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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import matplotlib.pyplot as plt
from matplotlib import gridspec
import numpy as np
import pandas as pd
import seaborn as sns
import cv2
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental import preprocessing
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.callbacks import EarlyStopping
from keras.applications.vgg16 import VGG16
plt.rc('figure', autolayout=True)
plt.rc('axes', labelweight='bold', labelsize='large',
       titleweight='bold', titlesize=18, titlepad=10)
plt.rc('image', cmap='magma')
```

```
ds_train = image_dataset_from_directory('/content/drive/MyDrive/chessboard/train',
    labels='inferred',
    label_mode='categorical',
    image_size=[224, 224],
    interpolation='nearest',
    batch_size=32,
    shuffle=True,
)
ds_valid = image_dataset_from_directory(
    '/content/drive/MyDrive/chessboard/valid',
    labels='inferred',
    label_mode='categorical',
    image_size=[224, 224],
    interpolation='nearest',
    batch_size=32,
    shuffle=True,
)
ds_test = image_dataset_from_directory(
    '/content/drive/MyDrive/chessboard/test',
    labels='inferred',
    label_mode='categorical',
    image_size=[224, 224],
    interpolation='nearest',
    batch_size=32,
    shuffle=True,
)

```

```
Found 1936 files belonging to 13 classes.
Found 236 files belonging to 13 classes.
Found 234 files belonging to 13 classes.
```

```
def convert_to_float(image, label):
    image = tf.image.convert_image_dtype(image, dtype=tf.float32)
    return image, label
AUTOTUNE = tf.data.experimental.AUTOTUNE
ds_train = (
    ds_train
    .map(convert_to_float)
    .cache()
    .prefetch(buffer_size=AUTOTUNE)
)
def convert_to_float(image, label):
    image = tf.image.convert_image_dtype(image, dtype=tf.float32)
    return image, label
AUTOTUNE = tf.data.experimental.AUTOTUNE
ds_train = (
    ds_train
```

```
    .map(convert_to_float)
    .cache()
    .prefetch(buffer_size=AUTOTUNE)
)
```

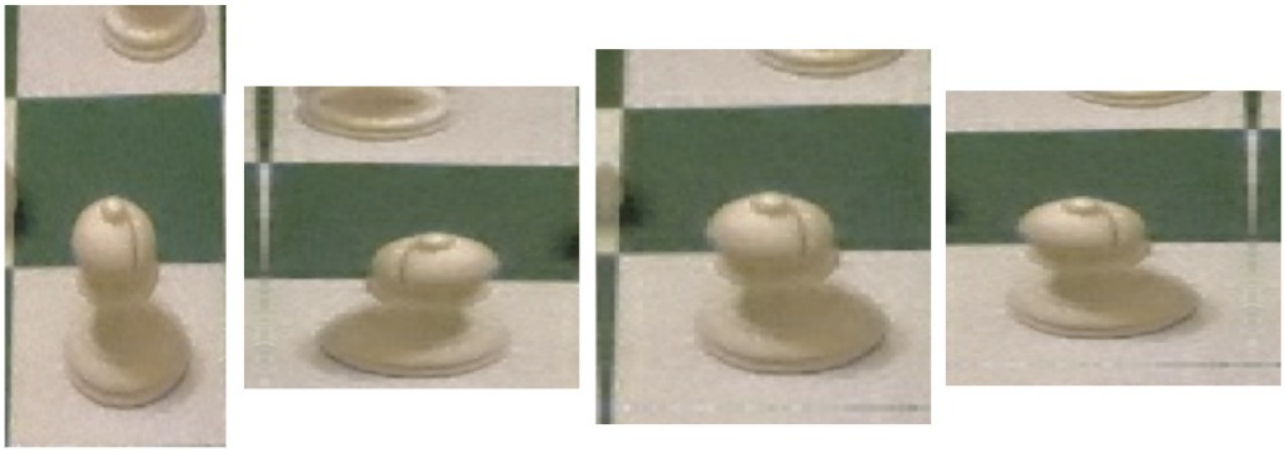
```
from google.colab.patches import cv2_imshow
img = cv2.imread(r'/content/drive/MyDrive/chessboard/data example.jpeg',1)
cv2_imshow(img)
```

