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: tqdm>=4.9.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (4.
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (
Requirement already satisfied: Pillow>=5.2.0 in /usr/local/lib/python3.10/dist-packages (from augmentor) (
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/resized_Balanced dataset')

# Defining augmentation parameters and generating 5 samples
p.zoom(probability = 0.2, min_factor = 0.8, max_factor = 1.5)
p.flip_top_bottom(probability=0.3)
p.random_brightness(probability=0.3, min_factor=0.3, max_factor=1.1)
p.random_distortion(probability=0.2, grid_width=4, grid_height=4, magnitude=8)
p.sample(7000)
```

```
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: Pillow>=5.2.0 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.
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/resized_Balanced dataset/output.Processing <PIL.Image.Imag
```

```
import os

# specify the folder path
folder_path = "/content/gdrive/MyDrive/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:  1188
```

```
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_Balanced dataset/output'
splitfolders.ratio(input_folder, output= r'/content/gdrive/MyDrive/aug7k_split dataset',
                   seed=42, ratio=(.8, .2),
                   group_prefix=None)
```

```
Copying files: 7000 files [02:41, 43.39 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.inception_v3 import InceptionV3
from tensorflow.keras.applications.inception_v3 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}
```

```
inceptionv3_model = Sequential()
pretrained_model = InceptionV3(input_shape=(299,299,3), weights='imagenet', include_top=False,
                   pooling='avg',classes=6)
for layer in pretrained_model.layers:
        layer.trainable=False

inceptionv3_model.add(pretrained_model)
inceptionv3_model.add(Flatten())
inceptionv3_model.add(Dense(512, activation='relu'))
inceptionv3_model.add(Dense(6, activation='softmax'))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_
87910968/87910968 [==============================] - 5s 0us/step
```

```
inceptionv3_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 2048)              21802784

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 512)               1049088

 dense_1 (Dense)             (None, 6)                 3078

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

```
inceptionv3_model.compile(optimizer=Adam(learning_rate=0.001),loss='categorical_crossentropy',metrics=['accuracy
```

