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) (
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (
Requirement already satisfied: tqdm>=4.9.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (4.
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.
Requirement already satisfied: Pillow>=5.2.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (
Initialised with 3310 image(s) found.
Output directory set to /content/gdrive/MyDrive/augmentation 8000/resized_Balanced dataset/output.Processi
```

```
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/aug7k_split dataset/train'
test_path = '/content/gdrive/MyDrive/aug7k_split dataset/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/aug7k_split dataset/train',target_size=(299
                                            batch_size=32,
                                            class_mode='categorical')
```

```
Found 5598 images belonging to 6 classes.
```

```
test_set = test_datagen.flow_from_directory('/content/gdrive/MyDrive/aug7k_split dataset/val',
                                            target_size = (299, 299),
```

```
                                                batch_size = 32,
                                                class_mode = 'categorical')
```

```
Found 1402 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_weigh
83683744/83683744 [==============================] - 5s 0us/step
```

```
xception_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 2048)              20861480

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 512)               1049088

 dense_1 (Dense)             (None, 6)                 3078

=================================================================
Total params: 21,913,646
Trainable params: 1,052,166
Non-trainable params: 20,861,480
_________________________________________________________________
```

```
xception_model.compile(optimizer=Adam(learning_rate=0.001),loss='categorical_crossentropy',metrics=['accuracy'])
```

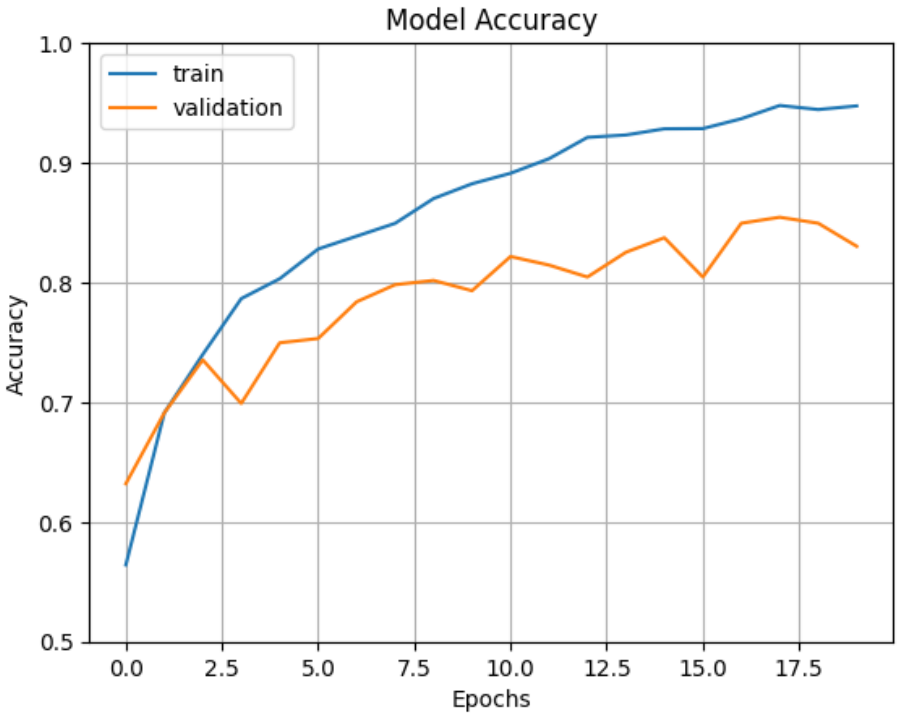
```
epochs=20
history = xception_model.fit(train_set,validation_data=test_set,epochs=epochs)
```

```
Epoch 1/20
175/175 [==============================] - 4021s 23s/step - loss: 1.1514 - accuracy: 0.5641 - val_loss: 0.
Epoch 2/20
175/175 [==============================] - 138s 787ms/step - loss: 0.8359 - accuracy: 0.6911 - val_loss: 0
Epoch 3/20
175/175 [==============================] - 136s 777ms/step - loss: 0.7118 - accuracy: 0.7399 - val_loss: 0
Epoch 4/20
175/175 [==============================] - 138s 789ms/step - loss: 0.6006 - accuracy: 0.7865 - val_loss: 0
Epoch 5/20
175/175 [==============================] - 136s 778ms/step - loss: 0.5596 - accuracy: 0.8031 - val_loss: 0
Epoch 6/20
175/175 [==============================] - 137s 782ms/step - loss: 0.4792 - accuracy: 0.8280 - val_loss: 0
Epoch 7/20
175/175 [==============================] - 136s 777ms/step - loss: 0.4510 - accuracy: 0.8387 - val_loss: 0
Epoch 8/20
175/175 [==============================] - 139s 796ms/step - loss: 0.4102 - accuracy: 0.8494 - val_loss: 0
Epoch 9/20
```