```
def get_labels_from_tfdataset(tfdataset, batched=False):
    labels = list(map(lambda x: x[1], tfdataset))
    if not batched:
        return tf.concat(labels, axis=0)
    return labels
array_train = get_labels_from_tfdataset(ds_train)
y_true_train = np.argmax(array_train,axis=1)
unique,counts= np.unique(y_true_train, return_counts = True)
unique = ['Bishop', 'King', "Knight", "Pawn", "Queen", "Rook", "Empty", "Bishop", "King", "Knight", "Pawn", "Queen", "Rook"]
color = ['Black', 'Black', "Black", "Black", "Black", "Black", "None", "White", "White", "White", "White", "White", "White"]
np.asarray((unique,counts)).T
frequency_table_train = pd.DataFrame(data=np.asarray((color,unique,counts)).T, columns=
                                     ["color",'figure','count'])
array_valid = get_labels_from_tfdataset(ds_valid)
y_true_valid = np.argmax(array_valid,axis=1)
unique,counts= np.unique(y_true_valid, return_counts = True)
unique = ['Bishop', 'King', "Knight", "Pawn", "Queen", "Rook", "Empty", "Bishop", "King", "Knight", "Pawn", "Queen", "Rook"]
color = ['Black', 'Black', "Black", "Black", "Black", "Black", "None", "White", "White", "White", "White", "White", "White"]
np.asarray((unique,counts)).T
frequency_table_test = pd.DataFrame(data=np.asarray((color,unique,counts)).T, columns=["color",'figure','count'])
array_test = get_labels_from_tfdataset(ds_test)
y_true_test = np.argmax(array_test,axis=1)
unique,counts= np.unique(y_true_test, return_counts = True)
unique = ['Bishop', 'King', "Knight", "Pawn", "Queen", "Rook", "Empty", "Bishop", "King", "Knight", "Pawn", "Queen", "Rook"]
color = ['Black', 'Black', "Black", "Black", "Black", "Black", "None", "White", "White", "White", "White", "White", "White"]
np.asarray((unique,counts)).T
frequency_table_valid = pd.DataFrame(data=np.asarray((color,unique,counts)).T, columns=["color",'figure','count'])
pd.concat([frequency_table_train, frequency_table_test["count"], frequency_table_valid["count"]], keys=['Train', 'Test', 'Valid'], axis=1)

```

	Train			Test	Valid
	color	figure	count	count	count
0	Black	Bishop	160	20	20
1	Black	King	83	10	10
2	Black	Knight	137	17	17
3	Black	Pawn	156	19	19
4	Black	Queen	149	18	18
5	Black	Rook	160	20	18
6	None	Empty	170	21	21
7	White	Bishop	173	21	21
8	White	King	88	11	11
9	White	Knight	165	20	20
10	White	Pawn	161	19	19
11	White	Queen	160	19	19
12	White	Rook	174	21	21

```
from google.colab.patches import cv2_imshow
img = cv2.imread(r'/content/drive/MyDrive/chessboard/data_augmentation.jpeg',1)
```

```
3 cv2_imshow(img)
```




```
1 pretrained_base = VGG16(weights='imagenet', include_top=False, input_shape=(224,224,3))
2 pretrained_base.summary()
3 pretrained_base.trainable = False
```

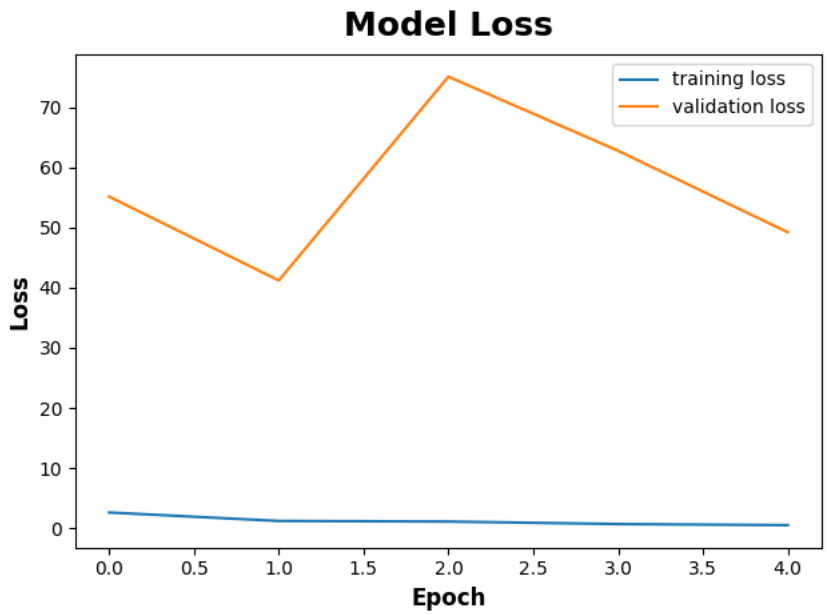
```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

