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

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

```
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/hidden hunger/original dataset'
splitfolders.ratio(input_folder, output= r'/content/gdrive/MyDrive/hidden hunger/split dataset_orig',
                   seed=42, ratio=(.8, .2),
                   group_prefix=None)
```

```
Copying files: 5763 files [02:00, 47.65 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 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/hidden hunger/split dataset_orig/train'
test_path = '/content/gdrive/MyDrive/hidden hunger/split dataset_orig/val'
```

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

```
IMAGE_SIZE = [224,224]
```

```
# 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/hidden hunger/split dataset_orig/train',tar
                                            batch_size=32,
                                            class_mode='categorical')
```

```
Found 4608 images belonging to 6 classes.
```

```
test_set = test_datagen.flow_from_directory('/content/gdrive/MyDrive/hidden hunger/split dataset_orig/val',
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
Found 1155 images belonging to 6 classes.
```

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

```
{'Iodine Deficiency': 0, 'Vitamin - B12  Deficiency': 1, 'Vitamin D deficiency': 2, 'Zinc Deficiency': 3,
```

```
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 [==============================] - 1s 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'])
```

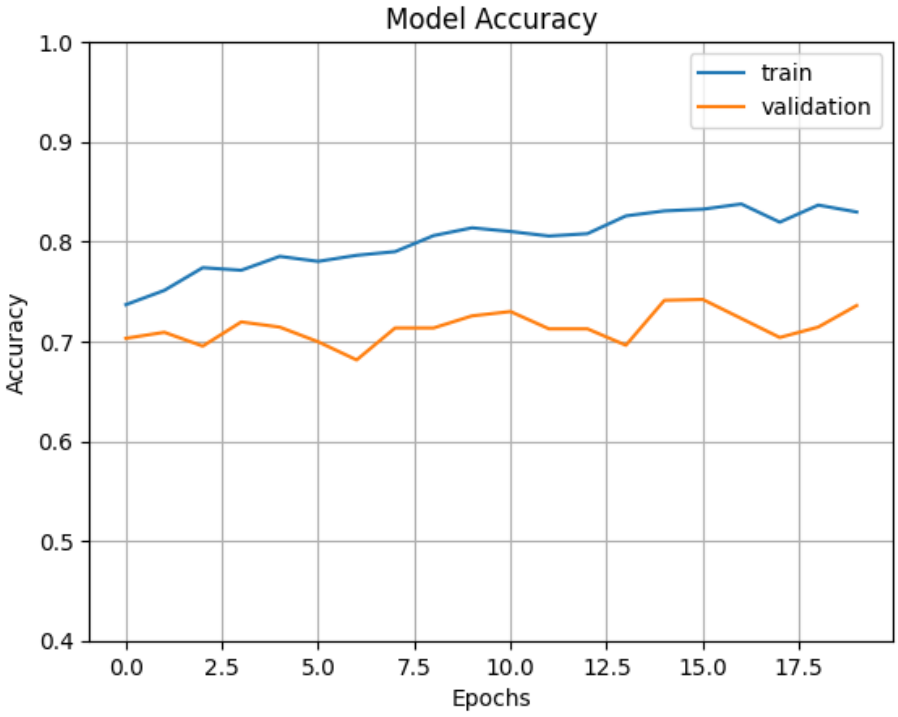
```
epochs=20
history = resnet_model.fit(train_set,validation_data=test_set,epochs=epochs)
```

```
Epoch 1/20
144/144 [==============================] - 85s 590ms/step - loss: 0.7535 - accuracy: 0.7368 - val_loss: 0.
Epoch 2/20
144/144 [==============================] - 84s 583ms/step - loss: 0.7001 - accuracy: 0.7511 - val_loss: 0.
Epoch 3/20
144/144 [==============================] - 86s 598ms/step - loss: 0.6506 - accuracy: 0.7739 - val_loss: 0.
Epoch 4/20
144/144 [==============================] - 84s 584ms/step - loss: 0.6464 - accuracy: 0.7713 - val_loss: 0.
Epoch 5/20
144/144 [==============================] - 84s 586ms/step - loss: 0.6091 - accuracy: 0.7852 - val_loss: 0.
Epoch 6/20
```

