Convolutional Neural Network

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

```
Mounted at /content/drive
```

Importing the libraries

```
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
```

```
tf.__version__
```

```
'2.12.0'
```

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/drive/MyDrive/hidden hunger/split dataset_orig/train',
                                                 target_size = (224,224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

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

Preprocessing the Test set

```
test_datagen = ImageDataGenerator(rescale = 1./255)
test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/hidden hunger/split dataset_orig/val',
                                            target_size = (224,224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

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

Part 2 - Building the CNN

Initialising the CNN

```
cnn = tf.keras.models.Sequential()
```

Step 1 - Convolution

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=[224,224,3]))
```

Step 2 - Pooling

```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

Adding a second convolutional layer

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

Step 3 - Flattening

```
cnn.add(tf.keras.layers.Flatten())
```

Step 4 - Full Connection

```
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
```

Step 5 - Output Layer

```
cnn.add(tf.keras.layers.Dense(units=6, activation='softmax'))
```

Part 3 - Training the CNN

Compiling the CNN

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

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

```
history=cnn.fit(training_set, validation_data = test_set, epochs = 20)
```

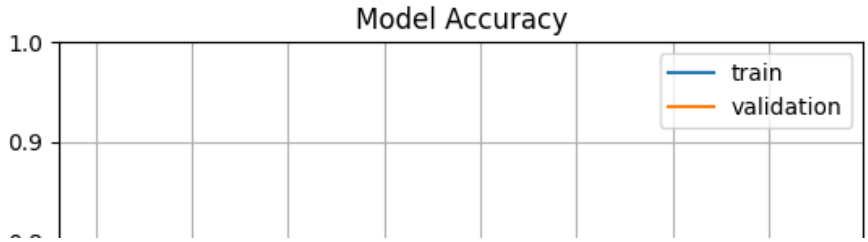
```
Epoch 1/20
144/144 [==============================] - 1910s 13s/step - loss: 1.6752 - accuracy: 0.3596 - val_loss: 1.
Epoch 2/20
144/144 [==============================] - 75s 523ms/step - loss: 1.6396 - accuracy: 0.3626 - val_loss: 1.
Epoch 3/20
144/144 [==============================] - 75s 523ms/step - loss: 1.6063 - accuracy: 0.3815 - val_loss: 1.
Epoch 4/20
144/144 [==============================] - 76s 527ms/step - loss: 1.5669 - accuracy: 0.3928 - val_loss: 1.
```