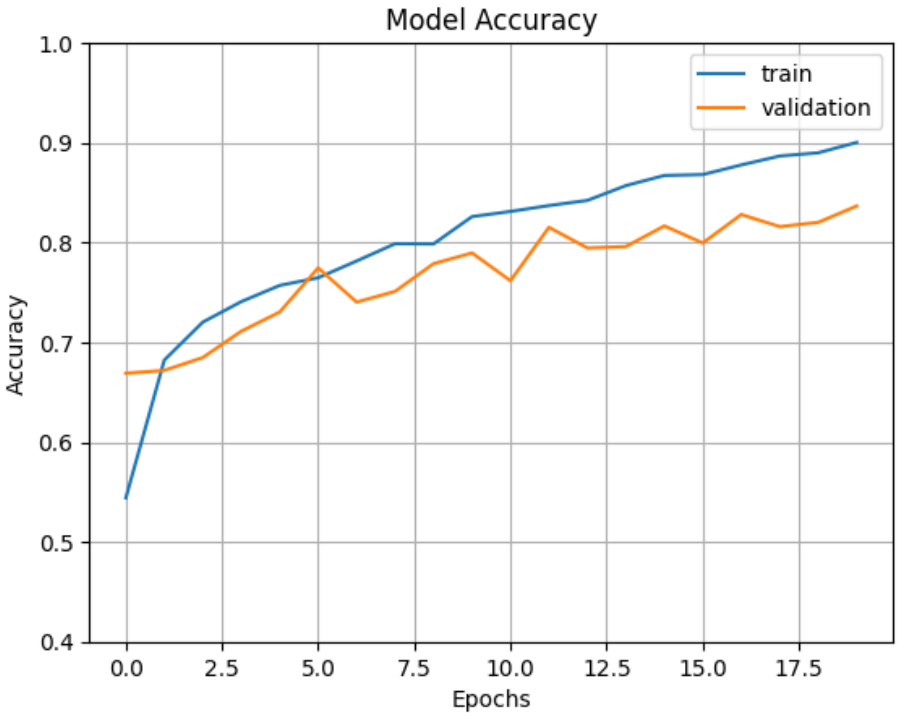
```
epochs=20
history = inceptionv3_model.fit(train_set,validation_data=test_set,epochs=epochs)
```

```
==================] - 177s 931ms/step - loss: 1.2392 - accuracy: 0.5441 - val_loss: 0.9208 - val_accuracy:

==================] - 140s 799ms/step - loss: 0.8631 - accuracy: 0.6822 - val_loss: 0.8585 - val_accuracy:

==================] - 140s 800ms/step - loss: 0.7637 - accuracy: 0.7203 - val_loss: 0.8551 - val_accuracy:

==================] - 143s 820ms/step - loss: 0.7151 - accuracy: 0.7408 - val_loss: 0.7851 - val_accuracy:

==================] - 142s 811ms/step - loss: 0.6600 - accuracy: 0.7571 - val_loss: 0.7294 - val_accuracy:

==================] - 143s 817ms/step - loss: 0.6359 - accuracy: 0.7647 - val_loss: 0.6346 - val_accuracy:

==================] - 148s 845ms/step - loss: 0.5944 - accuracy: 0.7815 - val_loss: 0.7195 - val_accuracy:

==================] - 140s 800ms/step - loss: 0.5399 - accuracy: 0.7989 - val_loss: 0.7272 - val_accuracy:
```

```
==================] - 140s 799ms/step - loss: 0.5381 - accuracy: 0.7987 - val_loss: 0.6080 - val_accuracy:
==================] - 142s 810ms/step - loss: 0.4854 - accuracy: 0.8260 - val_loss: 0.5929 - val_accuracy:
==================] - 138s 789ms/step - loss: 0.4538 - accuracy: 0.8312 - val_loss: 0.6859 - val_accuracy:
==================] - 141s 806ms/step - loss: 0.4530 - accuracy: 0.8371 - val_loss: 0.5381 - val_accuracy:
==================] - 141s 808ms/step - loss: 0.4288 - accuracy: 0.8423 - val_loss: 0.5898 - val_accuracy:
==================] - 146s 833ms/step - loss: 0.4051 - accuracy: 0.8571 - val_loss: 0.5730 - val_accuracy:
==================] - 141s 807ms/step - loss: 0.3648 - accuracy: 0.8673 - val_loss: 0.5058 - val_accuracy:
==================] - 140s 801ms/step - loss: 0.3640 - accuracy: 0.8682 - val_loss: 0.5738 - val_accuracy:
==================] - 140s 799ms/step - loss: 0.3512 - accuracy: 0.8778 - val_loss: 0.4971 - val_accuracy:
==================] - 139s 794ms/step - loss: 0.3109 - accuracy: 0.8867 - val_loss: 0.5268 - val_accuracy:
==================] - 141s 804ms/step - loss: 0.3074 - accuracy: 0.8900 - val_loss: 0.5189 - val_accuracy:
==================] - 139s 796ms/step - loss: 0.2734 - accuracy: 0.9003 - val_loss: 0.4932 - val_accuracy:
```