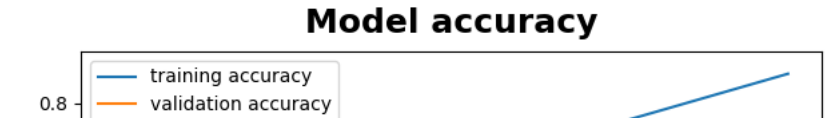
```
model = tf.keras.Sequential([
    preprocessing.RandomContrast(factor=0.5),
    preprocessing.RandomFlip(mode='horizontal'),
    preprocessing.RandomTranslation(height_factor=0.1, width_factor=0.1),
    pretrained_base,
    layers.BatchNormalization(renorm=True),
    layers.Conv2D(filters=256, kernel_size=3, activation='relu', padding='same'),
    layers.Conv2D(filters=256, kernel_size=3, activation='relu', padding='same'),
    tf.keras.layers.GlobalMaxPooling2D(),
    layers.Dropout(0.4),
    layers.BatchNormalization(renorm=True),
    layers.Dense(13, activation='softmax'),
])
early_stopping = EarlyStopping(
    min_delta = 0.001,
    patience = 30,
    restore_best_weights = True,
)
model.compile(
    optimizer= 'adam',
    loss='categorical_crossentropy',
    metrics=['categorical_accuracy'],
)
```

```
history = model.fit(ds_train,
    validation_data= ds_valid,
    epochs= 5,
    callbacks=[early_stopping]
   )
```

```
′5
=============================] - 1444s 24s/step - loss: 2.6144 - categorical_accuracy: 0.2546 - val_loss: 55.1258 - val_categorical_accu
′5
=============================] - 1432s 24s/step - loss: 1.2103 - categorical_accuracy: 0.5749 - val_loss: 41.1871 - val_categorical_accu
′5
=============================] - 1380s 23s/step - loss: 1.1109 - categorical_accuracy: 0.6829 - val_loss: 75.0667 - val_categorical_accu
′5
=============================] - 1444s 24s/step - loss: 0.6968 - categorical_accuracy: 0.7753 - val_loss: 62.7529 - val_categorical_accu
′5
=============================] - 1438s 24s/step - loss: 0.5110 - categorical_accuracy: 0.8383 - val_loss: 49.2059 - val_categorical_accu
```

```
history_frame = pd.DataFrame(history.history)
history_frame.loc[:, ['loss', 'val_loss']].plot()
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['training loss', 'validation loss'])
plt.show()
history_frame.loc[:, ['categorical_accuracy', 'val_categorical_accuracy']].plot();
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['training accuracy', 'validation accuracy'])
plt.show()
```