```
Epoch 5/20
144/144 [==============================] - 77s 535ms/step - loss: 1.5434 - accuracy: 0.3997 - val_loss: 1.
Epoch 6/20
144/144 [==============================] - 75s 522ms/step - loss: 1.5382 - accuracy: 0.4026 - val_loss: 1.
Epoch 7/20
144/144 [==============================] - 75s 519ms/step - loss: 1.5043 - accuracy: 0.4043 - val_loss: 1.
Epoch 8/20
144/144 [==============================] - 75s 520ms/step - loss: 1.4776 - accuracy: 0.4214 - val_loss: 1.
Epoch 9/20
144/144 [==============================] - 80s 558ms/step - loss: 1.4715 - accuracy: 0.4214 - val_loss: 1.
Epoch 10/20
144/144 [==============================] - 75s 521ms/step - loss: 1.4585 - accuracy: 0.4271 - val_loss: 1.
Epoch 11/20
144/144 [==============================] - 76s 526ms/step - loss: 1.4459 - accuracy: 0.4332 - val_loss: 1.
Epoch 12/20
144/144 [==============================] - 76s 528ms/step - loss: 1.4227 - accuracy: 0.4418 - val_loss: 1.
Epoch 13/20
144/144 [==============================] - 75s 520ms/step - loss: 1.4114 - accuracy: 0.4536 - val_loss: 1.
Epoch 14/20
144/144 [==============================] - 74s 517ms/step - loss: 1.3968 - accuracy: 0.4616 - val_loss: 1.
Epoch 15/20
144/144 [==============================] - 75s 521ms/step - loss: 1.3792 - accuracy: 0.4685 - val_loss: 1.
Epoch 16/20
144/144 [==============================] - 75s 522ms/step - loss: 1.3795 - accuracy: 0.4672 - val_loss: 1.
Epoch 17/20
144/144 [==============================] - 79s 548ms/step - loss: 1.3393 - accuracy: 0.4811 - val_loss: 1.
Epoch 18/20
144/144 [==============================] - 79s 547ms/step - loss: 1.3430 - accuracy: 0.4735 - val_loss: 1.
Epoch 19/20
144/144 [==============================] - 75s 523ms/step - loss: 1.3200 - accuracy: 0.4926 - val_loss: 1.
Epoch 20/20
144/144 [==============================] - 73s 507ms/step - loss: 1.3183 - accuracy: 0.4883 - val_loss: 1.
```

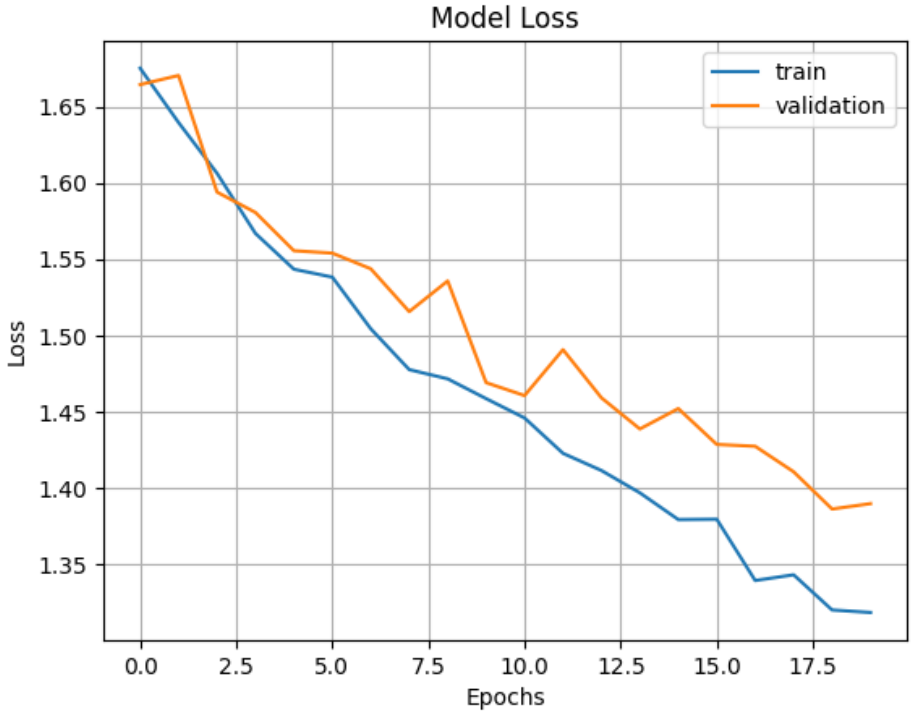
```
cnn.save("/content/drive/MyDrive/hidden hunger/split dataset_orig/Cnn.h5")
```

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=0.4,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()
```



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/drive/MyDrive/hidden hunger/split dataset_orig/Cnn.h5")
```

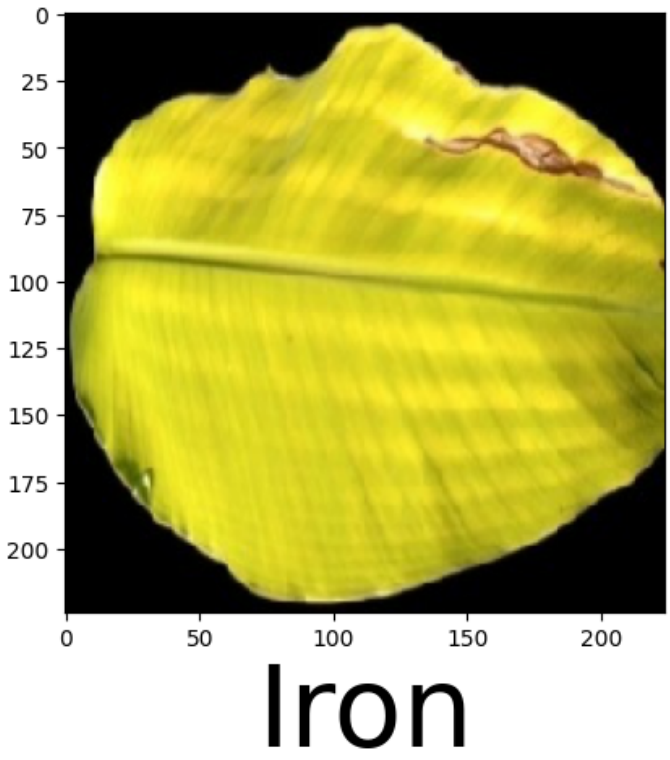
```
import numpy as np
import matplotlib.pyplot as plt