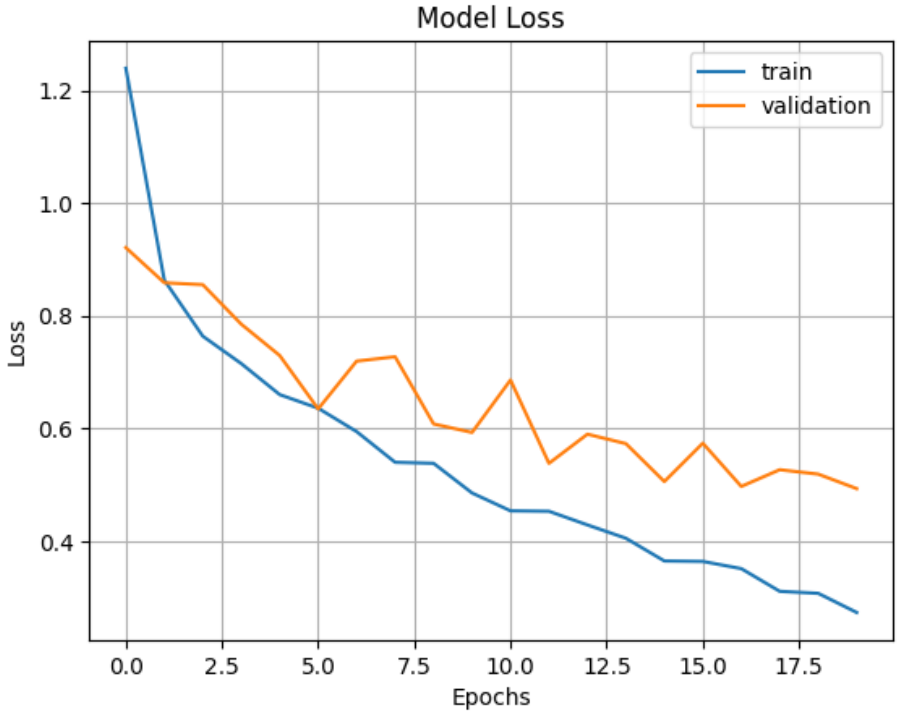
```
inceptionv3_model.save("/content/gdrive/MyDrive/aug7k_split dataset/inceptionv3_aug7k.h5")
```

```
fig1 = plt.gcf()
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.axis(ymin=0.4,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()
```

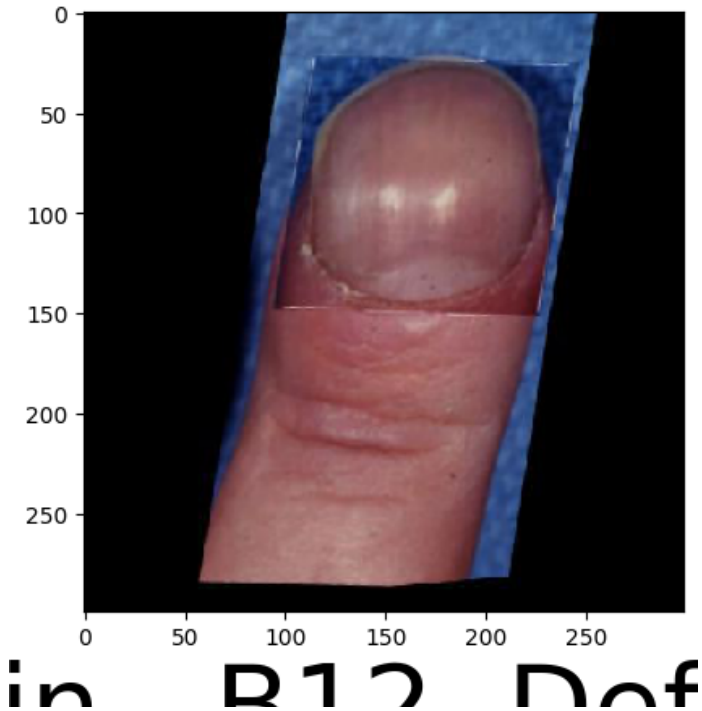



```
from keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model("/content/gdrive/MyDrive/aug7k_split dataset/inceptionv3_aug7k.h5")
```

```
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
```

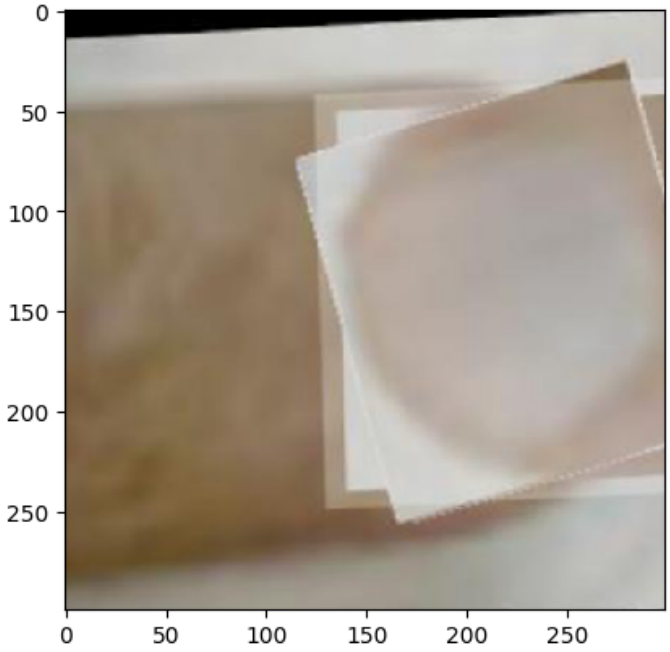



```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Iron Deficiency/Iron Deficiency_original_112_JPG.rf.4a61
```

