Author Nestor Batista Díaz

CHESS

IMPORTS

In [ ]:
```python
import splitfolders
import cv2
import os
import numpy as np
import keras
import matplotlib.pyplot as plt
import Augmentor
import shutil
```

DATASET

In [ ]:
```python
original_path = "Chessman-image-dataset/Chess"
print(os.listdir(original_path))
```

```
['Bishop', 'King', 'Knight', 'Pawn', 'Queen', 'Rook']
```

In [ ]:
```python
# Esta variable contiene un mapeo de número de clase a piezas de ajedrez.
MAP_PIECE = {
    0: 'Bishop', 1: 'King', 2: 'Knight',
    3: 'Pawn', 4: 'Queen', 5: 'Rook'
}

# Vamos a standarizar todas las imágenes a tamaño 128x128
IMG_SIZE = 128
```

LIMPIEZA DE DATOS

In [ ]:
```python
# Lista de archivos que deseas excluir
exclude = ["00000176.jpg"]

# Crear una nueva carpeta para los archivos que deseas mantener
keep_folder = "chessman-image-modified"
os.makedirs(keep_folder, exist_ok=True)

# Copiar los archivos que deseas mantener a la nueva carpeta
for label, piece in MAP_PIECE.items():
    path = os.path.join(original_path, piece)
    new_path = keep_folder+"/"+piece
    os.makedirs(new_path, exist_ok= True)
    files = os.listdir(path)
    for file in files:
        if file not in exclude and (file.endswith("jpg") or file.endswith("png") or file.endswith("jpeg") or file.endswith("JPG")):
            shutil.copy(os.path.join(path, file), new_path)
```

DIVIDIR EN SPLITS

In [ ]:
```python
# dividimos los archivos en tres conjuntos entrenamiento, validación y pruebas
splitfolders.ratio(keep_folder,seed=1337, output="Chess-Splitted", ratio=(0.6, 0.2, 0.2))
```

```
Copying files: 550 files [00:08, 62.91 files/s]
```

DATA AUGMENTATION

In [ ]:
```python
# Aumentamos los datos para tener más datos de entrenamiento
def dataset_aumentation(dirname, map_pieces):
    for label, piece in map_pieces.items():
        p = Augmentor.Pipeline(dirname+"/"+piece, output_directory="")
        # Definir las operaciones de aumento que deseas aplicar
        p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
        p.flip_left_right(probability=0.5)
        p.zoom_random(probability=0.5, percentage_area=0.8)
        p.random_contrast(probability=0.5, min_factor=0.5, max_factor=1.5)
        p.random_brightness(probability=0.5, min_factor=0.5, max_factor=1.5)
        p.random_color(probability=0.5, min_factor=0.5, max_factor=1.5)

        # Especificar cuántas imágenes aumentadas deseas generar
        p.sample(2000)  # Genera 2000 nuevas imágenes
```

In [ ]:
```python
DATASET_TRAIN_PATH = "Chess-Splitted/train"
dataset_aumentation(DATASET_TRAIN_PATH, MAP_PIECE)
```

```
Initialised with 51 image(s) found.
Output directory set to Chess-Splitted/train/Bishop\.
Processing <PIL.Image.Image image mode=RGB size=391x879 at 0x200EF91BFD0>: 100%|██████████| 2000/2000 [00:41<00:00, 47.97 Samples/s]
Initialised with 45 image(s) found.
Output directory set to Chess-Splitted/train/King\.
Processing <PIL.Image.Image image mode=RGB size=587x900 at 0x200EF96ED50>: 100%|██████████| 2000/2000 [01:48<00:00, 18.47 Samples/s]
Initialised with 63 image(s) found.
Output directory set to Chess-Splitted/train/Knight\.
Processing <PIL.Image.Image image mode=RGB size=880x680 at 0x200EF90EFD0>: 100%|██████████| 2000/2000 [00:59<00:00, 33.76 Samples/s]
Initialised with 63 image(s) found.
Output directory set to Chess-Splitted/train/Pawn\.
Processing <PIL.Image.Image image mode=RGB size=1920x1080 at 0x200EF5BF310>: 100%|██████████| 2000/2000 [01:03<00:00, 31.45 Samples/s]
Initialised with 46 image(s) found.
Output directory set to Chess-Splitted/train/Queen\.
Processing <PIL.Image.Image image mode=RGB size=1000x1080 at 0x200EF862690>: 100%|██████████| 2000/2000 [00:50<00:00, 39.99 Samples/s]
Initialised with 61 image(s) found.
Output directory set to Chess-Splitted/train/Rook\.
Processing <PIL.Image.Image image mode=RGB size=400x400 at 0x200EF34FB90>: 100%|██████████| 2000/2000 [00:37<00:00, 53.52 Samples/s]
```

