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

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdriv
```

```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
from tensorflow import keras
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
```



```
# 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/trai
                                            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 = (224, 224),
                                            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}
```

```
resnet_model = Sequential()

pretrained_model= tf.keras.applications.ResNet152V2(include_top=False,
                   input_shape=(224,224,3),
```

```
                    pooling='avg',classes=6,
                    weights='imagenet')
for layer in pretrained_model.layers:
        layer.trainable=False

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet152v2_weig
234545216/234545216 [==============================] - 2s 0us/step
```

```
resnet_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet152v2 (Functional)    (None, 2048)              58331648

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 512)               1049088

 dense_1 (Dense)             (None, 6)                 3078

=================================================================
Total params: 59,383,814
Trainable params: 1,052,166
Non-trainable params: 58,331,648
_________________________________________________________________
```



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

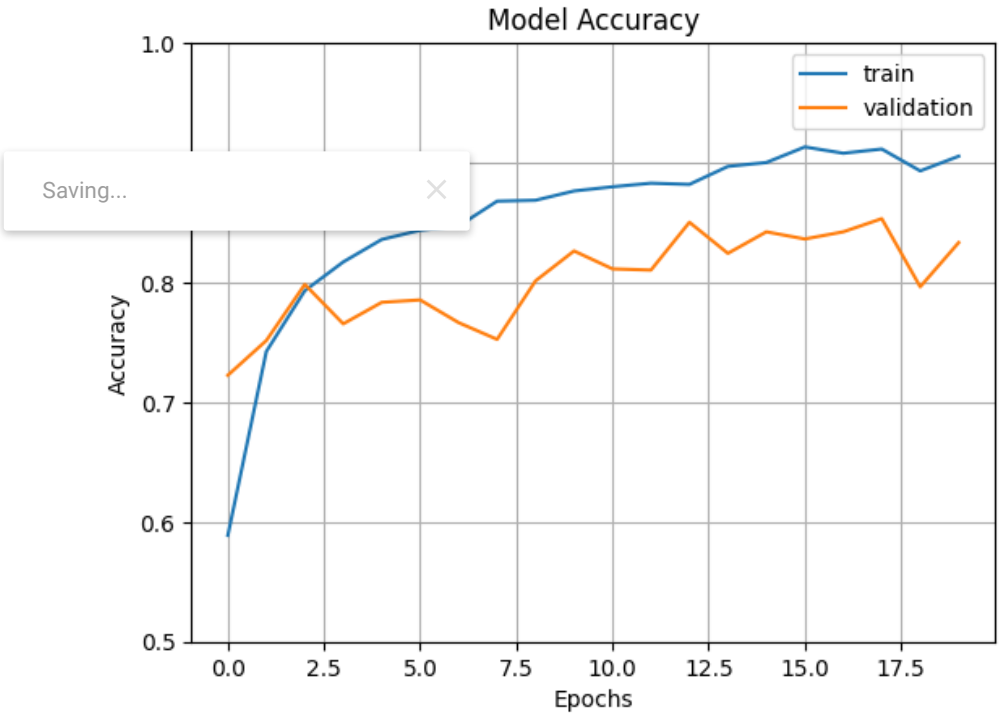
```
epochs=20
history = resnet_model.fit(train_set,validation_data=test_set,epochs=epochs)
```