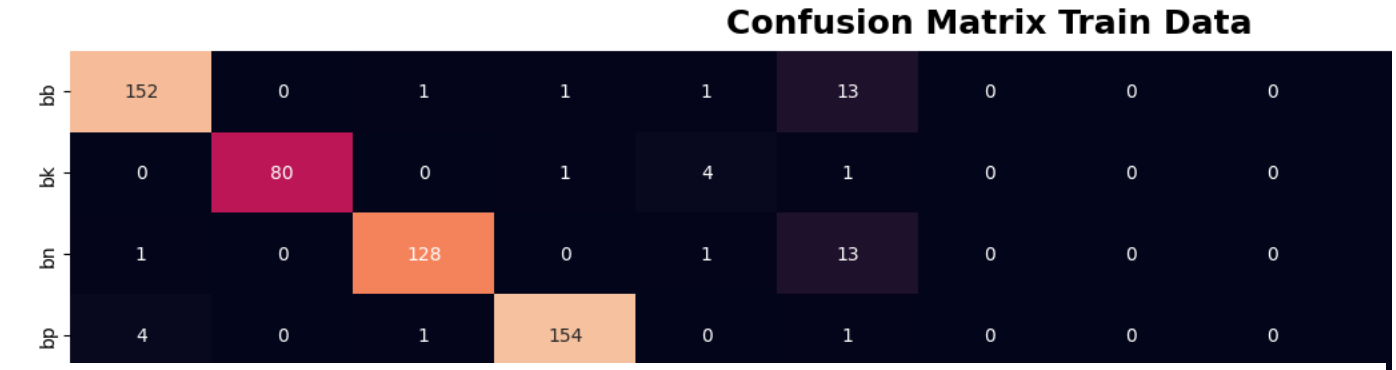



```
1 history_frame[history_frame.val_loss == history_frame.val_loss.min()]
```

	loss	categorical_accuracy	val_loss	val_categorical_accuracy
1	1.210268	0.574897	41.187061	0.40678