ENTRENAMIENTO

CARGAR DATOS

In [ ]:
```python
def load_train_set(dirname, map_pieces, verbose=True):
    """Esta función carga los datos de training en imágenes.

    Como las imágenes tienen tamaños distintas, utilizamos la librería opencv
    para hacer un resize y adaptarlas todas a tamaño IMG_SIZE x IMG_SIZE.

    Args:
        dirname: directorio completo del que leer los datos
        map_pieces: variable de mapeo entre labels y piezas
        verbose: si es True, muestra información de las imágenes cargadas

    Returns:
        X, y: X es un array con todas las imágenes cargadas con tamaño
                IMG_SIZE x IMG_SIZE
              y es un array con las labels de correspondientes a cada imagen
    """
    X_train = []
    y_train = []
    for label, piece in map_pieces.items():
        files = os.listdir(os.path.join(dirname, piece))
        images = [file for file in files if (file.endswith("jpg") or file.endswith("png") or file.endswith("jpeg") or file.endswith("JPG"))]
        if verbose:
          print("Leyendo {} imágenes encontradas de {}".format(len(images), piece))
        for image_name in images:
            image = cv2.imread(os.path.join(dirname, piece, image_name))
            X_train.append(cv2.resize(image,(IMG_SIZE, IMG_SIZE)))
            y_train.append(label)
    return np.array(X_train), np.array(y_train)
```

In [ ]:
```python
def load_test_set(dirname, map_pieces, verbose=True):
    """Esta función funciona de manera equivalente a la función load_train_set
    pero cargando los datos de test."""
    X_test = []
    y_test = []
    for label, piece in map_pieces.items():
        files = os.listdir(os.path.join(dirname, piece))
        images = [file for file in files if (file.endswith("jpg") or file.endswith("png") or file.endswith("jpeg") or file.endswith("JPG"))]
        for image_name in images:
            image = cv2.imread(os.path.join(dirname, piece, image_name))
            X_test.append(cv2.resize(image,(IMG_SIZE, IMG_SIZE)))
            y_test.append(label)
    if verbose:
        print("Leídas {} imágenes de test".format(len(X_test)))
    return np.array(X_test), np.array(y_test)
```

In [ ]:
```python
# Cargamos los datos. Si no estás trabajando en colab, cambia los paths por
# los de los ficheros donde hayas descargado los datos.
DATASET_TRAIN_PATH = "Chess-Splitted/train"
DATASET_TEST_PATH = "Chess-Splitted/test"

X, y = load_train_set(DATASET_TRAIN_PATH, MAP_PIECE)
X_t, y_t = load_test_set(DATASET_TEST_PATH, MAP_PIECE)
```

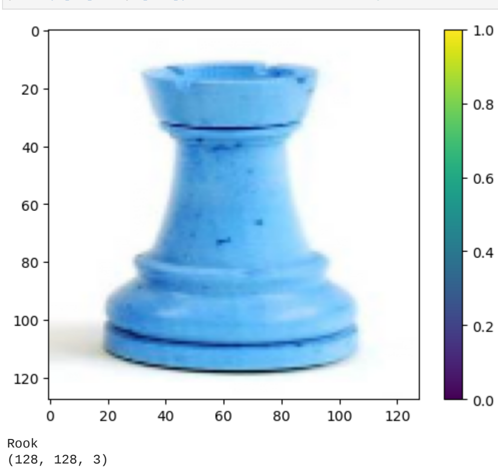
```
Leyendo 2051 imágenes encontradas de Bishop
Leyendo 2045 imágenes encontradas de King
Leyendo 2063 imágenes encontradas de Knight
Leyendo 2063 imágenes encontradas de Pawn
Leyendo 2046 imágenes encontradas de Queen
Leyendo 2061 imágenes encontradas de Rook
Leídas 112 imágenes de test
```

In [ ]:
```python
# Vamos a barajar aleatoriamente los datos. Esto es importante ya que si no
# lo hacemos y, por ejemplo, cogemos el 20% de los datos finales como validation
# set, estaremos utilizando solo un pequeño número de piezas, ya que
# las imágenes se leen secuencialmente pieza a pieza.
perm = np.random.permutation(len(X))
X, y = X[perm], y[perm]
```

In [ ]:
```python
# Preprocessing normalización de las imágenes
X = X / 255.0
X_t = X_t / 255.0
```

In [ ]:
```python
def visualize_example(x):
  plt.figure()
  plt.imshow(x)
  plt.colorbar()
  plt.grid(False)
  plt.show()
```

In [ ]:
```python
visualize_example(X[70]) # Visualiza la imagen normalizada con valores de 0 a 1
print(MAP_PIECE[y[70]]) # Acceso al diccionario
print(X[70].shape[0:3]) # Dimensiones de la imagen tras resize con los 3 canales RGB
```




```
Rook
(128, 128, 3)
```

