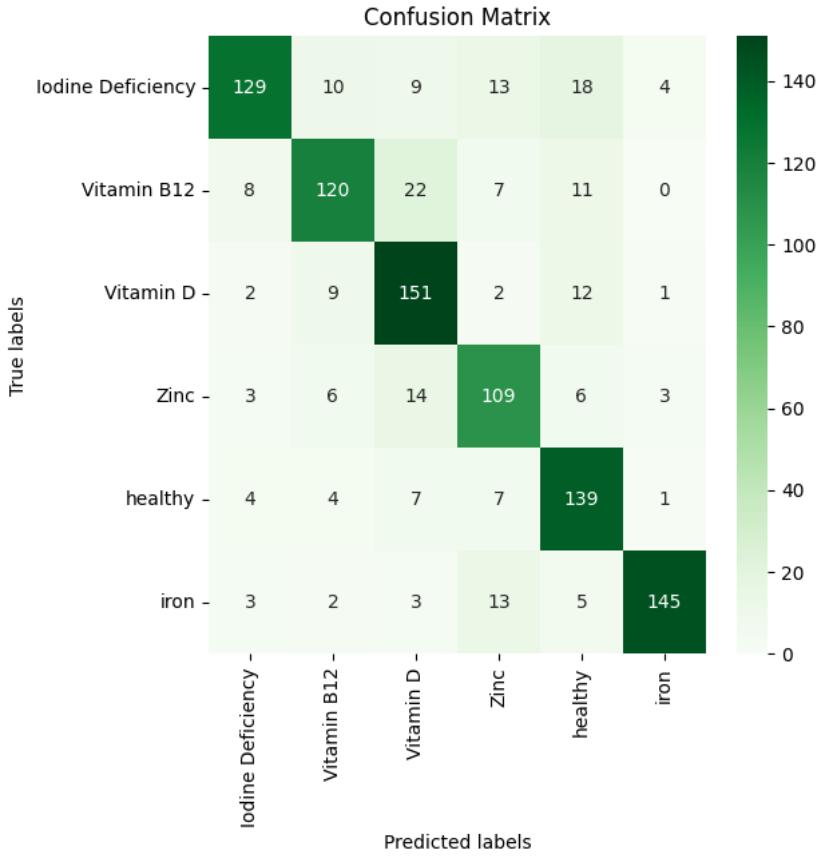
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# compute confusion matrix
cm = confusion_matrix(all_y_true.argmax(axis=1), all_y_pred.argmax(axis=1))

# create heatmap from confusion matrix
fig, ax = plt.subplots(figsize=(6,6))
sns.heatmap(cm, annot=True, cmap="Greens", fmt="d", xticklabels=train_set.class_indices.keys(),
            yticklabels=train_set.class_indices.keys(), ax=ax)

# set axis labels and title
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```




```
from sklearn.metrics import classification_report

# Get the predicted class labels
y_pred = np.argmax(all_y_pred, axis=1)
```

```
# Get the true class labels
y_true = np.argmax(all_y_true, axis=1)

# Compute classification report
report = classification_report(y_true, y_pred, target_names=train_set.class_indices.keys())

# Print classification report
print(report)
```

```
                      precision    recall  f1-score   support

     Iodine Deficiency       0.87      0.70      0.78       183
           Vitamin B12       0.79      0.71      0.75       168
             Vitamin D       0.73      0.85      0.79       177
                  Zinc       0.72      0.77      0.75       141
               healthy       0.73      0.86      0.79       162
                  iron       0.94      0.85      0.89       171

              accuracy                           0.79      1002
             macro avg       0.80      0.79      0.79      1002
          weighted avg       0.80      0.79      0.79      1002
```

```
# Import necessary libraries
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator

# Load the saved model
model = load_model('/content/gdrive/MyDrive/augmentation 8000/xception.h5')
scores = model.evaluate(test_set, steps=len(test_set), verbose=1)
scores2 = model.evaluate(train_set, steps=len(test_set), verbose=1)


# Print the accuracy score
print("Test Accuracy: %.2f%%" % (scores[1]*100))
print("Train Accuracy: %.2f%%" % (scores2[1]*100))
```

```
     32/32 [==============================] - 9s 237ms/step - loss: 0.6878 - accuracy: 0.7914
     32/32 [==============================] - 27s 855ms/step - loss: 0.2636 - accuracy: 0.9043
     Test Accuracy: 79.14%
```