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

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

```
!pip install augmentor
```

```
!pip install augmentor
# Importing necessary library
import Augmentor
# Passing the path of the image directory
p = Augmentor.Pipeline('/content/gdrive/MyDrive/Resized dataset/resized_Balanced dataset')

# Defining augmentation parameters and generating 5 samples
p.zoom(probability = 0.4, min_factor = 0.8, max_factor = 1.5)
p.flip_top_bottom(probability=0.3)
p.rotate(probability=0.3, max_left_rotation=5,max_right_rotation=10)
p.sample(7000)
```

```
pip install split-folders
```

```
import splitfolders

input_folder = r'/content/gdrive/MyDrive/Resized dataset/resized_Balanced dataset/output'
splitfolders.ratio(input_folder, output= r'/content/gdrive/MyDrive/augment 5000/aug5k_split',
                   seed=42, ratio=(.8, .2),
                   group_prefix=None)
```

Convolutional Neural Network

Importing the libraries

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

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.preprocessing.image import ImageDataGenerator
```

```
tf.__version__
```

```
'2.12.0'
```

Double-click (or enter) to edit

Part 1 - Data Preprocessing

Preprocessing the Training set

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
training_set = train_datagen.flow_from_directory('/content/gdrive/MyDrive/aug7k_split dataset/train',
                                                 target_size = (224,224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
Found 5598 images belonging to 6 classes.
```

Preprocessing the Test set

```
test_datagen = ImageDataGenerator(rescale = 1./255)
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.
```

Part 2 - Building the CNN

```
num_classes = 6

model = Sequential([
  layers.Conv2D(16, 3, padding='same',input_shape=(224,224,3), activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(128, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(256, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(256, activation='relu'),
  layers.Dense(num_classes,activation='softmax')
])
```

Compiling the CNN

```
model.compile(optimizer="adam",loss='categorical_crossentropy',metrics=['accuracy'])
```

Training the CNN on the Training set and evaluating it on the Test set

```
history=model.fit(training_set, validation_data = test_set, epochs = 25)
```