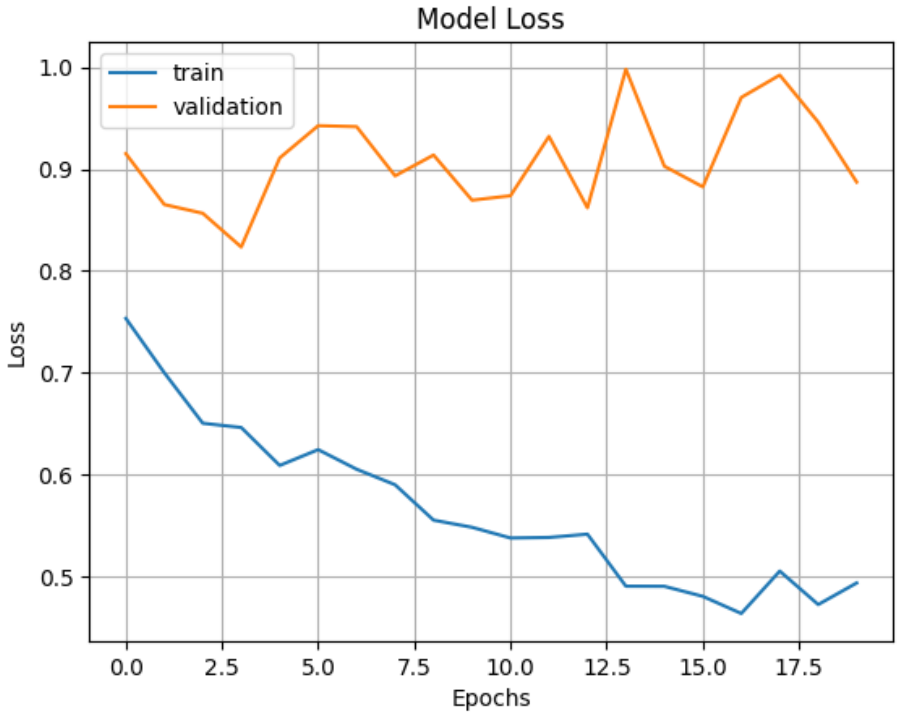
```
144/144 [==============================] - 84s 582ms/step - loss: 0.6247 - accuracy: 0.7802 - val_loss: 0.
Epoch 7/20
144/144 [==============================] - 86s 597ms/step - loss: 0.6054 - accuracy: 0.7862 - val_loss: 0.
Epoch 8/20
144/144 [==============================] - 85s 588ms/step - loss: 0.5902 - accuracy: 0.7899 - val_loss: 0.
Epoch 9/20
144/144 [==============================] - 85s 592ms/step - loss: 0.5554 - accuracy: 0.8060 - val_loss: 0.
Epoch 10/20
144/144 [==============================] - 83s 577ms/step - loss: 0.5484 - accuracy: 0.8138 - val_loss: 0.
Epoch 11/20
144/144 [==============================] - 85s 587ms/step - loss: 0.5379 - accuracy: 0.8101 - val_loss: 0.
Epoch 12/20
144/144 [==============================] - 85s 587ms/step - loss: 0.5385 - accuracy: 0.8056 - val_loss: 0.
Epoch 13/20
144/144 [==============================] - 85s 589ms/step - loss: 0.5417 - accuracy: 0.8079 - val_loss: 0.
Epoch 14/20
144/144 [==============================] - 84s 578ms/step - loss: 0.4906 - accuracy: 0.8257 - val_loss: 0.
Epoch 15/20
144/144 [==============================] - 86s 598ms/step - loss: 0.4905 - accuracy: 0.8307 - val_loss: 0.
Epoch 16/20
144/144 [==============================] - 85s 589ms/step - loss: 0.4806 - accuracy: 0.8325 - val_loss: 0.
Epoch 17/20
144/144 [==============================] - 85s 591ms/step - loss: 0.4638 - accuracy: 0.8377 - val_loss: 0.
Epoch 18/20
144/144 [==============================] - 85s 590ms/step - loss: 0.5054 - accuracy: 0.8194 - val_loss: 0.
Epoch 19/20
144/144 [==============================] - 85s 588ms/step - loss: 0.4726 - accuracy: 0.8366 - val_loss: 0.
Epoch 20/20
144/144 [==============================] - 85s 588ms/step - loss: 0.4938 - accuracy: 0.8296 - val_loss: 0.
```

```
resnet_model.save("/content/gdrive/MyDrive/hidden hunger/split dataset_orig/resnet152_orig.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()
```




```
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)
```

```
1/1 [==============================] - 2s 2s/step
[[1. 0. 0. 0. 0.]]
```