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("Boron",fontsize=50)
  elif (val==1).all():
    plt.xlabel("Healthy",fontsize=50)
  elif (val==2).all():
    plt.xlabel("Iron",fontsize=50)
  elif (val==3).all():
    plt.xlabel("Manganese",fontsize=50)
  elif (val==4).all():
    plt.xlabel("Zinc",fontsize=50)
```

```
predictImage("/content/drive/MyDrive/Nutrient Deficient RAW Images of Banana Leaves/iron/fe_77.jpg",model)
```

```
1/1 [==============================] - 0s 196ms/step
[[2.8564259e-16 4.4928363e-09 9.8998696e-01 5.3775331e-11 1.0013032e-02]]
2
```

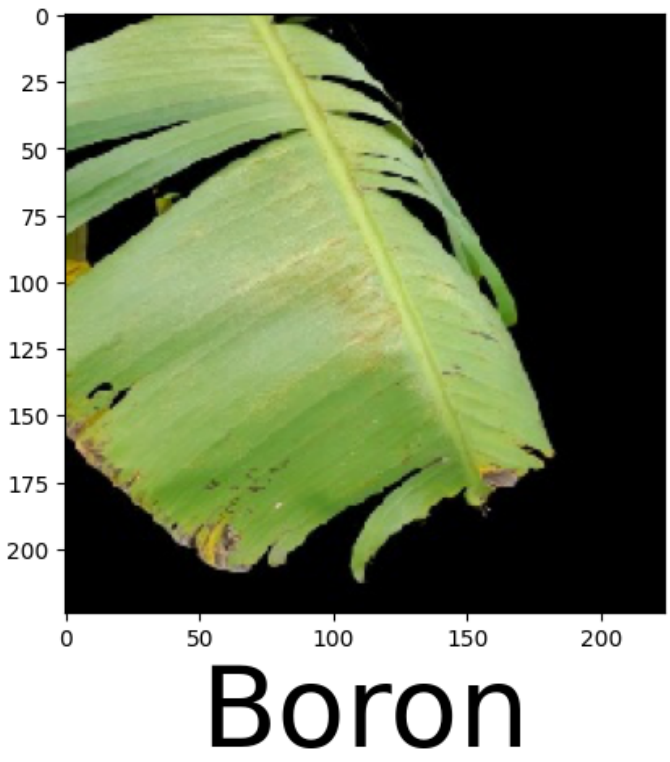



```
predictImage("/content/drive/MyDrive/Nutrient Deficient RAW Images of Banana Leaves/zinc/zn_115.jpg",model)
```

```
1/1 [==============================] - 0s 46ms/step
[[3.1781656e-06 5.2567376e-03 1.0283468e-05 6.3338524e-01 3.6134455e-01]]
3
```


```
predictImage("/content/drive/MyDrive/Nutrient Deficient RAW Images of Banana Leaves/boron/b_92.jpg",model)
```

```
1/1 [==============================] - 0s 48ms/step
[[9.9983943e-01 1.4630052e-04 9.0860640e-06 5.4595485e-08 5.2828177e-06]]
0
```




```
predictImage("/content/drive/MyDrive/th.jpg",model)
```

```
1/1 [==============================] - 0s 60ms/step
[[5.6386393e-02 8.3932799e-01 2.4401537e-08 1.0248621e-01 1.7994266e-03]]
1
```