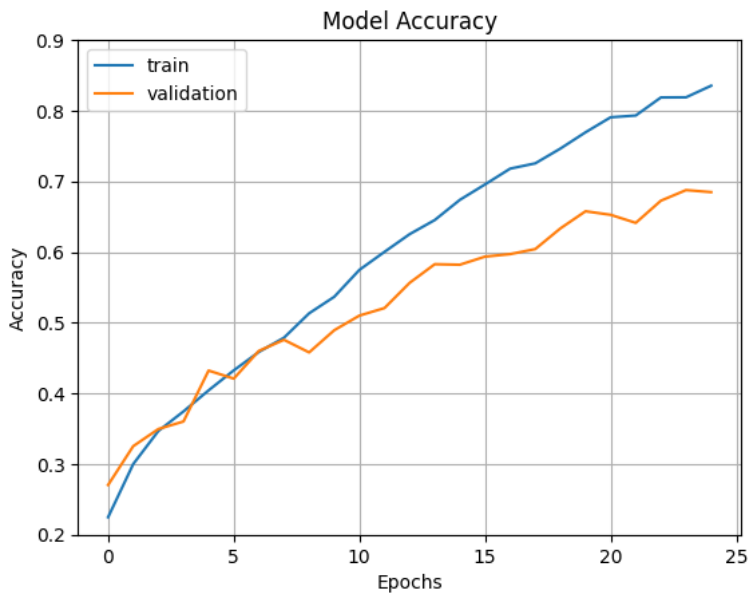
```
Epoch 1/25
175/175 [==============================] - 1313s 7s/step - loss: 1.7620 - accuracy: 0.2247 - val_loss: 1.6955 - val_accuracy: 0.270
Epoch 2/25
175/175 [==============================] - 75s 430ms/step - loss: 1.6730 - accuracy: 0.2999 - val_loss: 1.6481 - val_accuracy: 0.32
Epoch 3/25
175/175 [==============================] - 76s 432ms/step - loss: 1.5894 - accuracy: 0.3467 - val_loss: 1.6797 - val_accuracy: 0.34
Epoch 4/25
175/175 [==============================] - 76s 432ms/step - loss: 1.5454 - accuracy: 0.3744 - val_loss: 1.5718 - val_accuracy: 0.36
Epoch 5/25
175/175 [==============================] - 75s 426ms/step - loss: 1.4866 - accuracy: 0.4043 - val_loss: 1.4606 - val_accuracy: 0.43
Epoch 6/25
175/175 [==============================] - 74s 424ms/step - loss: 1.4367 - accuracy: 0.4325 - val_loss: 1.4542 - val_accuracy: 0.42
Epoch 7/25
175/175 [==============================] - 76s 433ms/step - loss: 1.3690 - accuracy: 0.4586 - val_loss: 1.3799 - val_accuracy: 0.46
Epoch 8/25
175/175 [==============================] - 76s 434ms/step - loss: 1.3182 - accuracy: 0.4787 - val_loss: 1.3466 - val_accuracy: 0.47
Epoch 9/25
175/175 [==============================] - 75s 430ms/step - loss: 1.2573 - accuracy: 0.5132 - val_loss: 1.3572 - val_accuracy: 0.45
Epoch 10/25
175/175 [==============================] - 76s 437ms/step - loss: 1.2042 - accuracy: 0.5366 - val_loss: 1.2854 - val_accuracy: 0.48
Epoch 11/25
175/175 [==============================] - 75s 430ms/step - loss: 1.1374 - accuracy: 0.5747 - val_loss: 1.2757 - val_accuracy: 0.51
Epoch 12/25
175/175 [==============================] - 76s 433ms/step - loss: 1.0722 - accuracy: 0.6002 - val_loss: 1.2424 - val_accuracy: 0.52
Epoch 13/25
175/175 [==============================] - 76s 433ms/step - loss: 1.0064 - accuracy: 0.6252 - val_loss: 1.1966 - val_accuracy: 0.55
Epoch 14/25
175/175 [==============================] - 76s 435ms/step - loss: 0.9538 - accuracy: 0.6452 - val_loss: 1.1835 - val_accuracy: 0.58
Epoch 15/25
175/175 [==============================] - 75s 431ms/step - loss: 0.8774 - accuracy: 0.6738 - val_loss: 1.1662 - val_accuracy: 0.58
Epoch 16/25
175/175 [==============================] - 76s 435ms/step - loss: 0.8305 - accuracy: 0.6956 - val_loss: 1.1172 - val_accuracy: 0.59
Epoch 17/25
175/175 [==============================] - 75s 427ms/step - loss: 0.7870 - accuracy: 0.7179 - val_loss: 1.1705 - val_accuracy: 0.59
Epoch 18/25
175/175 [==============================] - 76s 434ms/step - loss: 0.7548 - accuracy: 0.7254 - val_loss: 1.1932 - val_accuracy: 0.60
Epoch 19/25
```

```
175/175 [==============================] - 76s 436ms/step - loss: 0.6983 - accuracy: 0.7463 - val_loss: 1.1005 - val_accuracy: 0.63
Epoch 20/25
175/175 [==============================] - 75s 430ms/step - loss: 0.6419 - accuracy: 0.7694 - val_loss: 1.0331 - val_accuracy: 0.65
Epoch 21/25
175/175 [==============================] - 75s 430ms/step - loss: 0.6057 - accuracy: 0.7906 - val_loss: 1.0795 - val_accuracy: 0.65
Epoch 22/25
175/175 [==============================] - 75s 430ms/step - loss: 0.5645 - accuracy: 0.7931 - val_loss: 1.1918 - val_accuracy: 0.64
Epoch 23/25
175/175 [==============================] - 76s 437ms/step - loss: 0.5030 - accuracy: 0.8187 - val_loss: 1.1351 - val_accuracy: 0.67
Epoch 24/25
175/175 [==============================] - 77s 440ms/step - loss: 0.5093 - accuracy: 0.8189 - val_loss: 1.0783 - val_accuracy: 0.68
Epoch 25/25
175/175 [==============================] - 77s 439ms/step - loss: 0.4592 - accuracy: 0.8353 - val_loss: 1.0684 - val_accuracy: 0.68
```

```
model.save("/content/gdrive/MyDrive/aug7k_split dataset/Cnnmodel_7k.h5")
```

Part 4 - Evaluationg the accuracy and loss

```
import matplotlib.pyplot as plt
fig1 = plt.gcf()
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.axis(ymin=0.2,ymax=0.9)
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()
```

