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

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

```
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/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 = [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/aug7k_split dataset/train',target_size=(224
                                                  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 = (224, 224),
                                            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}
```

```
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 [==============================] - 8s 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=0.001),loss='categorical_crossentropy',metrics=['accuracy'])
```

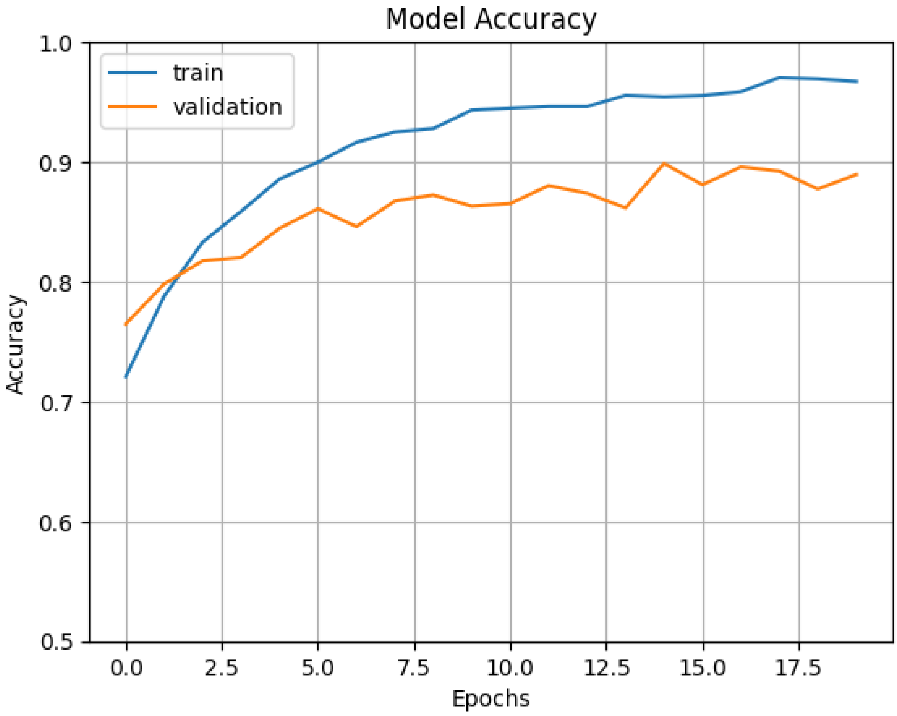
```
epochs=20
history = resnet_model.fit(train_set,validation_data=test_set,epochs=epochs)
```

```
Epoch 1/20
175/175 [==============================] - 3000s 17s/step - loss: 0.7830 - accuracy: 0.7206 - val_loss: 0.
Epoch 2/20
175/175 [==============================] - 100s 570ms/step - loss: 0.5776 - accuracy: 0.7880 - val_loss: 0
Epoch 3/20
175/175 [==============================] - 99s 566ms/step - loss: 0.4601 - accuracy: 0.8330 - val_loss: 0.
Epoch 4/20
175/175 [==============================] - 99s 563ms/step - loss: 0.3929 - accuracy: 0.8587 - val_loss: 0.
Epoch 5/20
175/175 [==============================] - 97s 555ms/step - loss: 0.3316 - accuracy: 0.8857 - val_loss: 0.
Epoch 6/20
175/175 [==============================] - 98s 561ms/step - loss: 0.2874 - accuracy: 0.8998 - val_loss: 0.
Epoch 7/20
175/175 [==============================] - 97s 555ms/step - loss: 0.2484 - accuracy: 0.9164 - val_loss: 0.
Epoch 8/20
175/175 [==============================] - 98s 559ms/step - loss: 0.2153 - accuracy: 0.9250 - val_loss: 0.
Epoch 9/20
175/175 [==============================] - 98s 558ms/step - loss: 0.2045 - accuracy: 0.9278 - val_loss: 0.
Epoch 10/20
175/175 [==============================] - 97s 551ms/step - loss: 0.1652 - accuracy: 0.9432 - val_loss: 0.
Epoch 11/20
175/175 [==============================] - 97s 552ms/step - loss: 0.1591 - accuracy: 0.9448 - val_loss: 0.
Epoch 12/20
175/175 [==============================] - 96s 551ms/step - loss: 0.1544 - accuracy: 0.9462 - val_loss: 0.
Epoch 13/20
175/175 [==============================] - 95s 542ms/step - loss: 0.1513 - accuracy: 0.9462 - val_loss: 0.
Epoch 14/20
175/175 [==============================] - 96s 546ms/step - loss: 0.1275 - accuracy: 0.9555 - val_loss: 0.
Epoch 15/20
175/175 [==============================] - 96s 549ms/step - loss: 0.1340 - accuracy: 0.9543 - val_loss: 0.
Epoch 16/20
175/175 [==============================] - 95s 544ms/step - loss: 0.1199 - accuracy: 0.9553 - val_loss: 0.
```