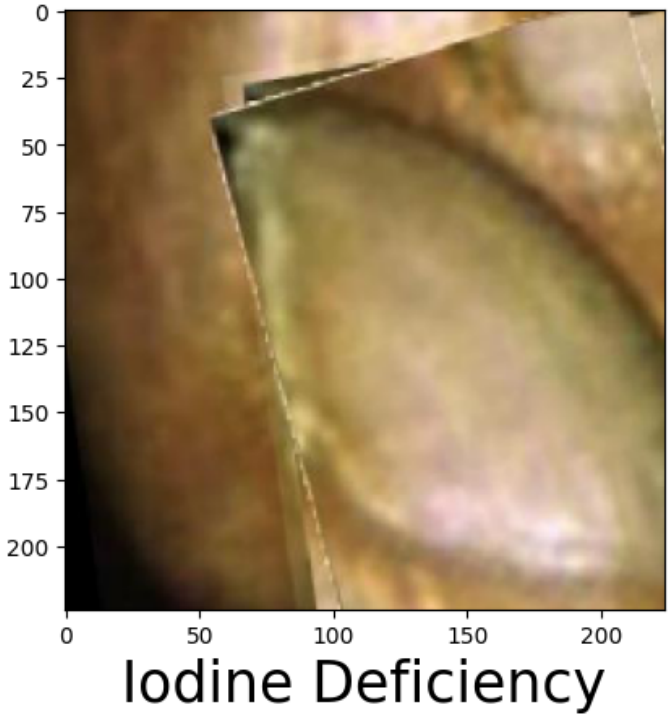
```
from keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model("/content/gdrive/MyDrive/hidden hunger/split dataset_orig/resnet152_orig.h5")
```

```
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
```

```
1/1 [==============================] - 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
```

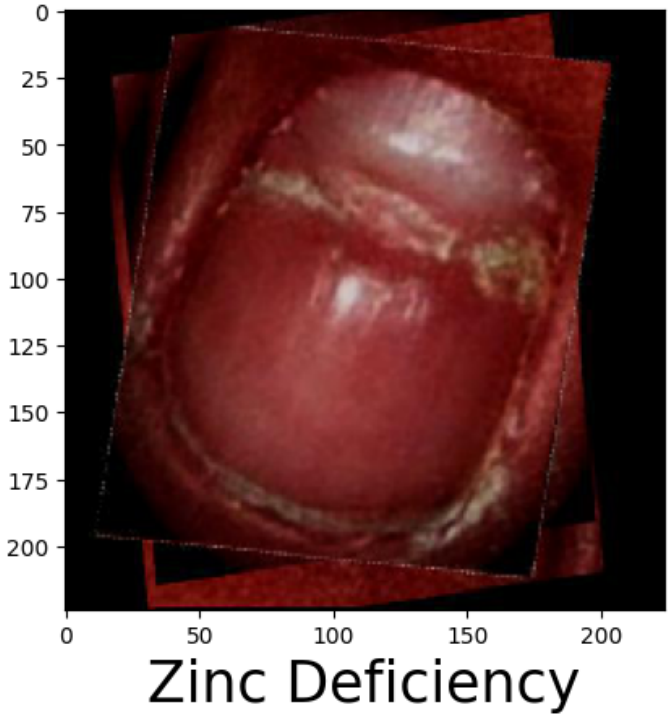
```
1/1 [==============================] - 0s 35ms/step
[[0.00568694 0.00286138 0.8571615  0.07117879 0.00137404 0.06173729]]
2
```