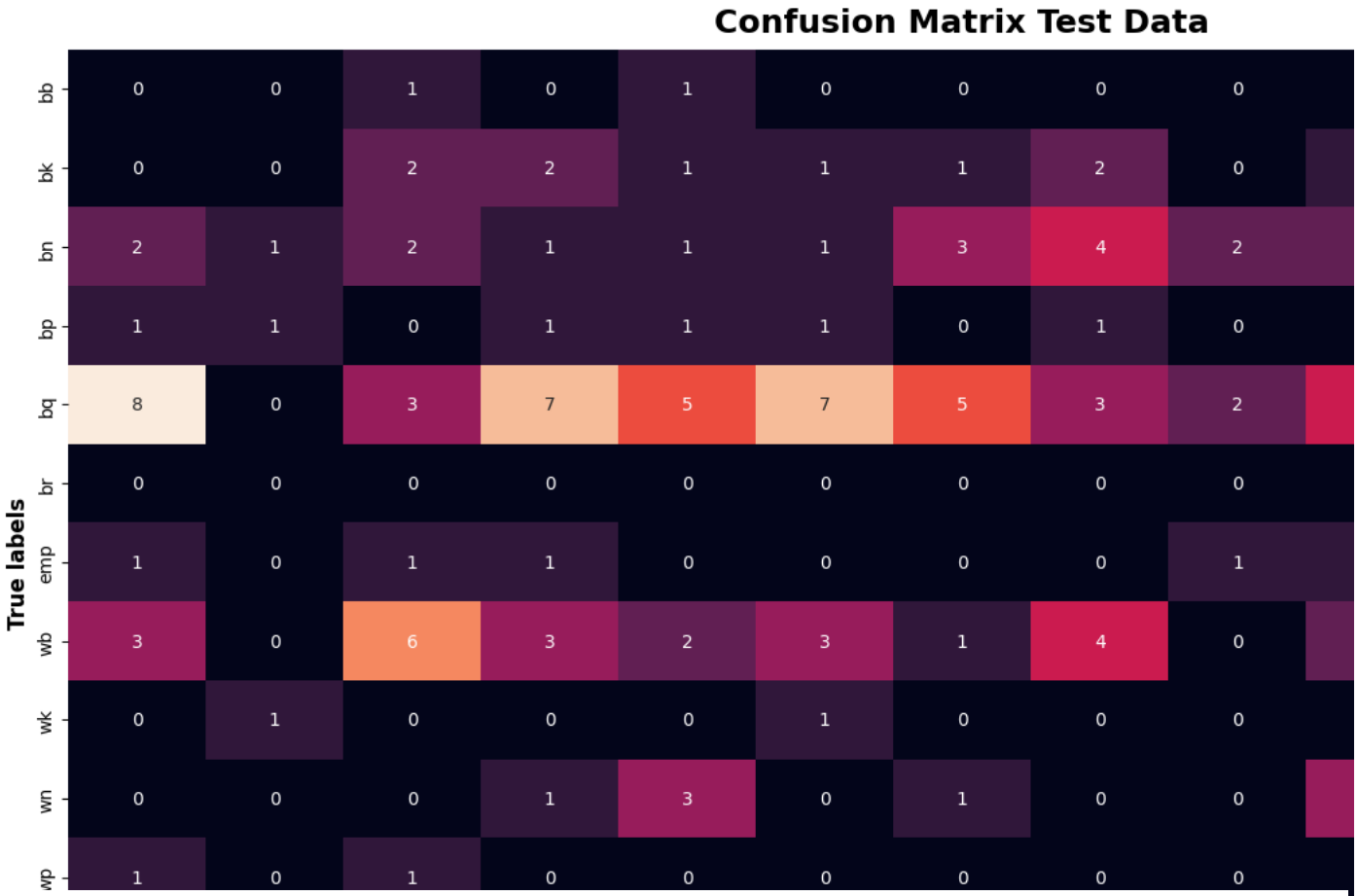
```
 1 array_train = get_labels_from_tfdataset(ds_train)
 2 y_pred_train_prob = model.predict(ds_train)
 3 y_pred_train = np.argmax(y_pred_train_prob,axis=1)
 4 y_true_train = np.argmax(array_train,axis=1)
 5 cm_train = confusion_matrix(y_pred_train, y_true_train)
 6 plt.figure(figsize=(18,9))
 7 ax_train = plt.subplot()
 8 sns.heatmap(cm_train, annot=True, fmt='g', ax=ax_train);
 9 ax_train.set_xlabel('Predicted labels');
10 ax_train.set_ylabel('True labels');
11 ax_train.set_title('Confusion Matrix Train Data');
12 ax_train.xaxis.set_ticklabels(['bb', 'bk', "bn", "bp", "bq", "br", "empty", "wb", "wk", "wn", "wp", "wq", "wr"]);
13 ax_train.yaxis.set_ticklabels(['bb', 'bk', "bn", "bp", "bq", "br", "empty", "wb", "wk", "wn", "wp", "wq", "wr"]);
14 print(['bb = black Bishop', 'bk = black King', "bn = black Knight",
15                 "bp = black Pawn", "bq = black Queen", "br = black Rook",
16                 "emp = Empty", "wb = white Bishop", "wk = white King",
17                 "wn = white Knight","wp = white Pawn", "wq = white Queen",
18                 "wr = white Rook"])
```

```
61/61 [==============================] - 1258s 21s/step
['bb = black Bishop', 'bk = black King', 'bn = black Knight', 'bp = black Pawn', 'bq = black Queen', 'br = black Rook', 'emp = Empty', 'w
```




```
array_test = get_labels_from_tfdataset(ds_test)
y_pred_test_prob = model.predict(ds_test)
y_pred_test = np.argmax(y_pred_test_prob, axis=1)
y_true_test = np.argmax(array_test,axis=1)
cm_test = confusion_matrix(y_pred_test, y_true_test)
plt.figure(figsize=(18,9))
ax_test = plt.subplot()
sns.heatmap(cm_test, annot=True, fmt='g', ax=ax_test);
ax_test.set_xlabel('Predicted labels');
ax_test.set_ylabel('True labels');
ax_test.set_title('Confusion Matrix Test Data');
ax_test.xaxis.set_ticklabels(['bb', 'bk', "bn", "bp", "bq", "br", "emp", "wb", "wk", "wn", "wp", "wq", "wr"]);
ax_test.yaxis.set_ticklabels(['bb', 'bk', "bn", "bp", "bq", "br", "emp", "wb", "wk", "wn", "wp", "wq", "wr"]);
print(['bb = black Bishop', 'bk = black King', "bn = black Knight",
                "bp = black Pawn", "bq = black Queen", "br = black Rook",
                "emp = Empty", "wb = white Bishop", "wk = white King",
                "wn = white Knight","wp = white Pawn", "wq = white Queen",
                "wr = white Rook"])
```

```
8/8 [==============================] - 146s 18s/step
['bb = black Bishop', 'bk = black King', 'bn = black Knight', 'bp = black Pawn', 'bq = black Queen', 'br = black Rook', 'emp = Empty', '
```