```
Epoch 1/20
125/125 [==============================] - 1058s 8s/step - loss: 1.8488 - accuracy: 0.5888 - val_loss: 0.7
Epoch 2/20
125/125 [==============================] - 69s 554ms/step - loss: 0.7202 - accuracy: 0.7424 - val_loss: 0.
Epoch 3/20
125/125 [==============================] - 69s 553ms/step - loss: 0.6014 - accuracy: 0.7931 - val_loss: 0.
Epoch 4/20
125/125 [==============================] - 69s 551ms/step - loss: 0.5125 - accuracy: 0.8172 - val_loss: 0.
Epoch 5/20
125/125 [==============================] - 69s 551ms/step - loss: 0.4669 - accuracy: 0.8359 - val_loss: 0.
Epoch 6/20
125/125 [==============================] - 69s 553ms/step - loss: 0.4625 - accuracy: 0.8437 - val_loss: 0.
Epoch 7/20
125/125 [==============================] - 69s 552ms/step - loss: 0.4356 - accuracy: 0.8472 - val_loss: 0.
Epoch 8/20
125/125 [==============================] - 69s 549ms/step - loss: 0.3766 - accuracy: 0.8679 - val_loss: 0.
Epoch 9/20
125/125 [==============================] - 69s 548ms/step - loss: 0.3888 - accuracy: 0.8687 - val_loss: 0.
Epoch 10/20
125/125 [==============================] - 69s 550ms/step - loss: 0.3477 - accuracy: 0.8764 - val_loss: 0.
Epoch 11/20
125/125 [==============================] - 68s 544ms/step - loss: 0.3346 - accuracy: 0.8799 - val_loss: 0.
Epoch 12/20
125/125 [==============================] - 68s 544ms/step - loss: 0.3539 - accuracy: 0.8829 - val_loss: 0.
Epoch 13/20
125/125 [==============================] - 68s 541ms/step - loss: 0.3433 - accuracy: 0.8819 - val_loss: 0.
Epoch 14/20
125/125 [==============================] - 77s 616ms/step - loss: 0.3163 - accuracy: 0.8969 - val_loss: 0.
Epoch 15/20
125/125 [==============================] - 70s 557ms/step - loss: 0.2968 - accuracy: 0.9002 - val_loss: 0.
Epoch 16/20
```

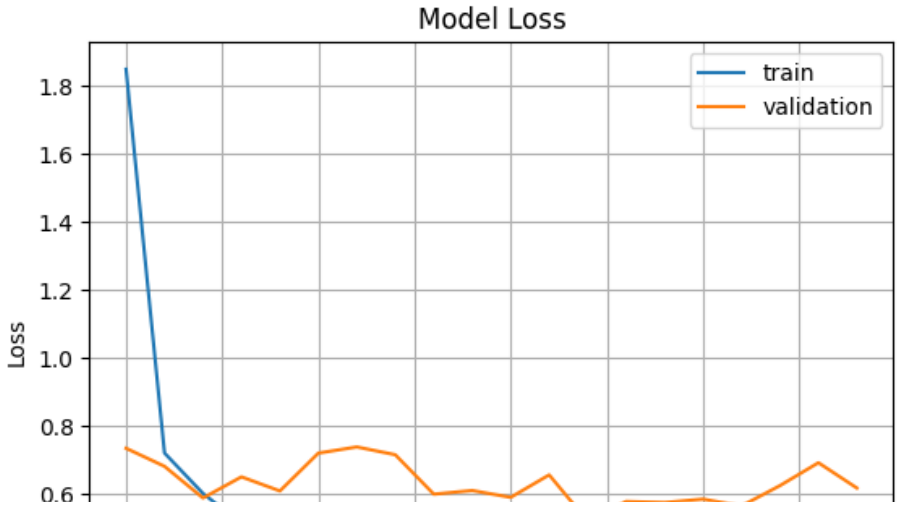
```
125/125 [==============================] - 70s 560ms/step - loss: 0.2625 - accuracy: 0.9132 - val_loss: 0.
Epoch 17/20
125/125 [==============================] - 69s 552ms/step - loss: 0.2738 - accuracy: 0.9080 - val_loss: 0.
Epoch 18/20
125/125 [==============================] - 68s 547ms/step - loss: 0.2484 - accuracy: 0.9115 - val_loss: 0.
Epoch 19/20
125/125 [==============================] - 69s 548ms/step - loss: 0.3231 - accuracy: 0.8932 - val_loss: 0.
Epoch 20/20
125/125 [==============================] - 68s 545ms/step - loss: 0.2877 - accuracy: 0.9055 - val_loss: 0.
```

```
resnet_model.save("/content/gdrive/MyDrive/augmentation 8000/resnet152_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()
```




```
import cv2
image=cv2.imread('/content/drive/MyDrive/Nutrient Deficient RAW Images of Banana Leaves/iron/fe_1.jpg')
image_resized= cv2.resize(image, (224,224))
image=np.expand_dims(image_resized,axis=0)
print(image.shape)
```