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

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

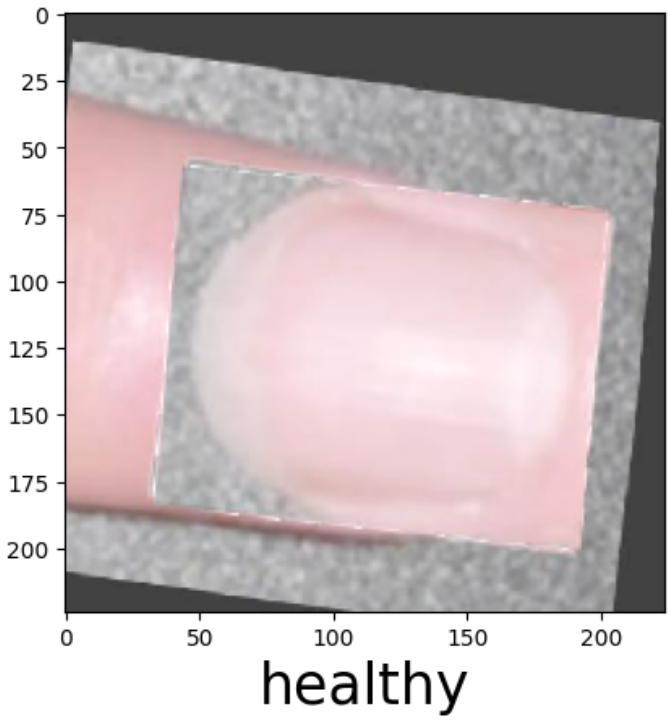
```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Zinc Deficiency/Zinc Deficiency_original_Screen-Shot-202
```

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



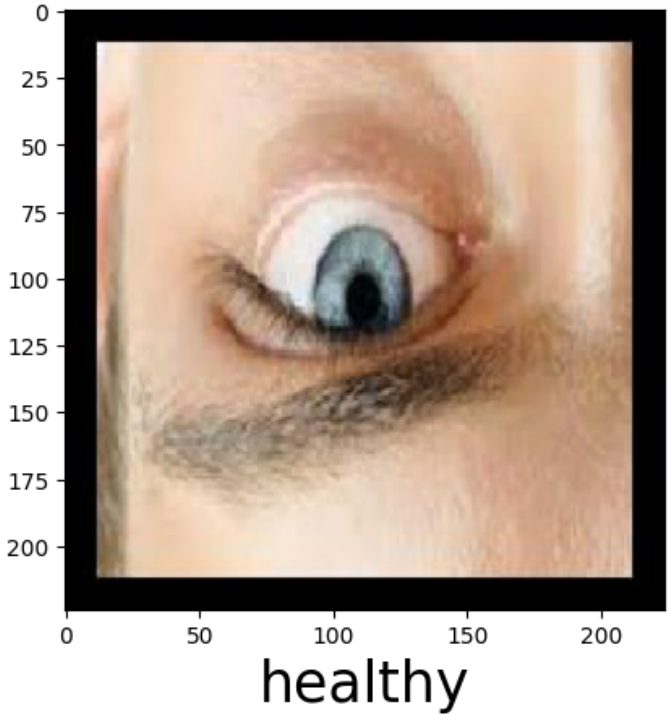

```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/healthy/healthy_original_Screen-Shot-2021-11-15-at-11-13
```