In [ ]:
```python
#batch_size=128
num_classes=6
epochs=10
img_rows,img_cols=IMG_SIZE,IMG_SIZE
input_shape=(img_rows,img_cols,3) #(128,128,3)
```

In [ ]:
```python
# convert class vectors to binary class matrices
y=keras.utils.to_categorical(y,num_classes)
y_t=keras.utils.to_categorical(y_t,num_classes)
```

In [ ]:
```python
X.shape
```

Out[ ]:
```
(12329, 128, 128, 3)
```

In [ ]:
```python
y.shape
```

Out[ ]:
```
(12329, 6)
```

CARGAR MODELO

In [ ]:
```python
# Definimos un modelo de red neuronal convolucional para clasificación con varias capas convolucionales,
# capas de reducción de dimensionalidad, capas densas y regularización para prevenir el sobreajuste
# durante el entrenamiento.

model = keras.Sequential([
keras.layers.Conv2D(32, kernel_size=(3, 3), data_format="channels_last",input_shape=input_shape, activation='relu'),
keras.layers.MaxPooling2D(pool_size=(2, 2)),
keras.layers.Conv2D(128, kernel_size=(3, 3), activation='relu'),
keras.layers.MaxPooling2D(pool_size=(2, 2)),
keras.layers.Conv2D(512, kernel_size=(3, 3), activation='relu'),
keras.layers.MaxPooling2D(pool_size=(2, 2)),
keras.layers.MaxPooling2D(pool_size=(2, 2)),
keras.layers.Flatten(),
keras.layers.Dense(128, activation='relu', kernel_regularizer=keras.regularizers.l1_l2(l1=0.0001, l2=0.0001)),
keras.layers.Dropout(0.5),
keras.layers.Dense(64, activation='relu', kernel_regularizer=keras.regularizers.l1_l2(l1=0.0001, l2=0.0001)),
keras.layers.Dense(6, activation='softmax')
])
```

In [ ]:
```python
model.summary()
model.compile(optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 126, 126, 32)      896

 max_pooling2d_20 (MaxPooli  (None, 63, 63, 32)        0
 ng2D)

 conv2d_16 (Conv2D)          (None, 61, 61, 128)       36992

 max_pooling2d_21 (MaxPooli  (None, 30, 30, 128)       0
 ng2D)

 conv2d_17 (Conv2D)          (None, 28, 28, 512)       590336

 max_pooling2d_22 (MaxPooli  (None, 14, 14, 512)       0
 ng2D)

 max_pooling2d_23 (MaxPooli  (None, 7, 7, 512)         0
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 126, 126, 32)      896

 max_pooling2d_20 (MaxPooli  (None, 63, 63, 32)        0
 ng2D)

 conv2d_16 (Conv2D)          (None, 61, 61, 128)       36992

 max_pooling2d_21 (MaxPooli  (None, 30, 30, 128)       0
 ng2D)

 conv2d_17 (Conv2D)          (None, 28, 28, 512)       590336

 max_pooling2d_22 (MaxPooli  (None, 14, 14, 512)       0
 ng2D)

 max_pooling2d_23 (MaxPooli  (None, 7, 7, 512)         0
 ng2D)

 flatten_5 (Flatten)         (None, 25088)             0

 dense_15 (Dense)            (None, 128)               3211392

 dropout_5 (Dropout)         (None, 128)               0

 dense_16 (Dense)            (None, 64)                8256

 dense_17 (Dense)            (None, 6)                 390

=================================================================
Total params: 3848262 (14.68 MB)
Trainable params: 3848262 (14.68 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

ENTRENAMIENTO

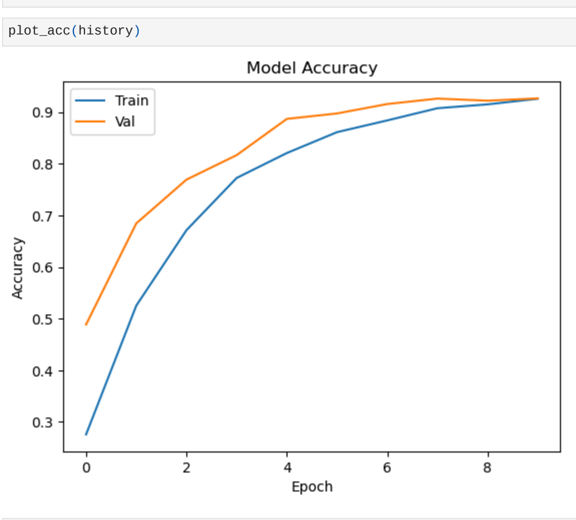
In [ ]:
```python
history=model.fit(X, y, epochs=epochs, validation_split = 0.2)
```