Part 4 - Making a single prediction

```
from keras.models import load_model
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
model=load_model("/content/gdrive/MyDrive/aug7k_split dataset/Cnnmodel_7k.h5")
```

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

```
{'Iodine Deficiency': 0, 'Vitamin B12': 1, 'Vitamin D': 2, 'Zinc': 3, 'healthy': 4, 'iron': 5}
```

```
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("Vitamin - B12 Deficiency",fontsize=25)
  elif (val==2).all():
    plt.xlabel("Vitamin D  Deficiency",fontsize=25)
  elif (val==3).all():
    plt.xlabel("Zinc Deficiency",fontsize=25)
  elif (val==4).all():
    plt.xlabel("healthy ",fontsize=25)
  elif (val==5).all():
    plt.xlabel("Iron Deficiency",fontsize=25)
```

```
predictImage("/content/gdrive/MyDrive/resized_Balanced dataset/Iodine Deficiency/Screen-Shot-2021-11-22-at-10-01-02-AM_png.rf.be29c3cc64f
```

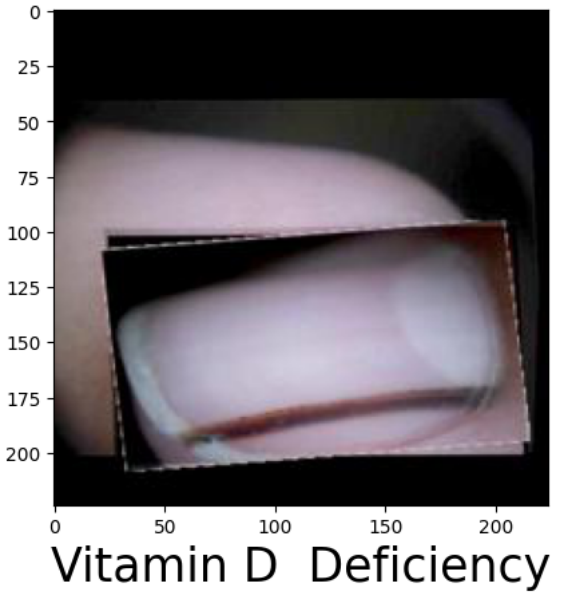
```
1/1 [==============================] - 0s 275ms/step
[[0.6386174  0.17026477 0.00378855 0.00491036 0.13051756 0.05190139]]
```

```
predictImage("/content/gdrive/MyDrive/resized_Balanced dataset/Vitamin B12/Screen-Shot-2021-10-15-at-10-36-15-AM_png.rf.4dc9d4e3de78f1e78
```

```
1/1 [==============================] - 0s 18ms/step
[[3.1655133e-03 9.9589252e-01 6.2716595e-07 4.5251454e-05 8.3436759e-04
  6.1719948e-05]]
1
```


```
predictImage("/content/gdrive/MyDrive/resized_Balanced dataset/Vitamin D/4229e494114681c808066e262f2f82ad1c89676d_jpg.rf.c06c1d1f4666c86b
```

```
1/1 [==============================] - 0s 20ms/step
[[0.00259545 0.01932205 0.9288994  0.02838967 0.00371467 0.01707881]]
2
```

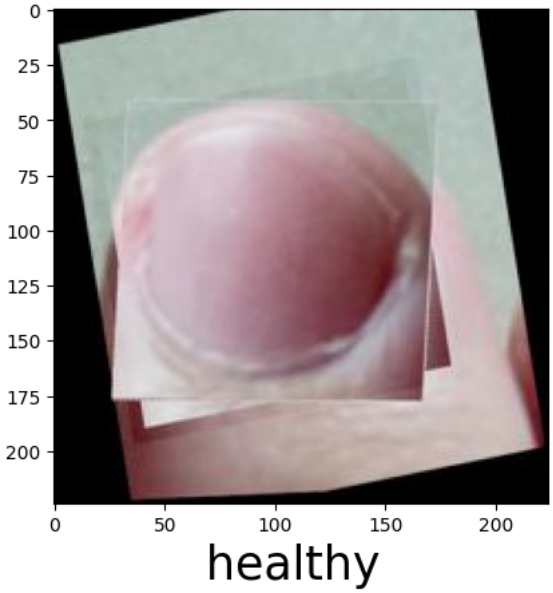



```
predictImage("/content/gdrive/MyDrive/resized_Balanced dataset/Zinc/Habit-tic_deformity_example_on_thumb_jpg.rf.ca86e5dfdaf3be44029f32fa8
```

```
1/1 [==============================] - 0s 29ms/step
[[2.5891416e-11 3.2714887e-08 1.3632850e-03 9.9845231e-01 1.1086853e-04
  7.3528769e-05]]
3
```

