DATA AUGMENTATION

Double-click (or enter) to edit

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

```
    Copying files: 3310 files [01:21, 40.78 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/Resized dataset/resized_split dataset/train'
test_path = '/content/gdrive/MyDrive/Resized dataset/resized_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/Resized dataset/resized_split dataset/trair
                                                 batch_size=32,
                                                 class_mode='categorical')
```

```
    Found 2646 images belonging to 6 classes.
```

```
test_set = test_datagen.flow_from_directory('/content/gdrive/MyDrive/Resized dataset/resized_split dataset/val',
                                            target_size = (299, 299),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
    Found 664 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 [==============================] - 0s 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=0.01),loss='categorical_crossentropy',metrics=['accuracy'])
```

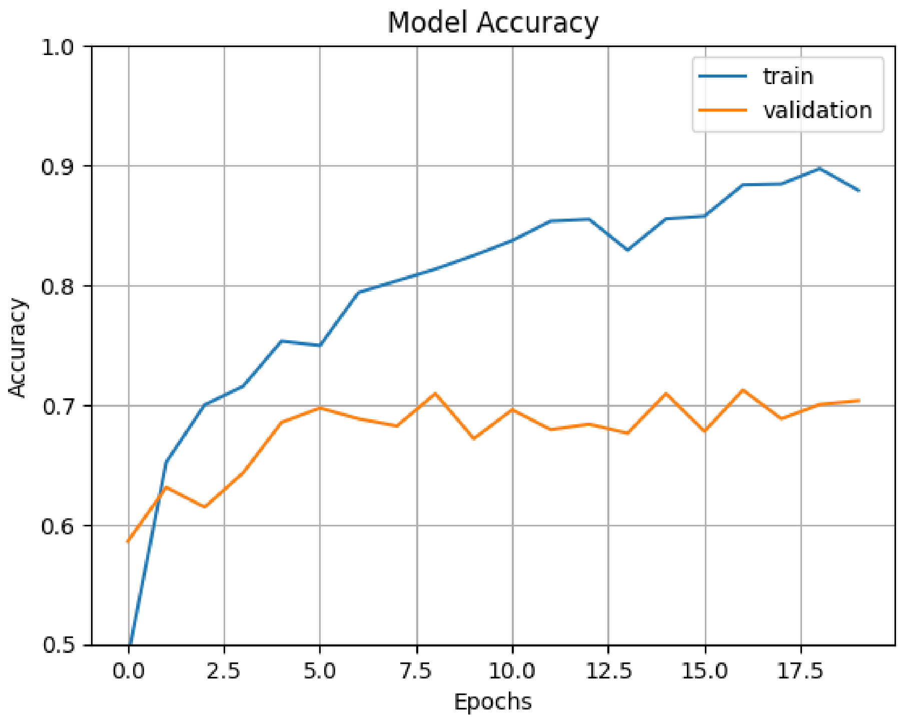
```
epochs=20
history = xception_model.fit(train_set,validation_data=test_set,epochs=epochs)
```

```
    Epoch 1/20
    83/83 [==============================] - 496s 6s/step - loss: 1.6396 - accuracy: 0.4883 - val_loss: 1.0887
    Epoch 2/20
    83/83 [==============================] - 69s 836ms/step - loss: 0.9361 - accuracy: 0.6519 - val_loss: 1.01
    Epoch 3/20
    83/83 [==============================] - 70s 839ms/step - loss: 0.8281 - accuracy: 0.6999 - val_loss: 1.04
    Epoch 4/20
    83/83 [==============================] - 70s 845ms/step - loss: 0.7786 - accuracy: 0.7154 - val_loss: 0.97
    Epoch 5/20
    83/83 [==============================] - 71s 849ms/step - loss: 0.6672 - accuracy: 0.7532 - val_loss: 0.86
    Epoch 6/20
    83/83 [==============================] - 70s 843ms/step - loss: 0.6659 - accuracy: 0.7494 - val_loss: 0.93
    Epoch 7/20
    83/83 [==============================] - 71s 854ms/step - loss: 0.5632 - accuracy: 0.7937 - val_loss: 0.93
    Epoch 8/20
    83/83 [==============================] - 71s 857ms/step - loss: 0.5177 - accuracy: 0.8035 - val_loss: 1.02
    Epoch 9/20
```