```
(1, 224, 224, 3)
```

```
pred=model.predict(image)
print(pred)
```

```
=====] - 2s 2s/step
```

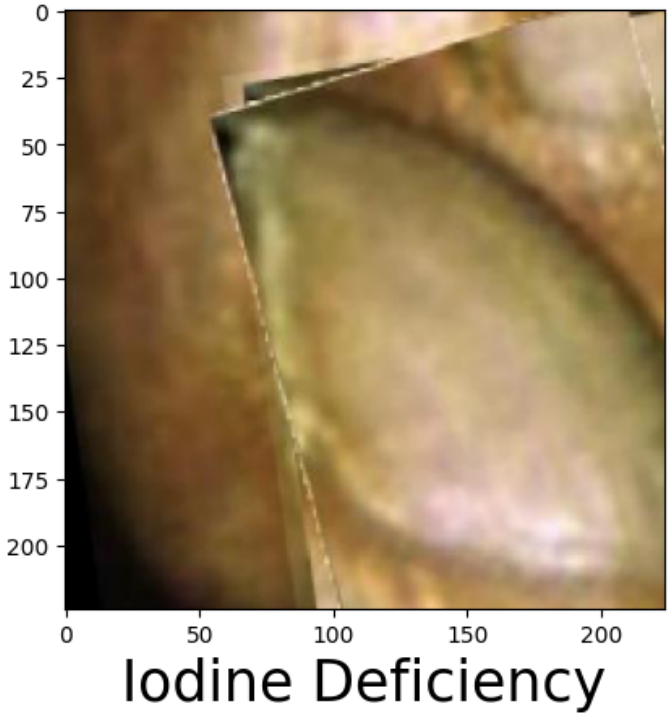


```
from keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model("/content/gdrive/MyDrive/augmentation 8000/resnet152_orig.h5")
```

```
import numpy as np
def predictImage(filename,model):
  img1=image.load_img(filename,target_size=(224,224))
  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=25)
  elif (val==1).all():
    plt.xlabel("Iron Deficiency",fontsize=25)
  elif (val==2).all():
    plt.xlabel("Vitamin - B12  Deficiency",fontsize=25)
  elif (val==3).all():
    plt.xlabel("Vitamin D - Deficiency",fontsize=25)
  elif (val==4).all():
    plt.xlabel("Zinc Deficiency",fontsize=25)
  elif (val==5).all():
    plt.xlabel("healthy",fontsize=25)
```

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

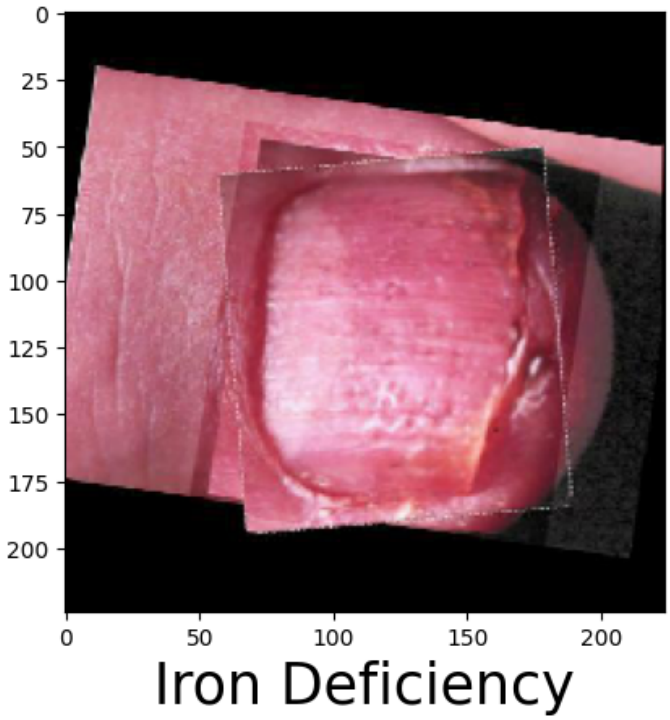
```
1/1 [==============================] - 0s 33ms/step
[[8.3490944e-01 1.0522603e-01 2.4139363e-02 3.5205598e-05 3.1027449e-03
  3.2587245e-02]]
0
```




```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Iron Deficiency/Iron Deficiency_original_11_png.rf.3b282
```

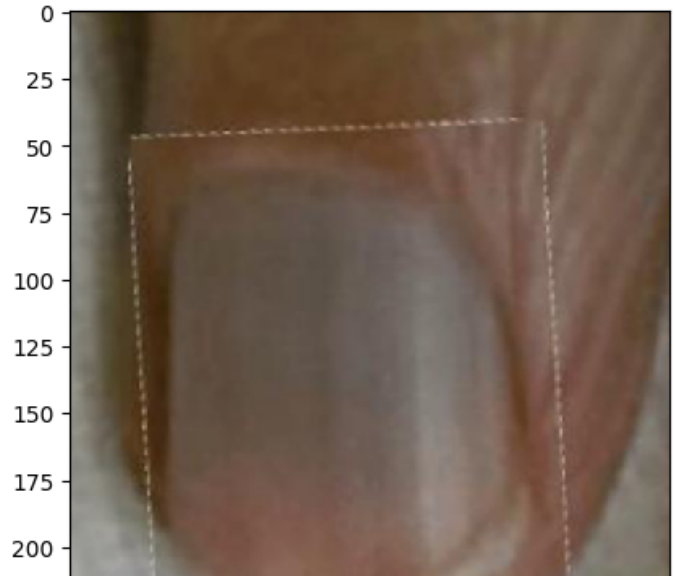
Saving...