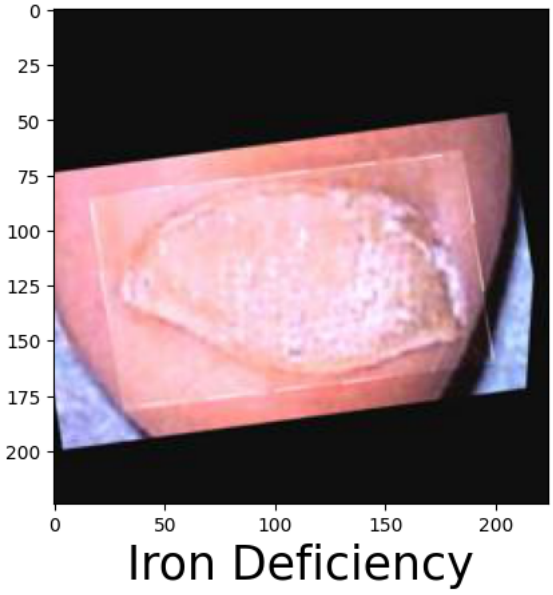
```
predictImage("/content/gdrive/MyDrive/resized_Balanced dataset/healthy/Screen-Shot-2021-11-15-at-2-26-37-PM_png.rf.4b1105baddb4bc4a1a6ff5
```

```
1/1 [==============================] - 0s 18ms/step
[[4.3624036e-02 4.8422298e-05 4.3137744e-02 1.5432524e-05 9.1308212e-01
  9.2247181e-05]]
4
```




```
predictImage("/content/gdrive/MyDrive/resized_Balanced dataset/iron/195_JPG.rf.1f7f796030ff5bb0e454c4938659416e_resized.jpg",model)
```

```
1/1 [==============================] - 0s 24ms/step
[[1.6988021e-06 2.0354685e-09 5.1754117e-03 4.0138650e-05 4.0329686e-09
  9.9478275e-01]]
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/Cnnmodel_7k.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)
```