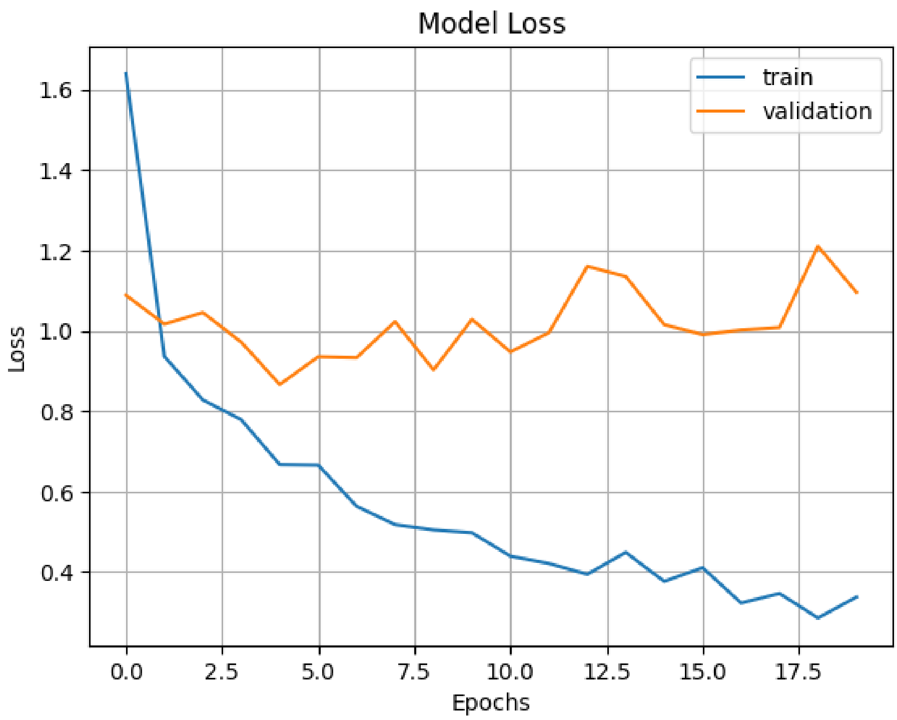
```
83/83 [==============================] - 70s 848ms/step - loss: 0.5047 - accuracy: 0.8133 - val_loss: 0.90
Epoch 10/20
83/83 [==============================] - 72s 867ms/step - loss: 0.4975 - accuracy: 0.8246 - val_loss: 1.02
Epoch 11/20
83/83 [==============================] - 69s 829ms/step - loss: 0.4393 - accuracy: 0.8371 - val_loss: 0.94
Epoch 12/20
83/83 [==============================] - 73s 874ms/step - loss: 0.4211 - accuracy: 0.8534 - val_loss: 0.99
Epoch 13/20
83/83 [==============================] - 75s 904ms/step - loss: 0.3943 - accuracy: 0.8549 - val_loss: 1.16
Epoch 14/20
83/83 [==============================] - 78s 933ms/step - loss: 0.4487 - accuracy: 0.8292 - val_loss: 1.13
Epoch 15/20
83/83 [==============================] - 76s 913ms/step - loss: 0.3765 - accuracy: 0.8553 - val_loss: 1.01
Epoch 16/20
83/83 [==============================] - 72s 868ms/step - loss: 0.4107 - accuracy: 0.8575 - val_loss: 0.99
Epoch 17/20
83/83 [==============================] - 73s 880ms/step - loss: 0.3227 - accuracy: 0.8836 - val_loss: 1.00
Epoch 18/20
83/83 [==============================] - 72s 864ms/step - loss: 0.3465 - accuracy: 0.8844 - val_loss: 1.00
Epoch 19/20
83/83 [==============================] - 72s 861ms/step - loss: 0.2854 - accuracy: 0.8972 - val_loss: 1.21
Epoch 20/20
83/83 [==============================] - 71s 857ms/step - loss: 0.3375 - accuracy: 0.8791 - val_loss: 1.09
```

```
xception_model.save("/content/gdrive/MyDrive/Resized dataset/xception_orig.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()
```