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



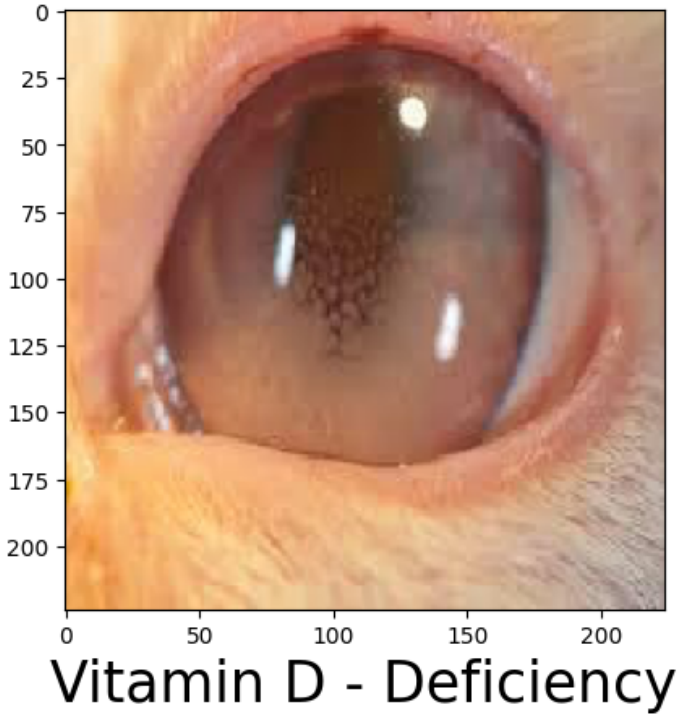
```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/healthy/healthy_original_Healthy human eyes_original_ima
```

```
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_U
```

```
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/hidden hunger/split dataset_orig/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 55ms/step
 1/1 [==============================] - 0s 42ms/step
 1/1 [==============================] - 0s 38ms/step
 1/1 [==============================] - 0s 48ms/step
 1/1 [==============================] - 0s 62ms/step
 1/1 [==============================] - 0s 42ms/step
 1/1 [==============================] - 0s 86ms/step
 1/1 [==============================] - 0s 67ms/step
 1/1 [==============================] - 0s 42ms/step
 1/1 [==============================] - 0s 40ms/step
 1/1 [==============================] - 0s 40ms/step
 1/1 [==============================] - 0s 41ms/step
 1/1 [==============================] - 0s 49ms/step
 1/1 [==============================] - 0s 47ms/step
 1/1 [==============================] - 0s 41ms/step
 1/1 [==============================] - 0s 43ms/step
 1/1 [==============================] - 0s 42ms/step
 1/1 [==============================] - 0s 41ms/step
 1/1 [==============================] - 0s 40ms/step
 1/1 [==============================] - 0s 61ms/step
 1/1 [==============================] - 0s 57ms/step
 1/1 [==============================] - 0s 97ms/step
 1/1 [==============================] - 0s 57ms/step
 1/1 [==============================] - 0s 94ms/step
 1/1 [==============================] - 0s 55ms/step
 1/1 [==============================] - 0s 69ms/step
 1/1 [==============================] - 0s 60ms/step
 1/1 [==============================] - 0s 40ms/step
 1/1 [==============================] - 0s 37ms/step
 1/1 [==============================] - 0s 40ms/step
 1/1 [==============================] - 0s 43ms/step
 1/1 [==============================] - 0s 47ms/step
 1/1 [==============================] - 0s 38ms/step
 1/1 [==============================] - 0s 38ms/step
 1/1 [==============================] - 0s 38ms/step
 1/1 [==============================] - 3s 3s/step
```