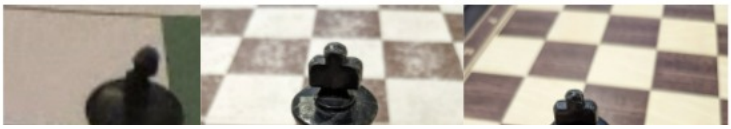



```
target_names = ['black Bishop', 'black King', "black Knight",
                "black Pawn", "black Queen", "black Rook",
```

```
                "Empty", " white Bishop", "white King",
                "white Knight","white Pawn", "white Queen",
                "white Rook"]
print(classification_report(y_pred_test, y_true_test, target_names = target_names, digits = 4))
print("total accuracy",accuracy_score(y_pred_test, y_true_test))
```

```
                  precision    recall  f1-score   support

     black Bishop     0.0000    0.0000    0.0000         2
       black King     0.0000    0.0000    0.0000        14
     black Knight     0.1176    0.0870    0.1000        23
       black Pawn     0.0526    0.1000    0.0690        10
      black Queen     0.2778    0.0847    0.1299        59
       black Rook     0.0000    0.0000    0.0000         0
            Empty     0.0000    0.0000    0.0000         6
     white Bishop     0.1905    0.1290    0.1538        31
       white King     0.0000    0.0000    0.0000         6
     white Knight     0.1500    0.3000    0.2000        10
       white Pawn     0.0000    0.0000    0.0000         2
      white Queen     0.2105    0.1176    0.1509        34
       white Rook     0.1905    0.1081    0.1379        37

        accuracy                         0.0983       234
       macro avg     0.0915    0.0713    0.0724       234
    weighted avg     0.1762    0.0983    0.1182       234

total accuracy 0.09829059829059829
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from google.colab.patches import cv2_imshow
img = cv2.imread(r'/content/drive/MyDrive/chessboard/different chess boards.jpeg',1)
cv2_imshow(img)
```

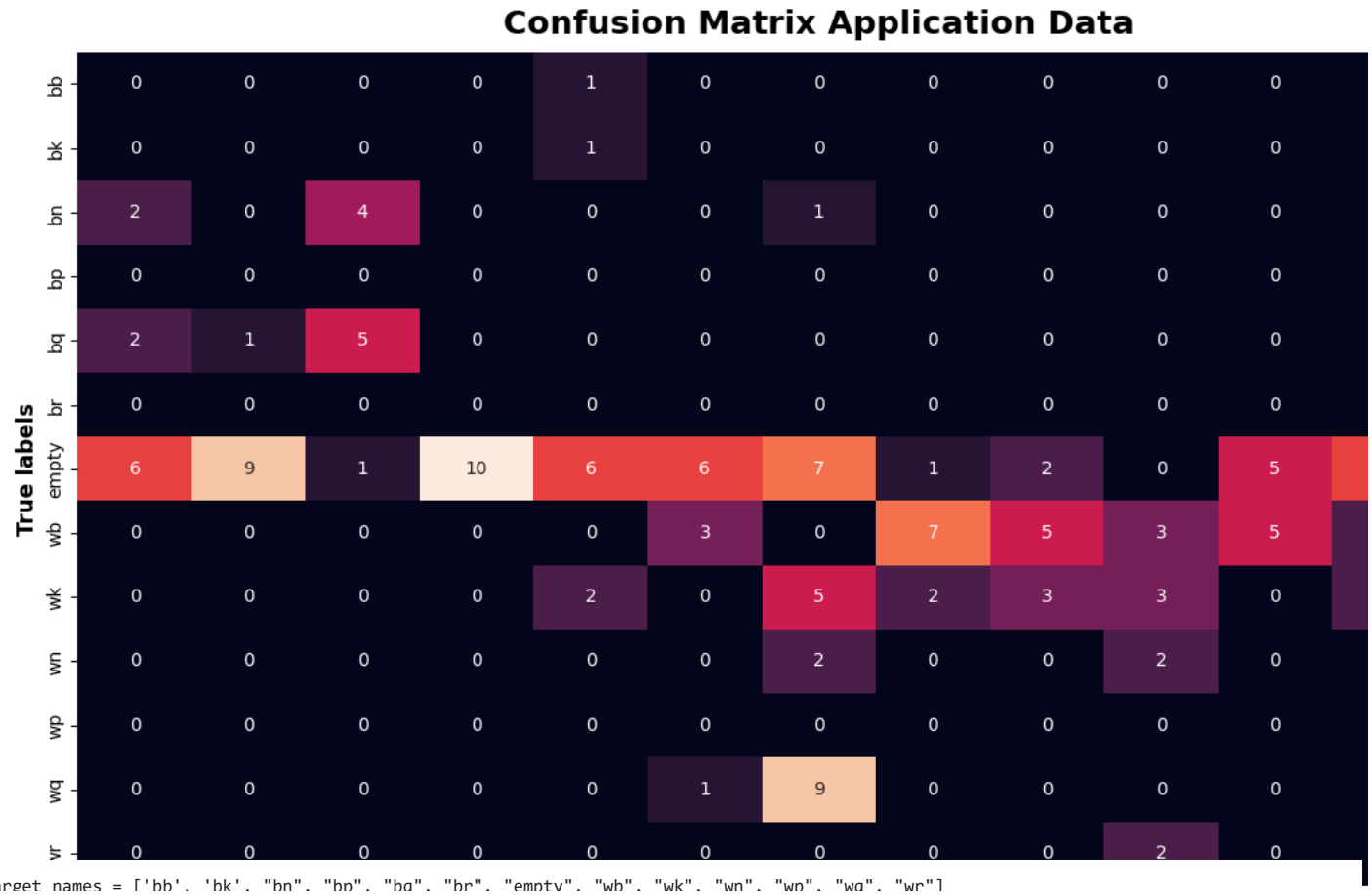


```
ds_application = image_dataset_from_directory(
   '/content/drive/MyDrive/chessboard/Application_data',
    labels='inferred',
    label_mode='categorical',
    image_size=[224, 224],
    interpolation='nearest',
    batch_size=32,
    shuffle=True,
)
AUTOTUNE = tf.data.experimental.AUTOTUNE
ds_application = (
    ds_application
   .map(convert_to_float)
   .cache()
   .prefetch(buffer_size=AUTOTUNE)
)