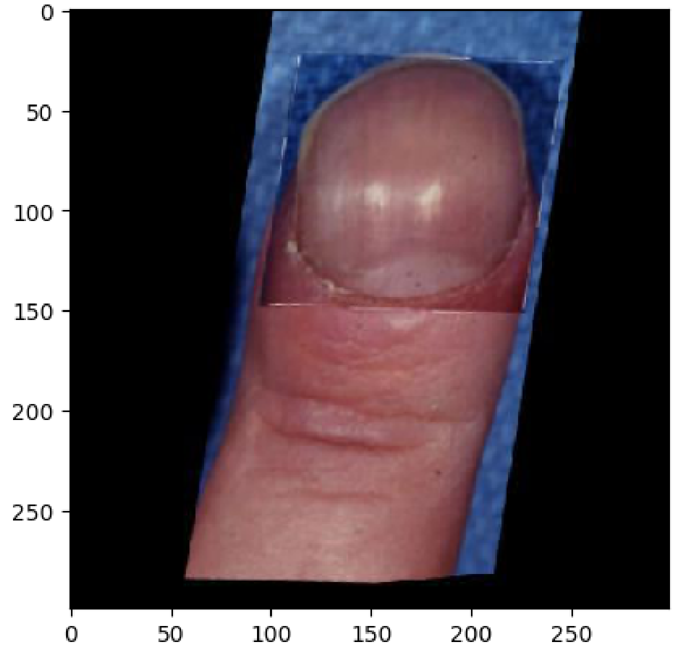
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.grid()
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.show()
```



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

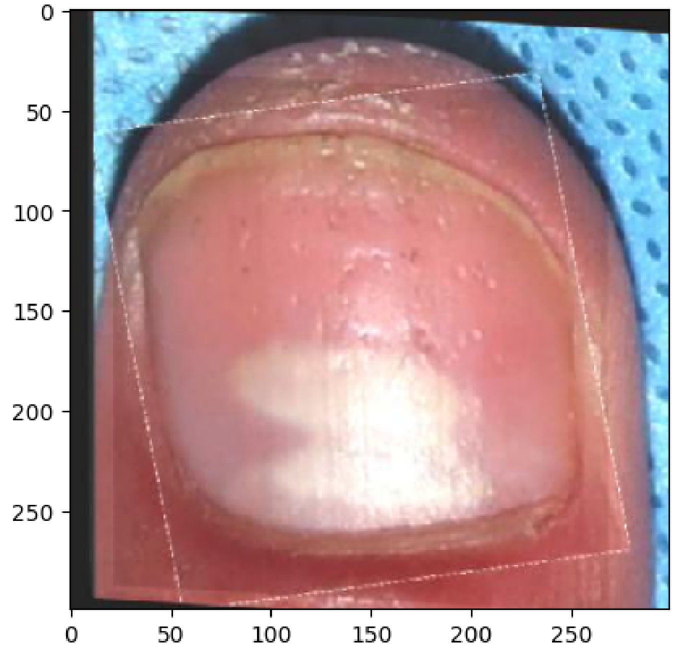
```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Iodine Deficiency/Iodine Deficiency_original_Screen-Shot
1/1 [==============================] - 3s 3s/step
[[1.4412683e-01 1.7236900e-02 7.4900180e-01 1.7844034e-02 7.1769923e-02
  2.0544147e-05]]
2
```



in B12 Def

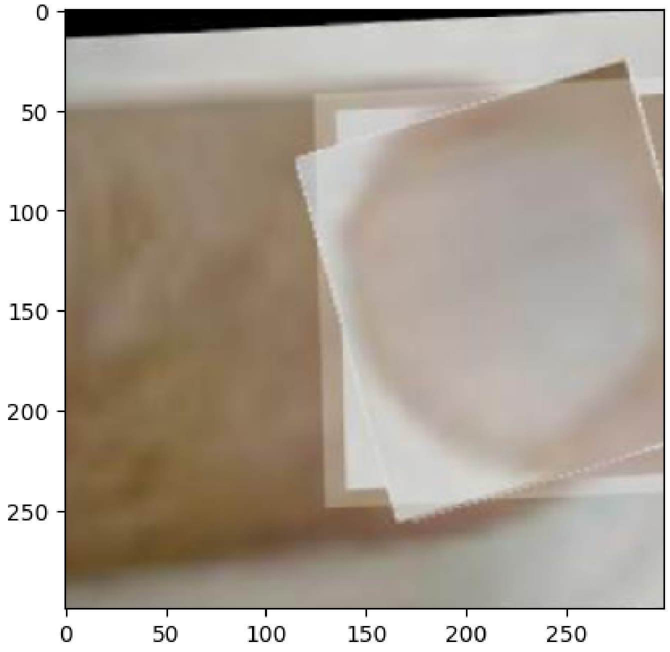
```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Iron Deficiency/Iron Deficiency_original_112_JPG.rf.4a61
```

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



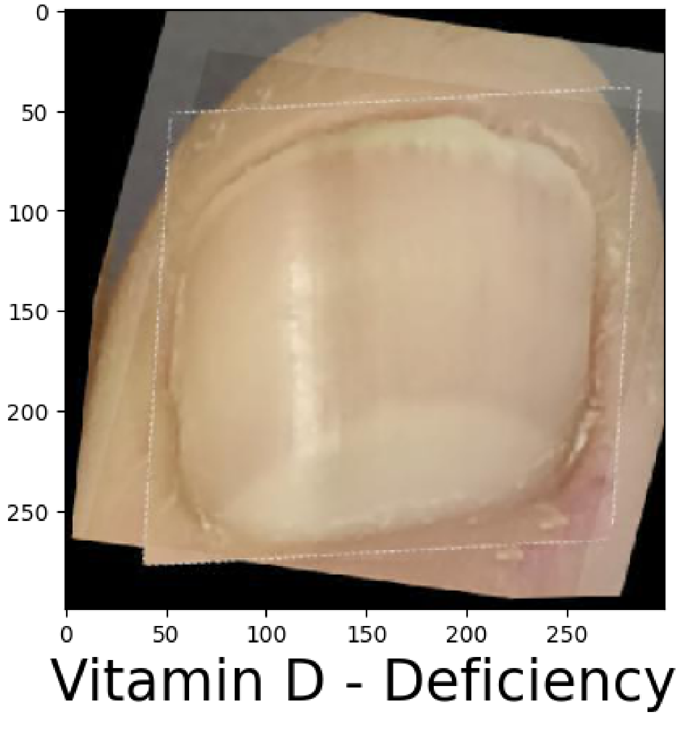
ron Deficien