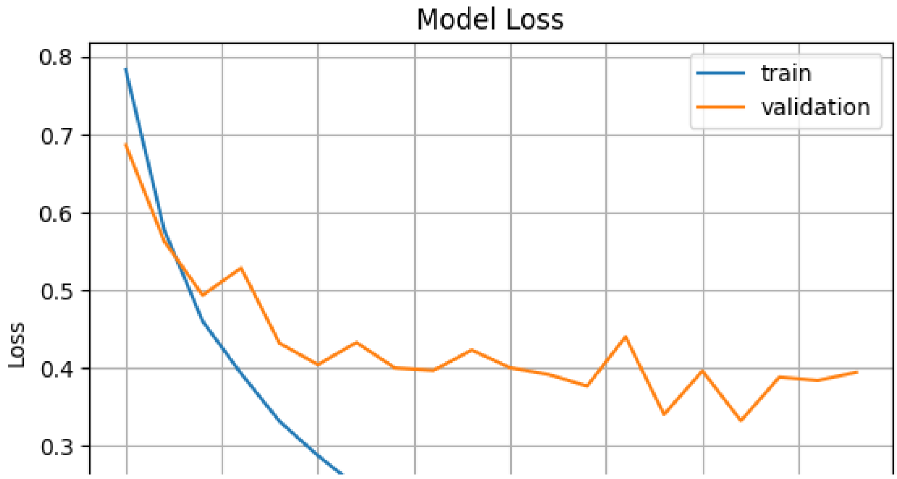
```
Epoch 17/20
175/175 [==============================] - 95s 543ms/step - loss: 0.1243 - accuracy: 0.9586 - val_loss: 0.
Epoch 18/20
175/175 [==============================] - 94s 539ms/step - loss: 0.0847 - accuracy: 0.9703 - val_loss: 0.
Epoch 19/20
175/175 [==============================] - 95s 540ms/step - loss: 0.0928 - accuracy: 0.9693 - val_loss: 0.
Epoch 20/20
175/175 [==============================] - 96s 549ms/step - loss: 0.0941 - accuracy: 0.9671 - val_loss: 0.
```

```
resnet_model.save("/content/gdrive/MyDrive/aug7k_split·dataset/resnet152v2_aug7k.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/gdrive/MyDrive/aug7k_split dataset/resnet152v2_aug7k.h5')
image_resized= cv2.resize(image, (224,224))
image=np.expand_dims(image_resized,axis=0)
print(image.shape)
```