```
                                ======] - 0s 99ms/step
[[2.3231039e-09 9.9280655e-01 1.1194920e-11 1.1104450e-08 7.1933996e-03
  1.9598227e-10]]
1
```




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

```
1/1 [==============================] - 0s 35ms/step
[[0.00568694 0.00286138 0.8571615  0.07117879 0.00137404 0.06173729]]
2
```




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

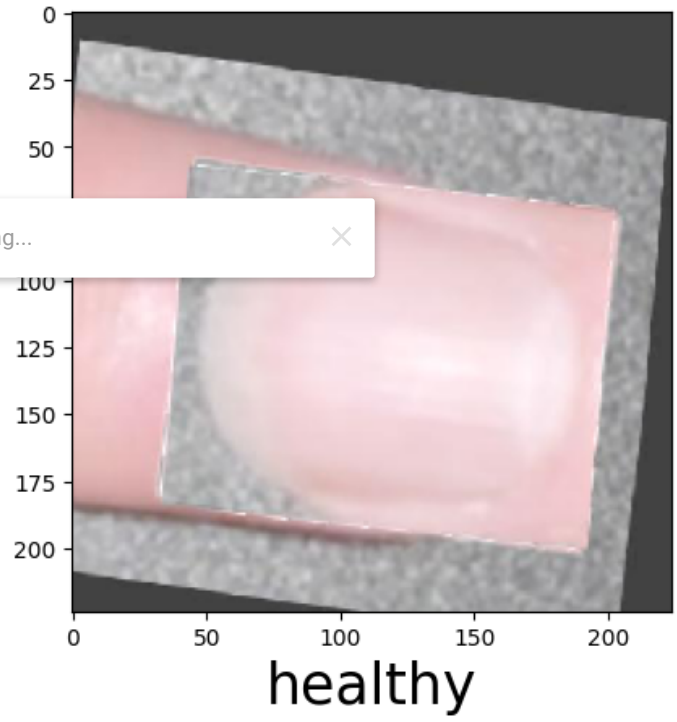
```
1/1 [==============================] - 0s 31ms/step
[[3.4222066e-06 3.4371600e-02 2.1960698e-02 9.2012465e-01 2.2095915e-02
  1.4436342e-03]]
3
```


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

```
1/1 [==============================] - 0s 31ms/step
[[1.1333108e-02 1.0551837e-03 6.5737623e-03 3.6725392e-05 9.8097789e-01
  2.3285716e-05]]
4
```


```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/healthy/healthy_original_Screen-Shot-2021-11-15-at-11-13
```

```
1/1 [==============================] - 0s 34ms/step
[[9.4219380e-05 1.1655289e-01 1.4973156e-02 1.2343110e-02 2.2258284e-04
  8.5581398e-01]]
5
```



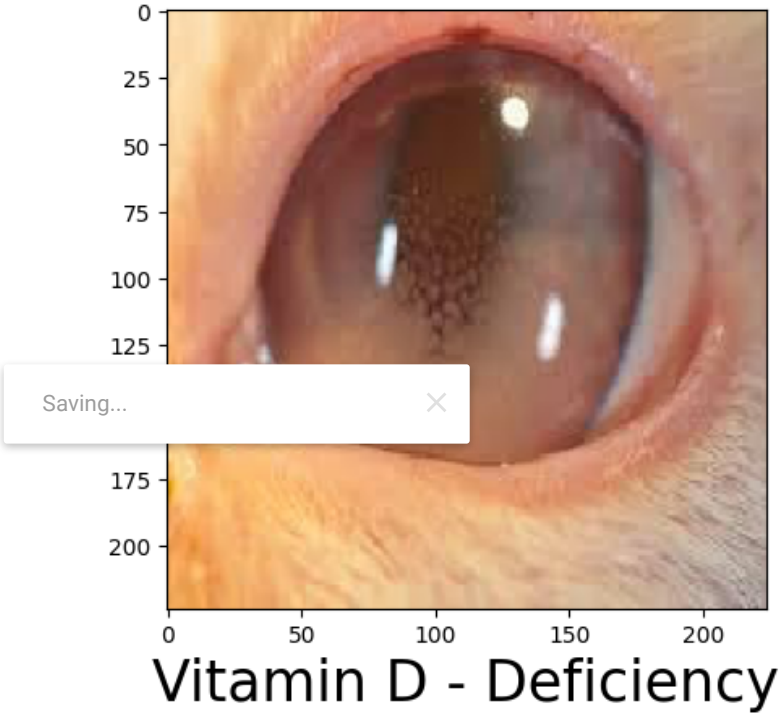
Saving...