```
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
```

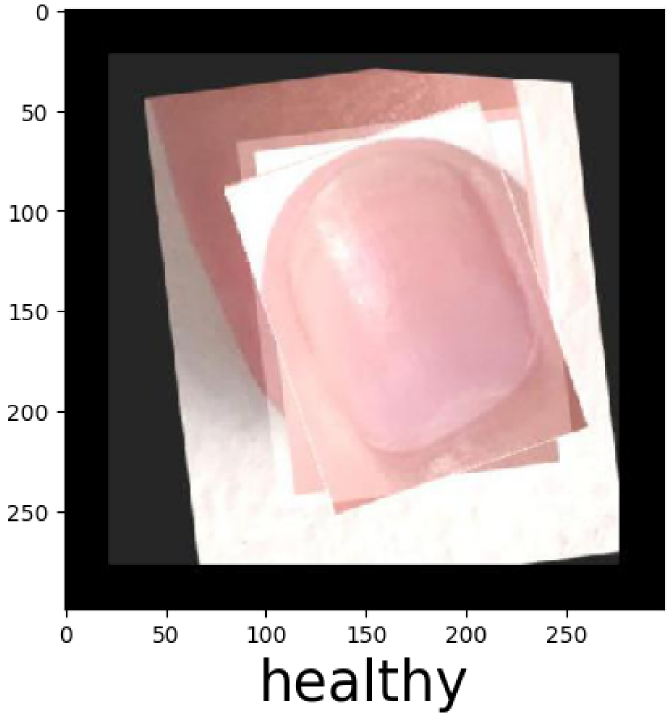
```
1/1 [==============================] - 0s 28ms/step
[[0.00152076 0.29042125 0.03032159 0.5030235  0.1166429  0.05807005]]
3
```