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

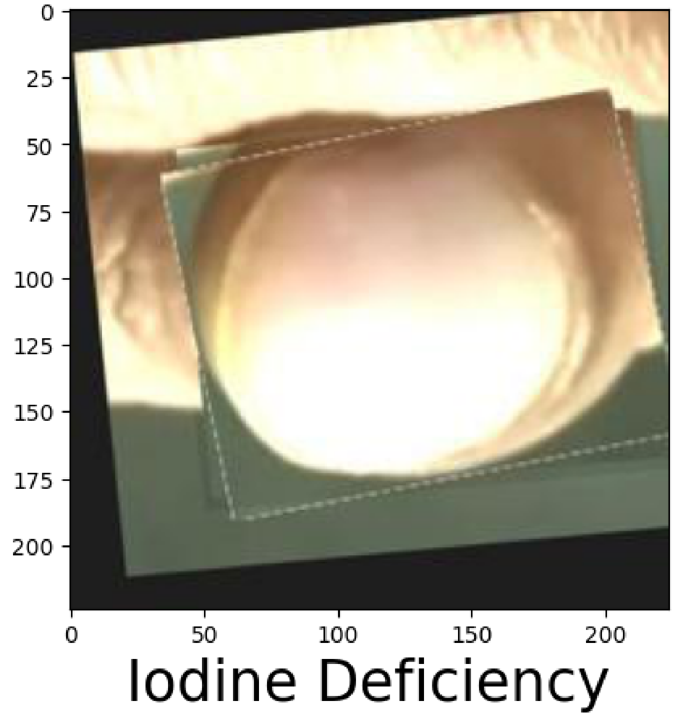
```
pred=model.predict(image)
print(pred)
```

```
    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/aug7k_split dataset/resnet152v2_aug7k.h5")
```

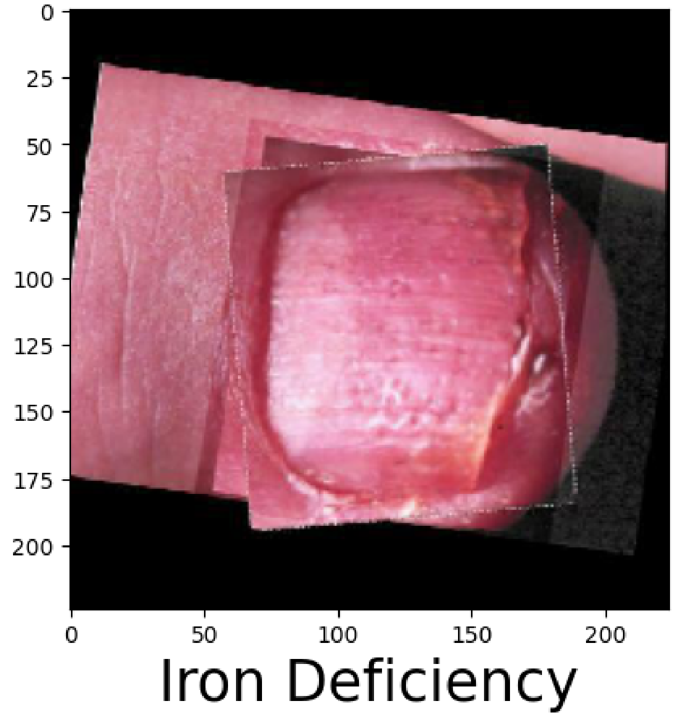
```
import numpy as np
def predictImage(filename,model):
  img1=image.load_img(filename,target_size=(224,224))
  plt.imshow(img1)
  Y=image.img_to_array(img1)
  X=np.expand_dims(Y,axis=0)
  pred=model.predict(X/255)
  print(pred)
  pred = np.array(pred)
  val = np.argmax(pred)
  print(val)
  if (val==0).all():
    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/aug7k_split dataset/val/Iodine Deficiency/Iodine Deficiency_original_Scree
1/1 [==============================] - 6s 6s/step
[[9.6771139e-01 1.1798078e-02 4.2215240e-04 5.8472605e-04 1.9314514e-02
  1.6911938e-04]]
0
```