```
175/175 [==============================] - 138s 789ms/step - loss: 0.3621 - accuracy: 0.8701 - val_loss: 0
Epoch 10/20
175/175 [==============================] - 136s 776ms/step - loss: 0.3321 - accuracy: 0.8825 - val_loss: 0
Epoch 11/20
175/175 [==============================] - 137s 782ms/step - loss: 0.3085 - accuracy: 0.8912 - val_loss: 0
Epoch 12/20
175/175 [==============================] - 137s 781ms/step - loss: 0.2785 - accuracy: 0.9034 - val_loss: 0
Epoch 13/20
175/175 [==============================] - 136s 778ms/step - loss: 0.2375 - accuracy: 0.9212 - val_loss: 0
Epoch 14/20
175/175 [==============================] - 136s 777ms/step - loss: 0.2243 - accuracy: 0.9232 - val_loss: 0
Epoch 15/20
175/175 [==============================] - 138s 786ms/step - loss: 0.2076 - accuracy: 0.9284 - val_loss: 0
Epoch 16/20
175/175 [==============================] - 137s 784ms/step - loss: 0.2132 - accuracy: 0.9285 - val_loss: 0
Epoch 17/20
175/175 [==============================] - 138s 790ms/step - loss: 0.1859 - accuracy: 0.9368 - val_loss: 0
Epoch 18/20
175/175 [==============================] - 136s 777ms/step - loss: 0.1599 - accuracy: 0.9478 - val_loss: 0
Epoch 19/20
175/175 [==============================] - 137s 783ms/step - loss: 0.1581 - accuracy: 0.9444 - val_loss: 0
Epoch 20/20
175/175 [==============================] - 137s 780ms/step - loss: 0.1557 - accuracy: 0.9475 - val_loss: 0
```

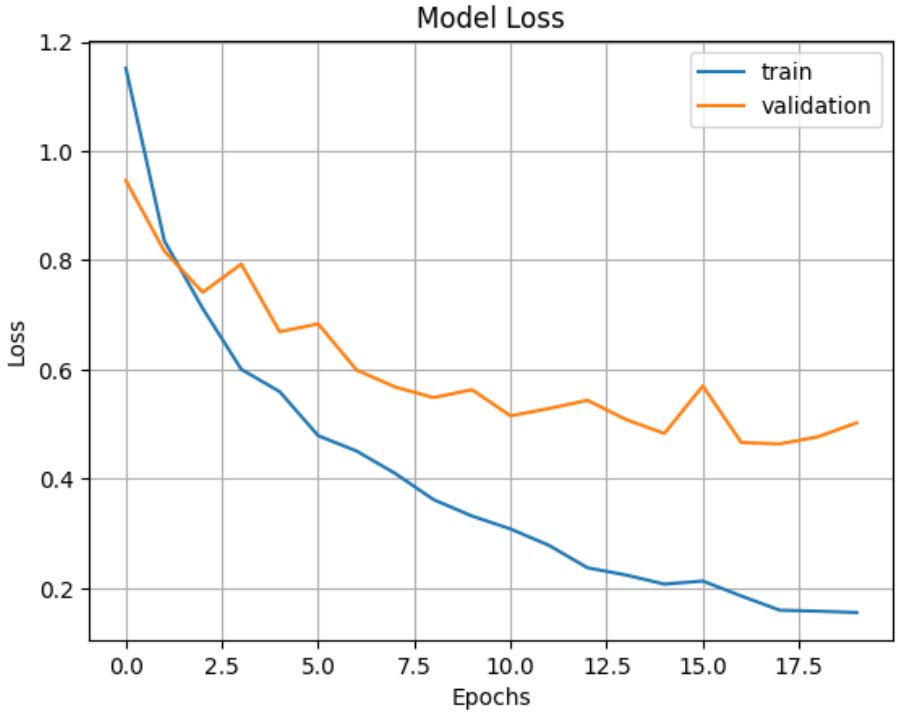
```
xception_model.save("/content/gdrive/MyDrive/aug7k_split dataset/xception.h5")
```