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

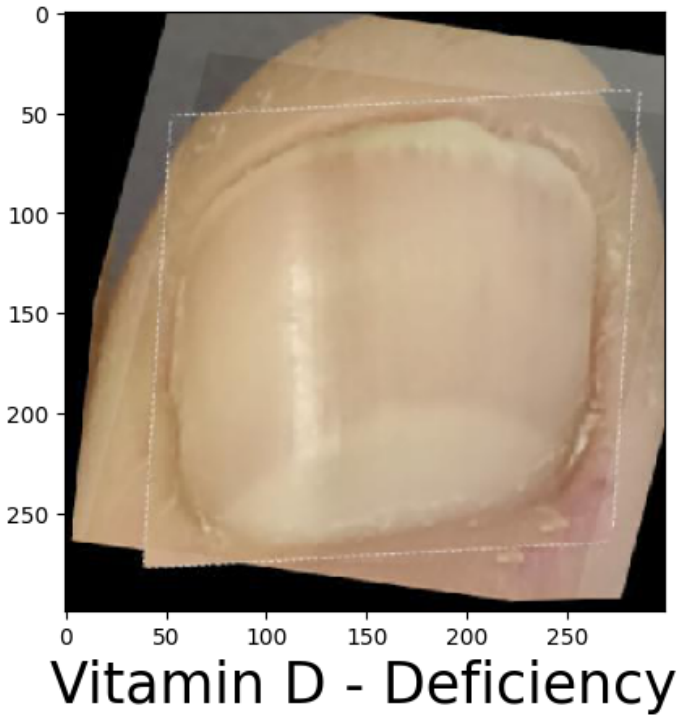

```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Vitamin - B12  Deficiency/Vitamin - B12  Deficiency_orig
```

```
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
```

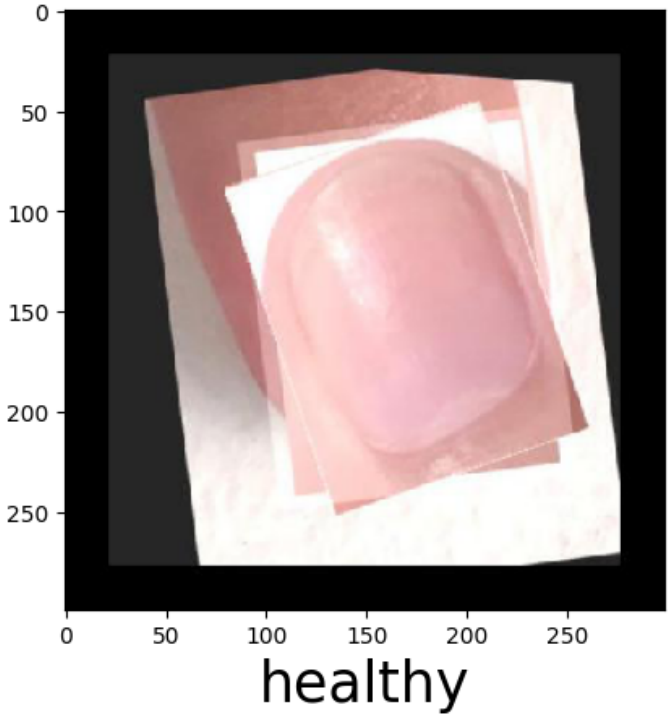



```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Zinc Deficiency/Zinc Deficiency_original_Screen-Shot-202
```

```
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
```