```
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/drive/MyDrive/hidden hunger/split dataset_orig/Cnn.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 137ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 100ms/step
```

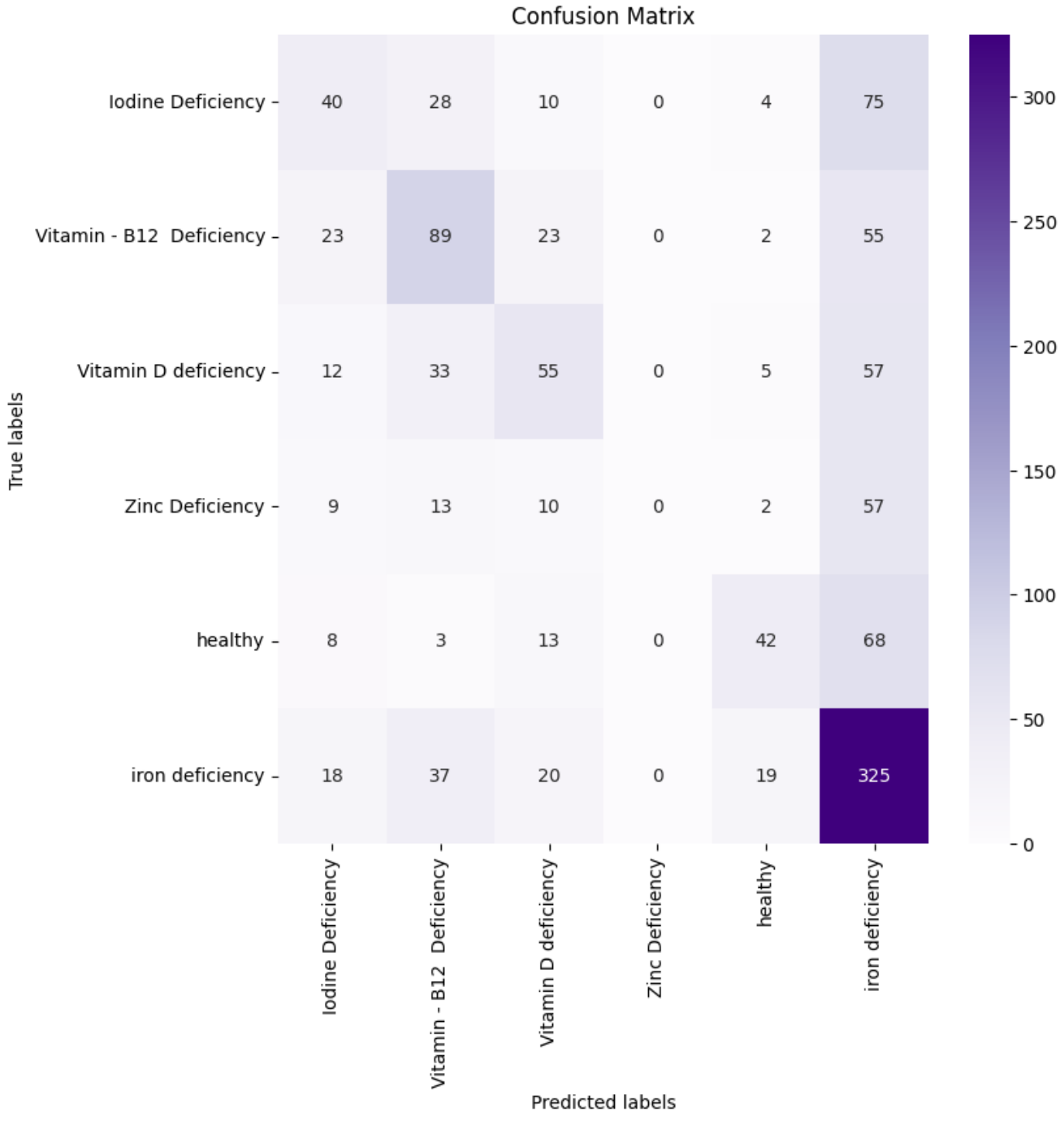
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# compute confusion matrix
cm = confusion_matrix(all_y_true.argmax(axis=1), all_y_pred.argmax(axis=1))

# 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       0.36      0.25      0.30       157
Vitamin - B12  Deficiency       0.44      0.46      0.45       192
     Vitamin D deficiency       0.42      0.34      0.38       162
          Zinc Deficiency       0.00      0.00      0.00        91
                  healthy       0.57      0.31      0.40       134
          iron deficiency       0.51      0.78      0.62       419

                 accuracy                           0.48      1155
                macro avg       0.38      0.36      0.36      1155
             weighted avg       0.43      0.48      0.44      1155

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: P
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: P
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: P
  _warn_prf(average, modifier, msg_start, len(result))
```

```
# Import necessary libraries
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator

# Load the saved model
model = load_model('/content/drive/MyDrive/hidden hunger/split dataset_orig/Cnn.h5')
scores = model.evaluate(test_set, steps=len(test_set), verbose=1)
scores2 = model.evaluate(training_set, steps=len(test_set), verbose=1)


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

```
37/37 [==============================] - 5s 137ms/step - loss: 1.3897 - accuracy: 0.4771
37/37 [==============================] - 18s 499ms/step - loss: 1.2600 - accuracy: 0.5296
Test Accuracy: 47.71%
Train Accuracy: 52.96%
```

✓ 30s completed at 1:27 PM