```
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()
```

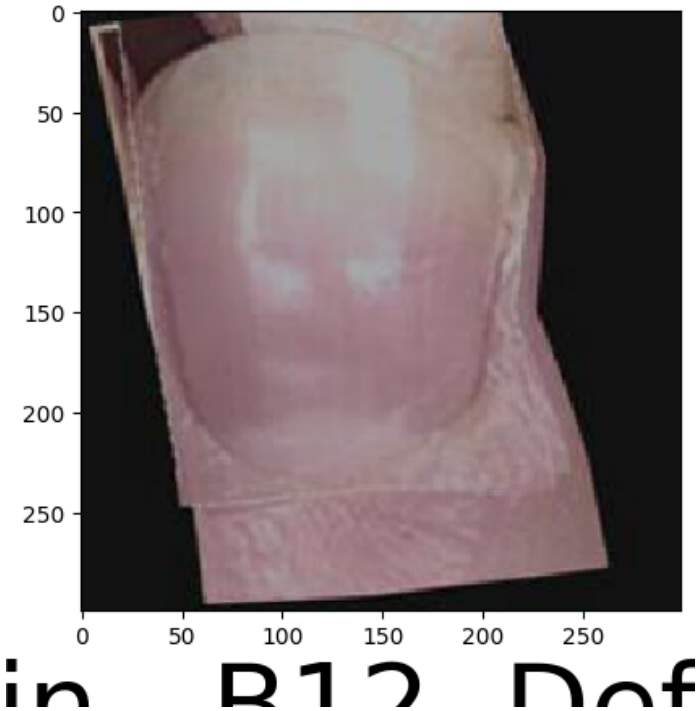



```
from keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model("/content/gdrive/MyDrive/aug7k_split dataset/xception.h5")
```

```
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)
```

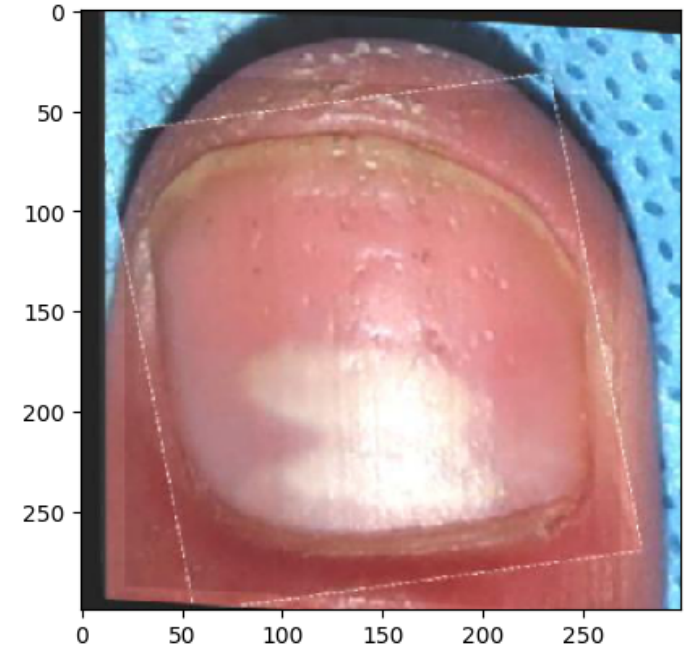
```
predictImage("/content/gdrive/MyDrive/aug7k_split dataset/val/Zinc/Zinc_original_Screen-Shot-2021-11-17-at-2-15-
```