```

```
Found 144 files belonging to 13 classes.
```

```
array_application = get_labels_from_tfdataset(ds_application)
y_pred_application_prob = model.predict(ds_application)
y_pred_application = np.argmax(y_pred_application_prob, axis=1)
y_true_application = np.argmax(array_application,axis=1)
cm = confusion_matrix(y_pred_application, y_true_application)
plt.figure(figsize=(15,7.5))
ax_application = plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax_application);
ax_application.set_xlabel('Predicted labels');
ax_application.set_ylabel('True labels');
ax_application.set_title('Confusion Matrix Application Data');
ax_application.xaxis.set_ticklabels(['bb', 'bk', "bn", "bp", "bq", "br", "empty", "wb", "wk", "wn", "wp", "wq", "wr"]);
ax_application.yaxis.set_ticklabels(['bb', 'bk', "bn", "bp", "bq", "br", "empty", "wb", "wk", "wn", "wp", "wq", "wr"]);
```

```
5/5 [==============================] - 95s 17s/step
```




```
target_names = ['bb', 'bk', "bn", "bp", "bq", "br", "empty", "wb", "wk", "wn", "wp", "wq", "wr"]
```

```
print(classification_report(y_pred_application, y_true_application, target_names=target_names, digits=4))
print("total accuracy",accuracy_score(y_pred_application, y_true_application))
```

```
                precision    recall  f1-score   support

            bb     0.0000    0.0000    0.0000         1
            bk     0.0000    0.0000    0.0000         1
            bn     0.4000    0.5714    0.4706         7
            bp     0.0000    0.0000    0.0000         0
            bq     0.0000    0.0000    0.0000         8
            br     0.0000    0.0000    0.0000         0
         empty     0.2917    0.1094    0.1591        64
            wb     0.7000    0.2692    0.3889        26
            wk     0.3000    0.1429    0.1935        21
            wn     0.2000    0.5000    0.2857         4
            wp     0.0000    0.0000    0.0000         0
            wq     0.0000    0.0000    0.0000        10
            wr     0.0000    0.0000    0.0000         2

    accuracy                         0.1597       144
   macro avg     0.1455    0.1225    0.1152       144
weighted avg     0.3248    0.1597    0.2000       144

total accuracy 0.1597222222222222
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
```