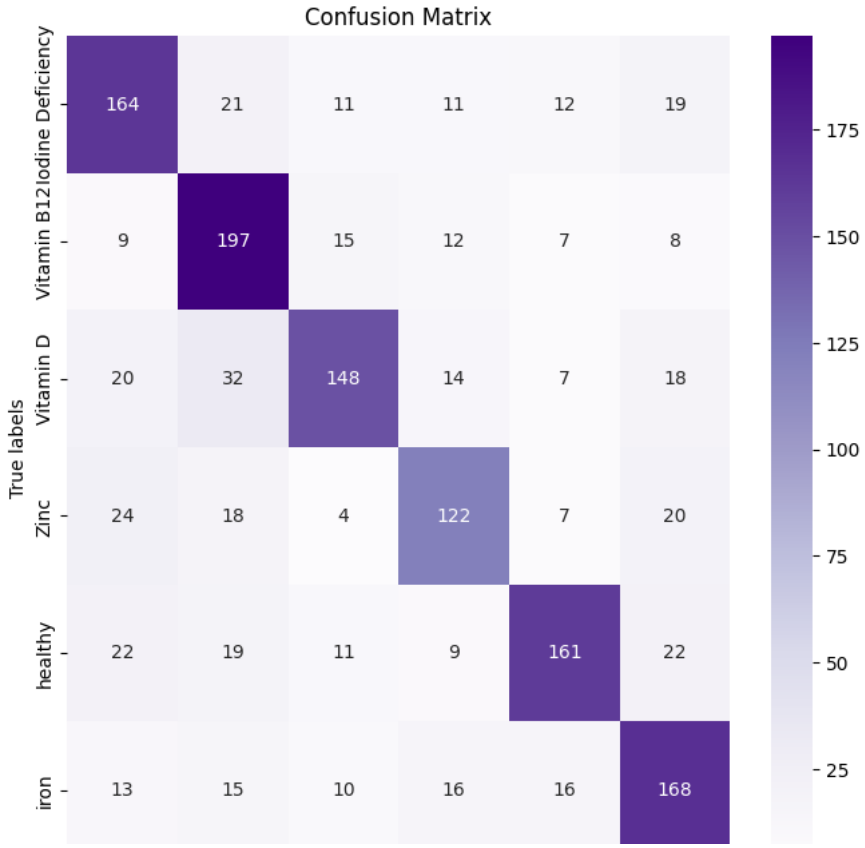
```
 1/1 [==============================] - 0s 105ms/step
 1/1 [==============================] - 0s 25ms/step
 1/1 [==============================] - 0s 27ms/step
 1/1 [==============================] - 0s 27ms/step
 1/1 [==============================] - 0s 34ms/step
 1/1 [==============================] - 0s 31ms/step
 1/1 [==============================] - 0s 26ms/step
 1/1 [==============================] - 0s 32ms/step
 1/1 [==============================] - 0s 27ms/step
 1/1 [==============================] - 0s 25ms/step
 1/1 [==============================] - 0s 27ms/step
 1/1 [==============================] - 0s 29ms/step
 1/1 [==============================] - 0s 28ms/step
 1/1 [==============================] - 0s 25ms/step
 1/1 [==============================] - 0s 29ms/step
 1/1 [==============================] - 0s 26ms/step
 1/1 [==============================] - 0s 28ms/step
 1/1 [==============================] - 0s 28ms/step
 1/1 [==============================] - 0s 26ms/step
 1/1 [==============================] - 0s 26ms/step
 1/1 [==============================] - 0s 26ms/step
 1/1 [==============================] - 0s 25ms/step
 1/1 [==============================] - 0s 28ms/step
 1/1 [==============================] - 0s 32ms/step
 1/1 [==============================] - 0s 26ms/step
 1/1 [==============================] - 0s 27ms/step
 1/1 [==============================] - 0s 25ms/step
 1/1 [==============================] - 0s 26ms/step
 1/1 [==============================] - 0s 28ms/step
 1/1 [==============================] - 0s 40ms/step
 1/1 [==============================] - 0s 40ms/step
 1/1 [==============================] - 0s 32ms/step
 1/1 [==============================] - 0s 36ms/step
 1/1 [==============================] - 0s 36ms/step
 1/1 [==============================] - 0s 36ms/step
 1/1 [==============================] - 0s 37ms/step
 1/1 [==============================] - 0s 35ms/step
 1/1 [==============================] - 0s 34ms/step
 1/1 [==============================] - 0s 34ms/step
 1/1 [==============================] - 0s 38ms/step
 1/1 [==============================] - 0s 41ms/step
 1/1 [==============================] - 0s 38ms/step
 1/1 [==============================] - 0s 38ms/step
 1/1 [==============================] - 0s 99ms/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=(8,8))
sns.heatmap(cm, annot=True, cmap="Purples", fmt="d", xticklabels=training_set.class_indices.keys(),
            yticklabels=training_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=training_set.class_indices.keys())

# Print classification report
print(report)
```

```
                    precision    recall  f1-score   support

Iodine Deficiency       0.65      0.69      0.67       238
      Vitamin B12       0.65      0.79      0.72       248
        Vitamin D       0.74      0.62      0.68       239
             Zinc       0.66      0.63      0.64       195
          healthy       0.77      0.66      0.71       244
             iron       0.66      0.71      0.68       238

         accuracy                           0.68      1402
        macro avg       0.69      0.68      0.68      1402
     weighted avg       0.69      0.68      0.68      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/Cnnmodel_7k.h5')
scores = model.evaluate(test_set, steps=len(test_set), verbose=1)
scores2 = model.evaluate(training_set, steps=len(training_set), verbose=1)


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

```
44/44 [==============================] - 4s 89ms/step - loss: 1.0684 - accuracy: 0.6847
175/175 [==============================] - 70s 401ms/step - loss: 0.4089 - accuracy: 0.8610
Test Accuracy: 68.47%
Train Accuracy: 86.10%
```

✓ 1m 27s completed at 20:30