```
1/1 [==============================] - 0s 62ms/step
[[0.04247129 0.1256127  0.34430793 0.28948155 0.15713476 0.04099176]]
2
```



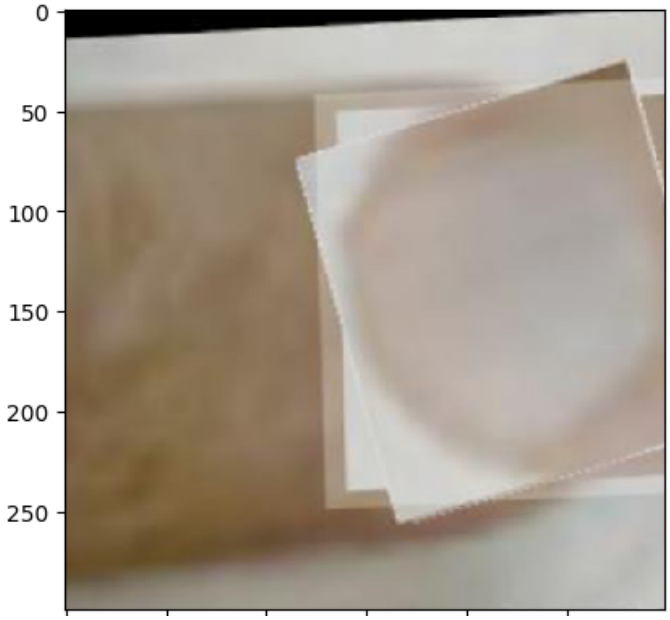

```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Iron Deficiency/Iron Deficiency_original_112_JPG.rf.4a61
```

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



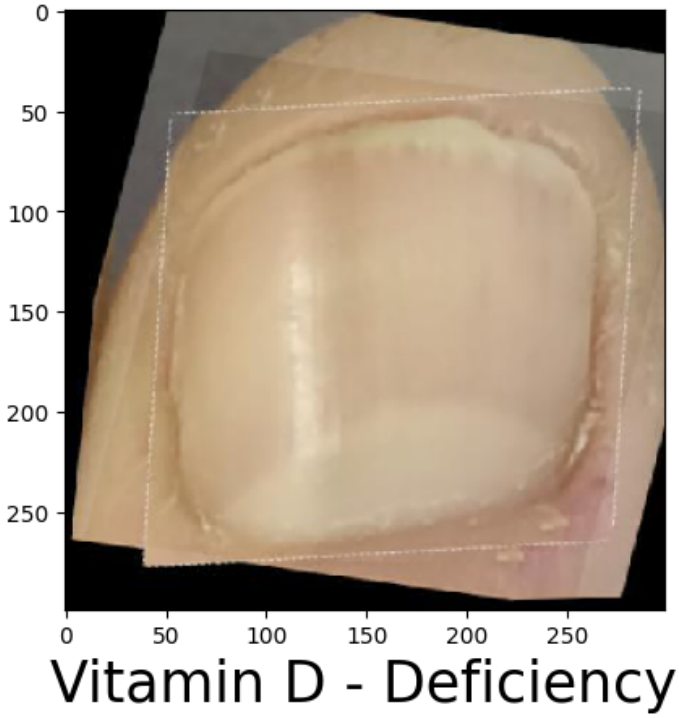

```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Vitamin - B12  Deficiency/Vitamin - B12  Deficiency_orig
```

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



```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Vitamin D - Deficiency/Vitamin D - Deficiency_original_S
```

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

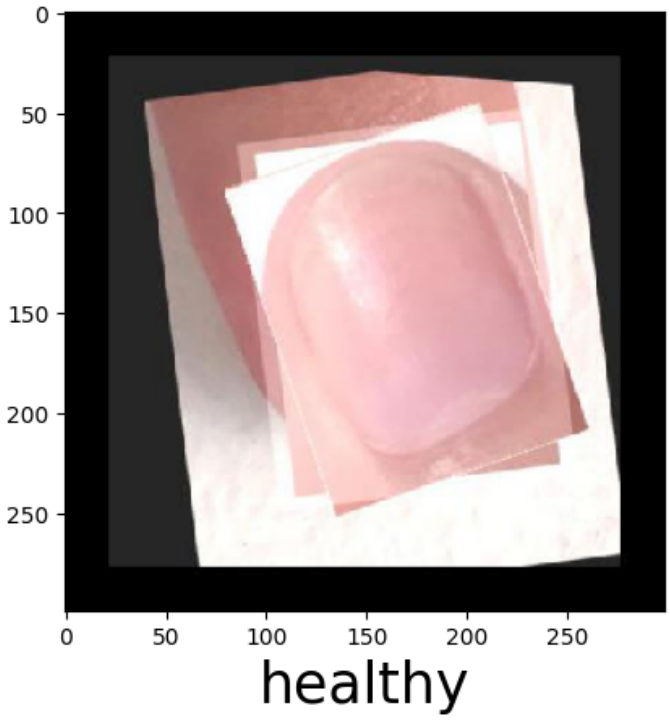