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

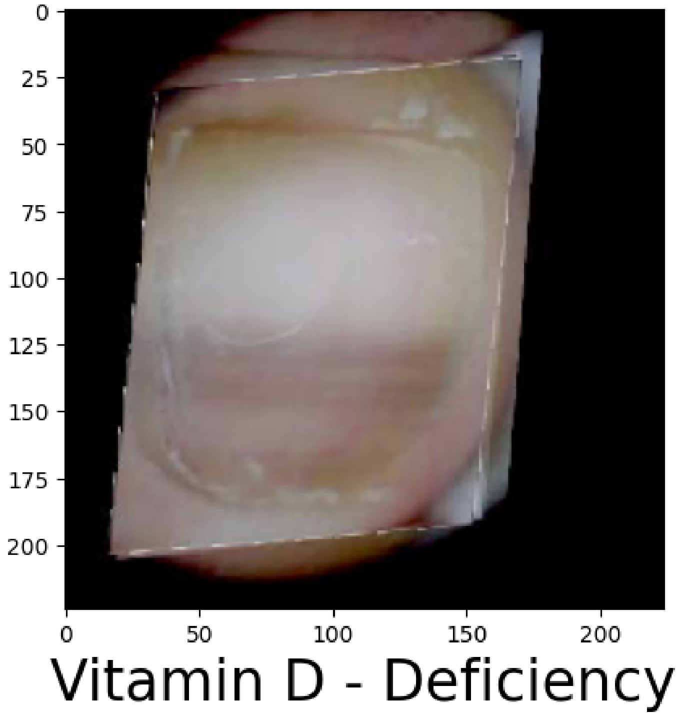
```
1/1 [==============================] - 0s 99ms/step
[[2.3231039e-09 9.9280655e-01 1.1194920e-11 1.1104450e-08 7.1933996e-03
  1.9598227e-10]]
1
```



```
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
```

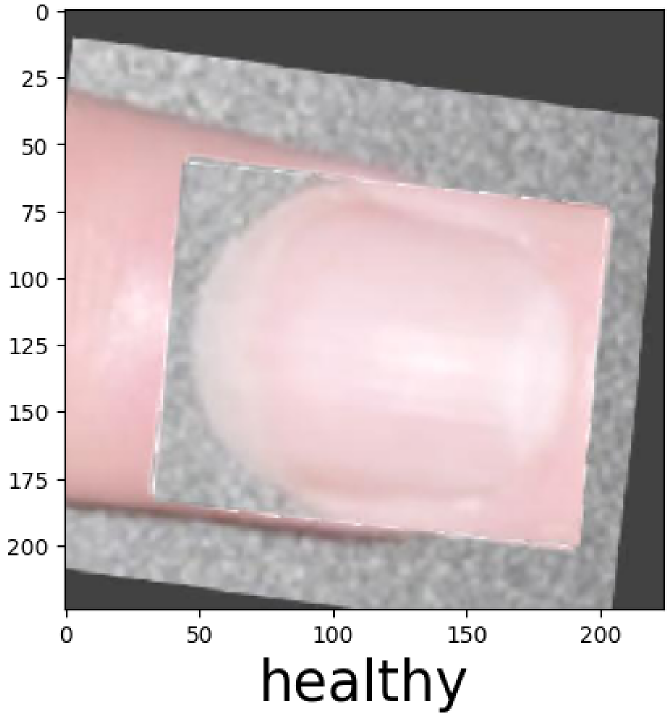
```
1/1 [==============================] - 0s 31ms/step
[[3.4222066e-06 3.4371600e-02 2.1960698e-02 9.2012465e-01 2.2095915e-02
  1.4436342e-03]]
3
```



```
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
```

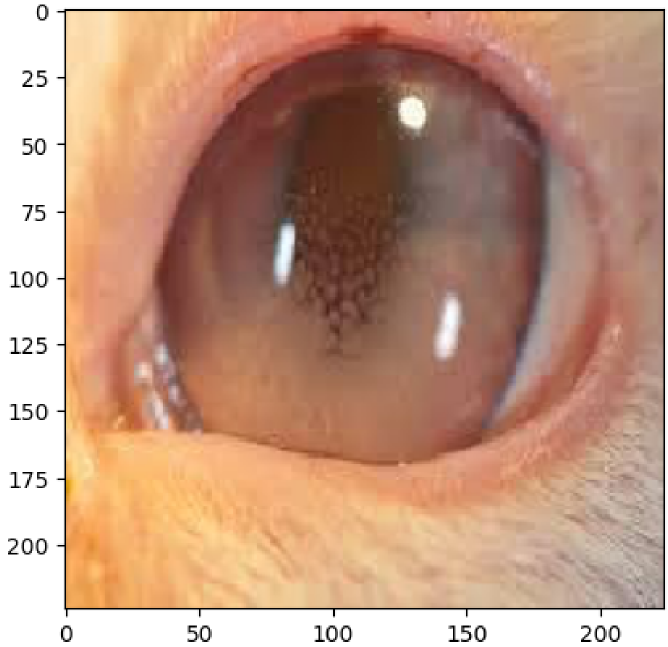
```
1/1 [==============================] - 0s 34ms/step
[[9.4219380e-05 1.1655289e-01 1.4973156e-02 1.2343110e-02 2.2258284e-04
  8.5581398e-01]]
5
```



```
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_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
```