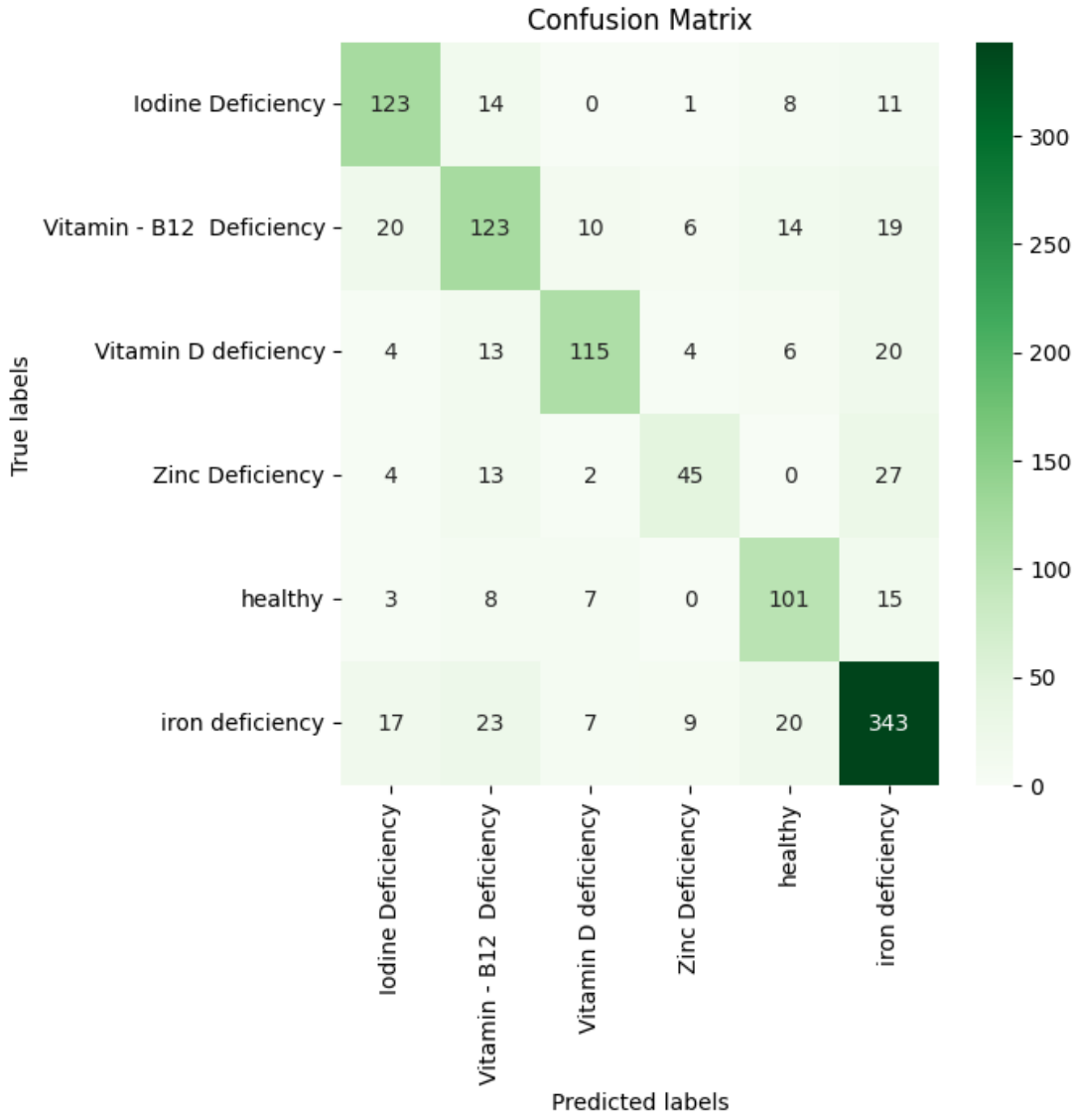
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# 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       0.72      0.78      0.75       157
Vitamin - B12  Deficiency       0.63      0.64      0.64       192
     Vitamin D deficiency       0.82      0.71      0.76       162
          Zinc Deficiency       0.69      0.49      0.58        91
                  healthy       0.68      0.75      0.71       134
          iron deficiency       0.79      0.82      0.80       419

                 accuracy                           0.74      1155
                macro avg       0.72      0.70      0.71      1155
             weighted avg       0.74      0.74      0.73      1155
```

```
import tensorflow as tf
from keras.models import load_model

# Load the saved model
model = load_model('/content/gdrive/MyDrive/hidden hunger/split dataset_orig/resnet152_orig.h5')

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.01),loss='categorical_crossentropy',metrics=['accuracy'])

# Load the training data
# Assuming X_train and y_train are already defined

# Evaluate the model on the training data
train_loss, train_accuracy = model.evaluate(train_set, verbose=0)

# Print the training accuracy
print("Training accuracy:", train_accuracy)
```

```
Training accuracy: 0.8611111044883728
```

✓ 1m 28s completed at 10:18 AM