```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Zinc Deficiency/Zinc Deficiency_original_Screen-Shot-202
```

```
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
```

```
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/aug7k_split dataset/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 959ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 71ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 1s 712ms/step
```

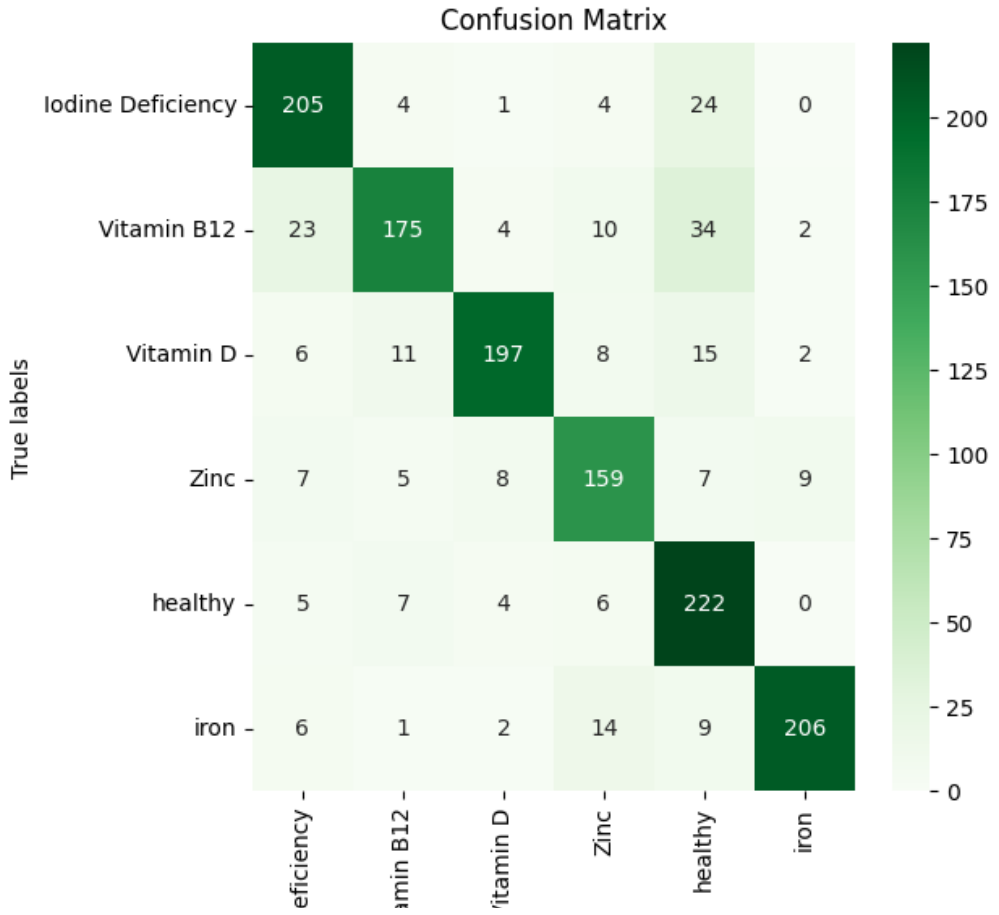
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# 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       0.81      0.86      0.84       238
      Vitamin B12       0.86      0.71      0.78       248
        Vitamin D       0.91      0.82      0.87       239
             Zinc       0.79      0.82      0.80       195
          healthy       0.71      0.91      0.80       244
             iron       0.94      0.87      0.90       238

         accuracy                           0.83      1402
        macro avg       0.84      0.83      0.83      1402
     weighted avg       0.84      0.83      0.83      1402
```

```
# 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/aug7k_split dataset/xception.h5')
scores = model.evaluate(test_set, steps=len(test_set), verbose=1)
scores2 = model.evaluate(train_set, steps=len(train_set), verbose=1)
```

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

```
44/44 [==============================] - 11s 226ms/step - loss: 0.5027 - accuracy: 0.8302
175/175 [==============================] - 131s 746ms/step - loss: 0.1601 - accuracy: 0.9452
Test Accuracy: 83.02%
Train Accuracy: 94.52%
```

✓ 2m 25s completed at 08:05