Vitamin D - Deficiency

```
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/resnet152v2_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 [==============================] - 2s 2s/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
```

```
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 75ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 75ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 82ms/step
1/1 [==============================] - 3s 3s/step
```

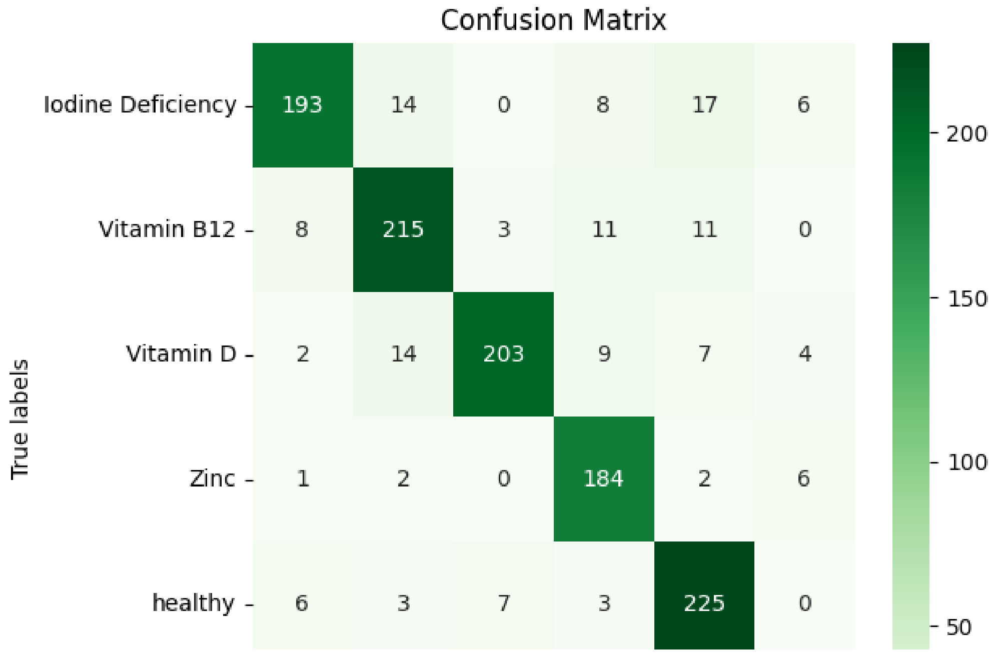
```python
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# 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       0.92      0.81      0.86       238
      Vitamin B12       0.86      0.87      0.86       248
        Vitamin D       0.94      0.85      0.89       239
             Zinc       0.83      0.94      0.88       195
          healthy       0.86      0.92      0.89       244
             iron       0.93      0.95      0.94       238

         accuracy                           0.89      1402
        macro avg       0.89      0.89      0.89      1402
     weighted avg       0.89      0.89      0.89      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/resnet152v2_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 [==============================] - 13s 233ms/step - loss: 0.3944 - accuracy: 0.8894
175/175 [==============================] - 90s 513ms/step - loss: 0.0967 - accuracy: 0.9661
```

```
Test Accuracy: 88.94%
Train Accuracy: 96.61%
```

✓ 2m 1s completed at 05:55 ● ✕