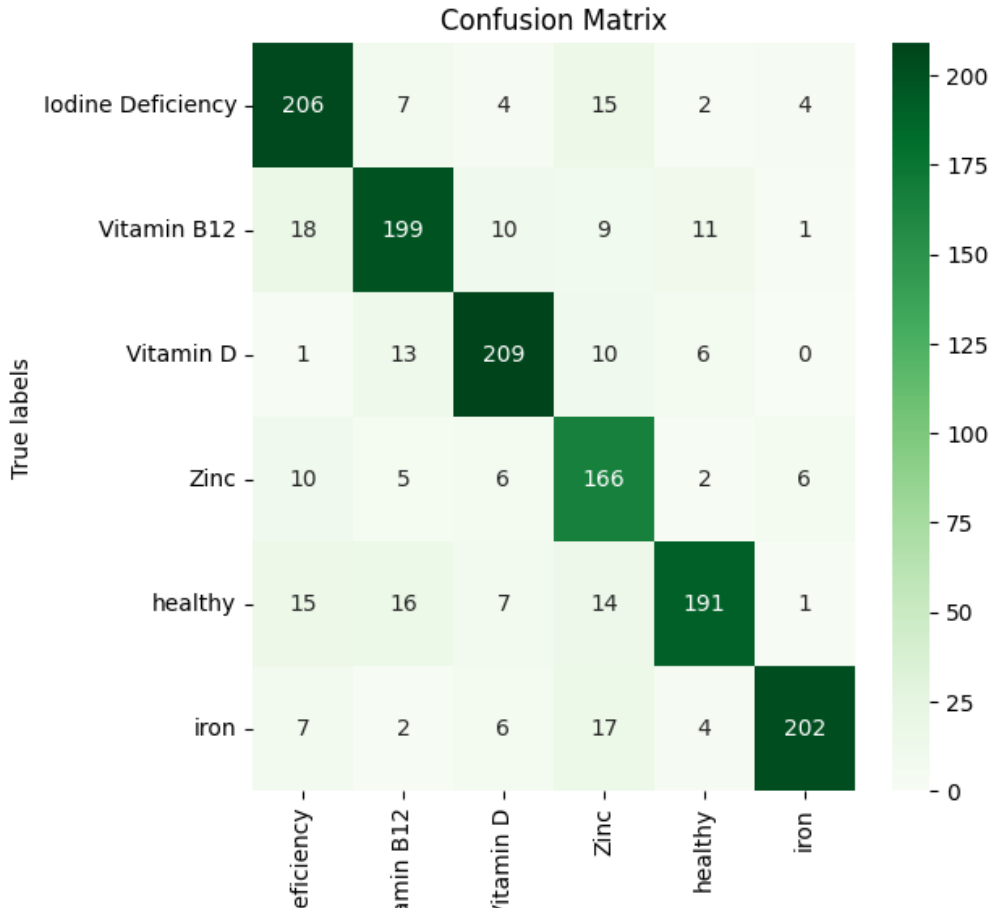

```
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import numpy as np


saved_model = load_model("/content/gdrive/MyDrive/aug7k_split dataset/inceptionv3_aug7k.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 43ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 73ms/step
1/1 [==============================] - 0s 75ms/step
1/1 [==============================] - 0s 71ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 2s 2s/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       0.80      0.87      0.83       238
      Vitamin B12       0.82      0.80      0.81       248
        Vitamin D       0.86      0.87      0.87       239
             Zinc       0.72      0.85      0.78       195
          healthy       0.88      0.78      0.83       244
             iron       0.94      0.85      0.89       238

         accuracy                           0.84      1402
        macro avg       0.84      0.84      0.84      1402
     weighted avg       0.84      0.84      0.84      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/inceptionv3_aug7k.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 [==============================] - 9s 158ms/step - loss: 0.4932 - accuracy: 0.8367
175/175 [==============================] - 135s 773ms/step - loss: 0.2211 - accuracy: 0.9246
Test Accuracy: 83.67%
Train Accuracy: 92.46%
```

✓ 2m 35s completed at 04:08