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: 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.
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/augmentation 8000/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)
```

```
magePlugin.JpegImageFile image mode=RGB size=224x224 at 0x7EFC8F44BE80>: 100%|██████████| 7000/7000 [01:31<
```

```
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:  1156
```

```
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/augmentation 8000/resized_Balanced dataset/augment7k'
splitfolders.ratio(input_folder, output= r'/content/gdrive/MyDrive/augmentation 7000/aug7k_split dataset',
                   seed=42, ratio=(.8, .2),
                   group_prefix=None)
```

```
Copying files: 6930 files [01:18, 103.99 files/s]
Copying files: 6944 files [01:18, 88.19 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.mobilenet_v2 import MobileNetV2
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 7000/aug7k_split dataset/train'
test_path = '/content/gdrive/MyDrive/augmentation 7000/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/augmentation 7000/aug7k_split dataset/trair
                                            batch_size=32,
                                            class_mode='categorical')
```

```
Found 5553 images belonging to 6 classes.
```

```
test_set = test_datagen.flow_from_directory('/content/gdrive/MyDrive/augmentation 7000/aug7k_split dataset/val',
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
Found 1391 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}
```

```
mobilenetv2_model = Sequential()
pretrained_model = MobileNetV2(input_shape=(224,224,3), weights='imagenet', include_top=False,
                   pooling='avg',classes=6)
for layer in pretrained_model.layers:
        layer.trainable=False

mobilenetv2_model.add(pretrained_model)
```

```
mobilenetv2_model.add(Flatten())
mobilenetv2_model.add(Dense(512, activation='relu'))
mobilenetv2_model.add(Dense(6, activation='softmax'))
```

```
mobilenetv2_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224 (Funct  (None, 1280)             2257984
 ional)

 flatten_1 (Flatten)         (None, 1280)              0

 dense_2 (Dense)             (None, 512)               655872

 dense_3 (Dense)             (None, 6)                 3078

=================================================================
Total params: 2,916,934
Trainable params: 658,950
Non-trainable params: 2,257,984
_________________________________________________________________
```

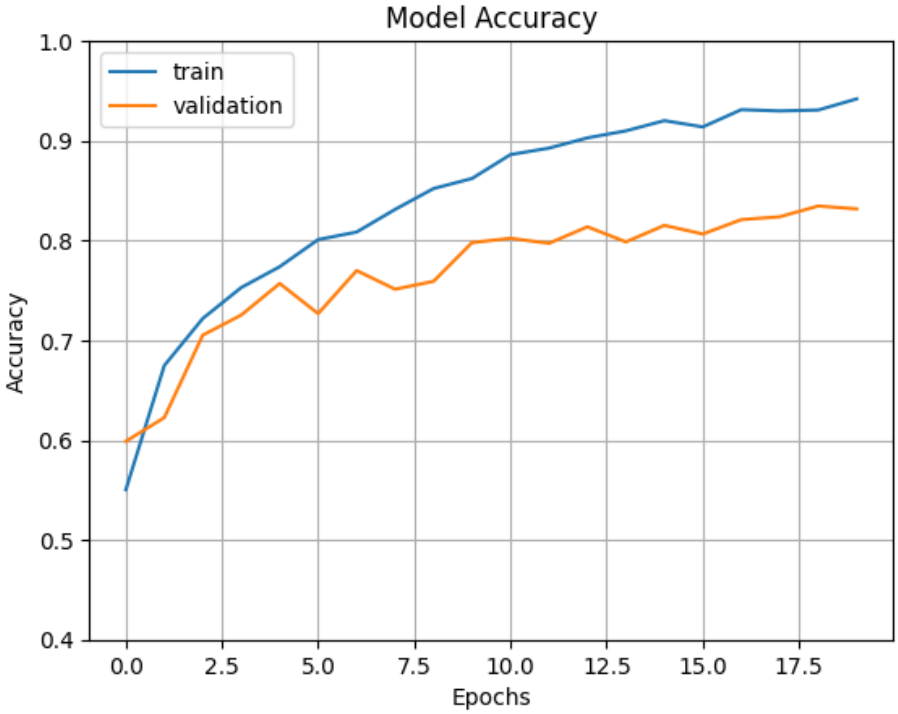
```
mobilenetv2_model.compile(optimizer=Adam(learning_rate=0.001),loss='categorical_crossentropy',metrics=['accuracy
```

```
epochs=20
history = mobilenetv2_model.fit(train_set,validation_data=test_set,epochs=epochs)
```