```
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/Resized dataset/xception_orig.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 802ms/step
     1/1 [==============================] - 0s 40ms/step
     1/1 [==============================] - 0s 50ms/step
     1/1 [==============================] - 0s 40ms/step
     1/1 [==============================] - 0s 52ms/step
     1/1 [==============================] - 0s 49ms/step
     1/1 [==============================] - 0s 78ms/step
     1/1 [==============================] - 0s 60ms/step
     1/1 [==============================] - 0s 74ms/step
     1/1 [==============================] - 0s 50ms/step
     1/1 [==============================] - 0s 56ms/step
     1/1 [==============================] - 0s 40ms/step
     1/1 [==============================] - 0s 50ms/step
     1/1 [==============================] - 0s 52ms/step
     1/1 [==============================] - 0s 50ms/step
     1/1 [==============================] - 0s 38ms/step
     1/1 [==============================] - 0s 43ms/step
     1/1 [==============================] - 0s 52ms/step
     1/1 [==============================] - 0s 54ms/step
     1/1 [==============================] - 0s 57ms/step
     1/1 [==============================] - 1s 753ms/step
```

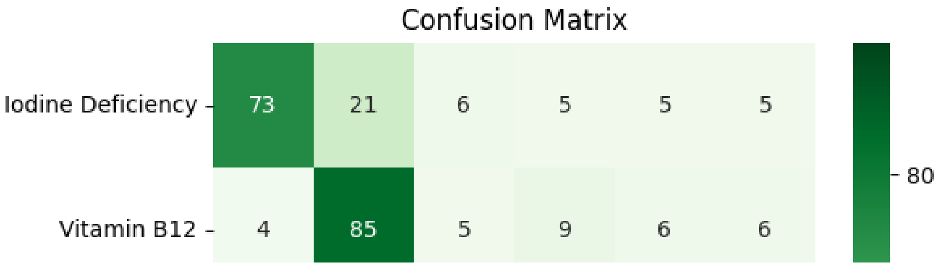
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# 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       0.78      0.63      0.70       115
      Vitamin B12       0.59      0.74      0.66       115
        Vitamin D       0.79      0.65      0.71       117
             Zinc       0.64      0.64      0.64        91
          healthy       0.80      0.70      0.74       112
             iron       0.68      0.85      0.76       114

         accuracy                           0.70       664
        macro avg       0.71      0.70      0.70       664
     weighted avg       0.72      0.70      0.70       664
```

```
# 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/Resized dataset/xception_orig.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))
```

```
21/21 [==============================] - 7s 231ms/step - loss: 1.0958 - accuracy: 0.7033
83/83 [==============================] - 66s 784ms/step - loss: 0.2454 - accuracy: 0.9119
Test Accuracy: 70.33%
Train Accuracy: 91.19%
```

✓ 1m 37s completed at 2:17 PM ● ✕

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.