```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/healthy/healthy_original_Healthy human eyes_original_ima
```

```
1/1 [==============================] - 0s 94ms/step
[[1.0771215e-13 4.5482822e-11 4.4333467e-15 2.0100113e-09 4.9911244e-17
  1.0000000e+00]]
5
```

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

```
1/1 [==============================] - 0s 182ms/step
[[1.4705297e-06 1.1237891e-05 5.5446083e-07 9.9998653e-01 1.1901161e-08
  7.7563591e-08]]
3
```




```
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/resnet152_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 [==============================] - 3s 3s/step
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 56ms/step
```

```
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 3s 3s/step
```

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

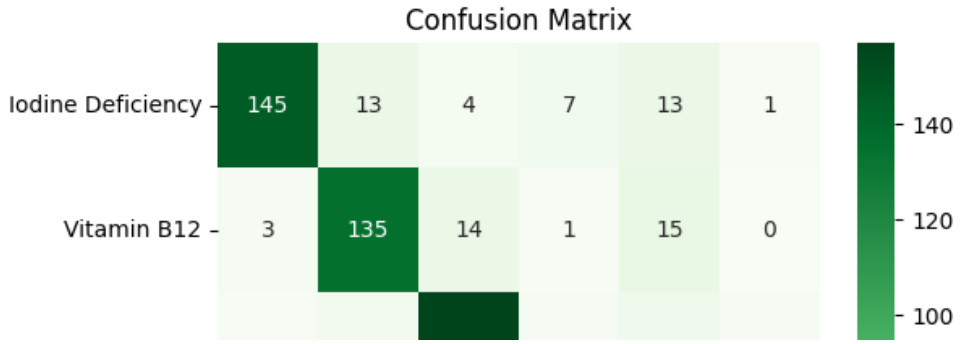


```
cm = confusion_matrix(all_y_true.argmax(axis=1), all_y_pred.argmax(axis=1))

# 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       0.92      0.79      0.85       183
                                     0.80      0.80       168
                                     0.89      0.84       177
                                     0.68      0.76       141
               healthy       0.73      0.91      0.81       162
                  iron       0.93      0.90      0.91       171

              accuracy                           0.83      1002
             macro avg       0.84      0.83      0.83      1002
          weighted avg       0.84      0.83      0.83      1002
```

Saving...

```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Load the saved model
model = load_model('/content/gdrive/MyDrive/augmentation 8000/resnet152_orig.h5')

# Create an image data generator with normalization
test_datagen = ImageDataGenerator(rescale=1./255)

# Load the test data
test_data = test_datagen.flow_from_directory(test_path, target_size=(224, 224), batch_size=32, shuffle=False)

# Use the predict method to obtain predictions on the test data
y_pred = model.predict(test_data)

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

# Get the true class labels
y_true = test_data.classes

# Compute the test accuracy
test_acc = np.mean(y_pred_classes == y_true)

# Use the predict method to obtain predictions on the training data
y_pred_train = model.predict(train_set)
```

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

# Get the true class labels
y_true_train = train_set.classes

# Compute the train accuracy
train_acc = np.mean(y_pred_classes_train == y_true_train)

# Print the train and test accuracy
print('Train accuracy:', train_acc)
print('Test accuracy:', test_acc)
```

```
Found 1002 images belonging to 6 classes.
32/32 [==============================] - 10s 230ms/step
125/125 [==============================] - 66s 526ms/step
Train accuracy: 0.17083541770885444
Test accuracy: 0.8333333333333334
```

```
# 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/resnet152_orig.h5')
scores = model.evaluate(test_set, steps=len(test_set), verbose=1)
scores2 = model.evaluate(train_set, steps=len(test_set), verbose=1)
```



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

```
32/32 [==============================] - 10s 232ms/step - loss: 0.6173 - accuracy: 0.8333
32/32 [==============================] - 16s 477ms/step - loss: 0.2514 - accuracy: 0.9229
Test Accuracy: 83.33%
Train Accuracy: 92.29%
```