```
Epoch 1/20
174/174 [==============================] - 119s 601ms/step - loss: 1.1992 - accuracy: 0.5503 - val_loss: 1
Epoch 2/20
174/174 [==============================] - 85s 490ms/step - loss: 0.8683 - accuracy: 0.6748 - val_loss: 0.
Epoch 3/20
174/174 [==============================] - 84s 480ms/step - loss: 0.7512 - accuracy: 0.7220 - val_loss: 0.
Epoch 4/20
174/174 [==============================] - 83s 478ms/step - loss: 0.6737 - accuracy: 0.7529 - val_loss: 0.
Epoch 5/20
174/174 [==============================] - 83s 478ms/step - loss: 0.6069 - accuracy: 0.7738 - val_loss: 0.
Epoch 6/20
174/174 [==============================] - 84s 480ms/step - loss: 0.5482 - accuracy: 0.8010 - val_loss: 0.
Epoch 7/20
174/174 [==============================] - 86s 492ms/step - loss: 0.5208 - accuracy: 0.8086 - val_loss: 0.
Epoch 8/20
174/174 [==============================] - 95s 548ms/step - loss: 0.4605 - accuracy: 0.8313 - val_loss: 0.
Epoch 9/20
174/174 [==============================] - 90s 518ms/step - loss: 0.4144 - accuracy: 0.8522 - val_loss: 0.
Epoch 10/20
174/174 [==============================] - 84s 483ms/step - loss: 0.3731 - accuracy: 0.8622 - val_loss: 0.
Epoch 11/20
174/174 [==============================] - 89s 510ms/step - loss: 0.3271 - accuracy: 0.8862 - val_loss: 0.
Epoch 12/20
174/174 [==============================] - 91s 522ms/step - loss: 0.3037 - accuracy: 0.8927 - val_loss: 0.
Epoch 13/20
174/174 [==============================] - 85s 491ms/step - loss: 0.2772 - accuracy: 0.9029 - val_loss: 0.
Epoch 14/20
174/174 [==============================] - 85s 491ms/step - loss: 0.2647 - accuracy: 0.9100 - val_loss: 0.
Epoch 15/20
174/174 [==============================] - 86s 495ms/step - loss: 0.2354 - accuracy: 0.9202 - val_loss: 0.
Epoch 16/20
174/174 [==============================] - 88s 503ms/step - loss: 0.2369 - accuracy: 0.9139 - val_loss: 0.
Epoch 17/20
174/174 [==============================] - 86s 492ms/step - loss: 0.2027 - accuracy: 0.9312 - val_loss: 0.
Epoch 18/20
174/174 [==============================] - 85s 486ms/step - loss: 0.1996 - accuracy: 0.9299 - val_loss: 0.
Epoch 19/20
174/174 [==============================] - 86s 493ms/step - loss: 0.1994 - accuracy: 0.9308 - val_loss: 0.
Epoch 20/20
174/174 [==============================] - 87s 503ms/step - loss: 0.1761 - accuracy: 0.9420 - val_loss: 0.
```

```
mobilenetv2_model.save("/content/gdrive/MyDrive/augmentation 7000/mobilenetv2.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/augmentation 7000/mobilenetv2.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=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/Zinc/Screen-Shot-2021-11-17-at-2-16-47-PM_png.rf.71503fe69ca9ac47223d646b1
```

```
1/1 [==============================] - 0s 27ms/step
[[3.4713700e-08 3.7590038e-09 3.9117161e-08 9.4012189e-01 9.0897174e-09
  5.9878066e-02]]
3
```

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

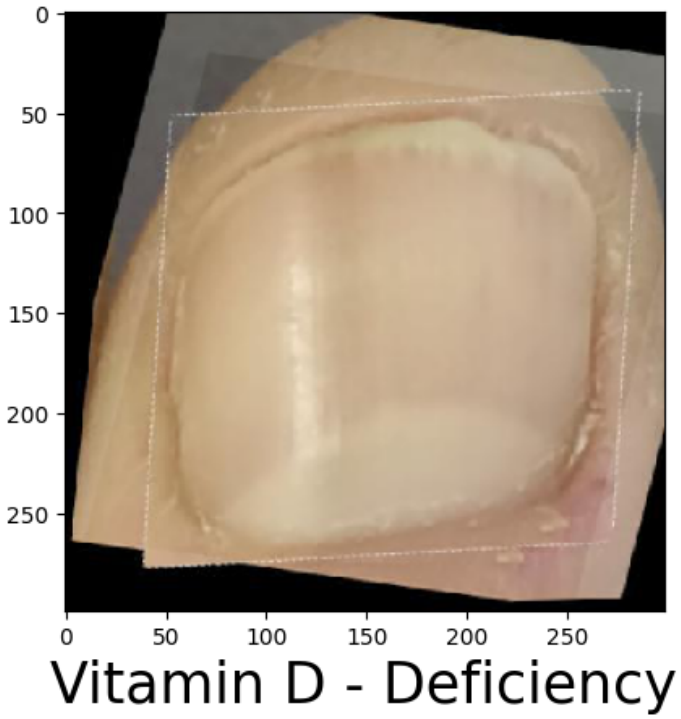
```
1/1 [==============================] - 0s 47ms/step
[[1.2578721e-01 7.1040863e-01 2.5478872e-02 8.6294105e-03 1.2921669e-01
  4.7909268e-04]]
1
```


```
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
```

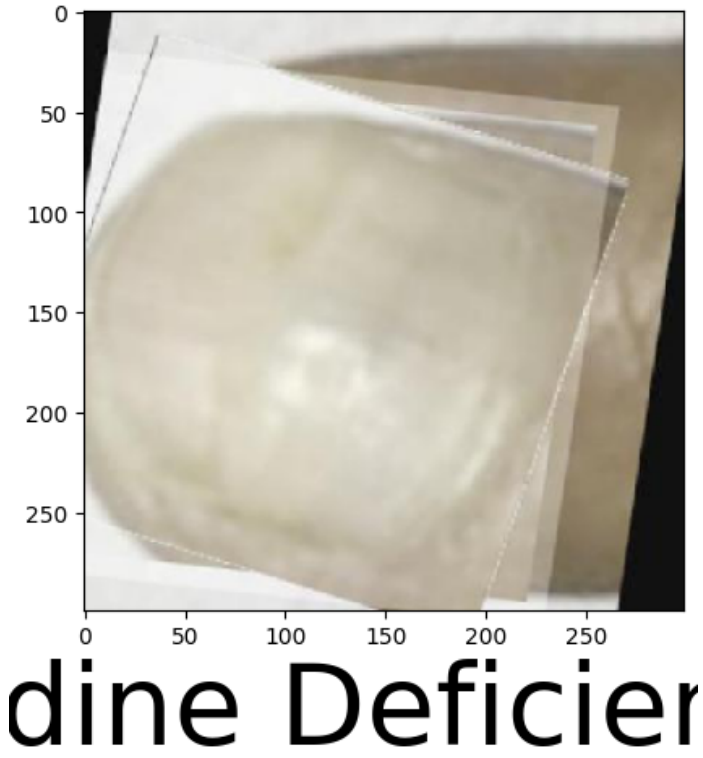
```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Vitamin D - Deficiency/Vitamin D - Deficiency_original_S
```

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



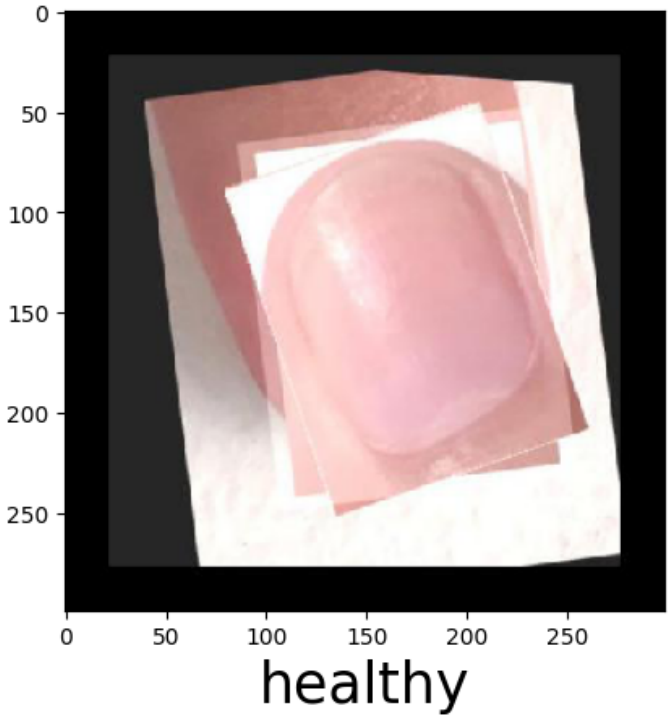

```
predictImage("/content/gdrive/MyDrive/Hidden hunger/val/Zinc Deficiency/Zinc Deficiency_original_Screen-Shot-202
```

```
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/augmentation 7000/mobilenetv2.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 773ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 36ms/step
```

```
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 1s 1s/step
```

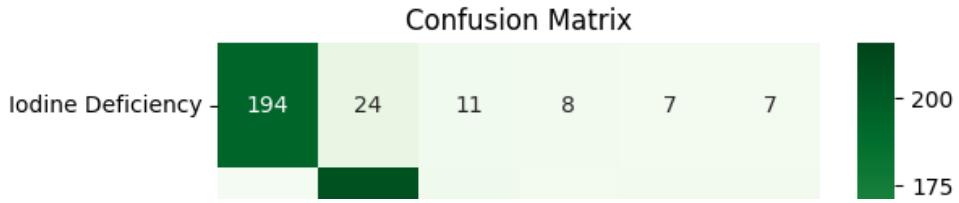
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# 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       0.94      0.77      0.85       251
           Vitamin B12       0.76      0.82      0.79       251
             Vitamin D       0.79      0.79      0.79       239
                  Zinc       0.82      0.76      0.79       189
               healthy       0.87      0.90      0.89       230
                  iron       0.83      0.94      0.88       231

              accuracy                           0.83      1391
             macro avg       0.84      0.83      0.83      1391
          weighted avg       0.84      0.83      0.83      1391
```

```
# 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 7000/mobilenetv2.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))
```

```
Copying files: 1438 files [49:13,  2.05s/ files]
44/44 [==============================] - 6s 109ms/step - loss: 0.5257 - accuracy: 0.8318
174/174 [==============================] - 81s 466ms/step - loss: 0.1367 - accuracy: 0.9552
Test Accuracy: 83.18%
Train Accuracy: 95.52%
```

✓ 1m 32s completed at 2:46 AM