```
Epoch 1/10
309/309 [==============================] - 234s 755ms/step - loss: 2.0273 - accuracy: 0.2758 - val_loss: 1.6166 - val_accuracy: 0.4886
Epoch 2/10
309/309 [==============================] - 238s 770ms/step - loss: 1.5557 - accuracy: 0.5253 - val_loss: 1.2871 - val_accuracy: 0.6845
Epoch 3/10
309/309 [==============================] - 243s 788ms/step - loss: 1.2810 - accuracy: 0.6714 - val_loss: 1.0959 - val_accuracy: 0.7693
Epoch 4/10
309/309 [==============================] - 223s 723ms/step - loss: 1.0871 - accuracy: 0.7725 - val_loss: 1.0132 - val_accuracy: 0.8167
Epoch 5/10
309/309 [==============================] - 222s 718ms/step - loss: 0.9883 - accuracy: 0.8207 - val_loss: 0.8383 - val_accuracy: 0.8869
Epoch 6/10
309/309 [==============================] - 229s 743ms/step - loss: 0.8773 - accuracy: 0.8612 - val_loss: 0.7846 - val_accuracy: 0.8974
Epoch 7/10
309/309 [==============================] - 221s 715ms/step - loss: 0.8293 - accuracy: 0.8839 - val_loss: 0.7588 - val_accuracy: 0.9157
Epoch 8/10
309/309 [==============================] - 220s 712ms/step - loss: 0.7689 - accuracy: 0.9075 - val_loss: 0.7380 - val_accuracy: 0.9262
Epoch 9/10
309/309 [==============================] - 220s 714ms/step - loss: 0.7412 - accuracy: 0.9150 - val_loss: 0.7290 - val_accuracy: 0.9221
Epoch 10/10
309/309 [==============================] - 219s 710ms/step - loss: 0.7067 - accuracy: 0.9259 - val_loss: 0.7317 - val_accuracy: 0.9266
```

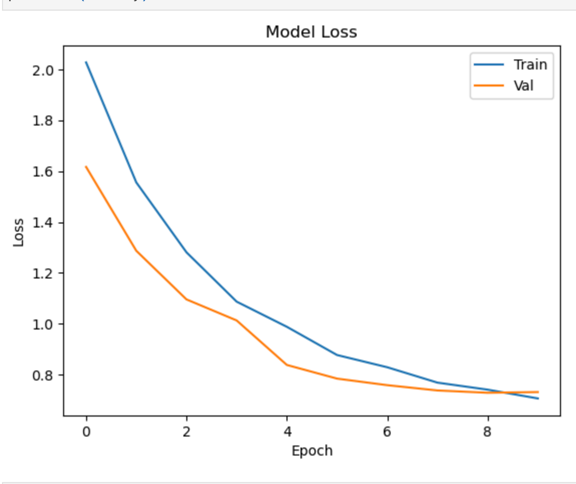
CONCLUSIONES

In [ ]:
```python
def plot_acc(history, title="Model Accuracy"):
  """Imprime una gráfica mostrando la accuracy por epoch obtenida en un entrenamiento"""
  plt.plot(history.history['accuracy'])
  plt.plot(history.history['val_accuracy'])
  plt.title(title)
  plt.ylabel('Accuracy')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Val'], loc='upper left')
  plt.show()
def plot_loss(history, title="Model Loss"):
  """Imprime una gráfica mostrando la pérdida por epoch obtenida en un entrenamiento"""
  plt.plot(history.history['loss'])
  plt.plot(history.history['val_loss'])
  plt.title(title)
  plt.ylabel('Loss')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Val'], loc='upper right')
  plt.show()
```

In [ ]:
```python
plot_acc(history)
```



In [ ]:
```python
plot_loss(history)
```



In [ ]:
```python
model.evaluate(X_t, y_t, batch_size=32, verbose=1)
```

```
4/4 [==============================] - 1s 113ms/step - loss: 2.5377 - accuracy: 0.6161
```

Out[ ]:
```
[2.5377461910247803, 0.6160714030265808]
```

In [ ]:
```python
prediccion=model.predict(X_t, batch_size=32, verbose=1)
```

```
4/4 [==============================] - 1s 107ms/step
```

In [ ]:
```python
print(prediccion)
prediccion.shape
```

```
[[9.99989986e-01 1.23765449e-06 1.43543982e-10 8.65401717e-07
  7.86538658e-06 1.43995635e-10]
 [9.38722670e-01 2.28202827e-02 8.29503506e-07 1.07248279e-03
  3.73432189e-02 4.05952633e-05]
 [2.00341158e-02 2.91920230e-02 3.47939171e-02 1.05111599e-02
  6.07802570e-02 8.44688535e-01]
 [9.99999762e-01 1.22084472e-07 8.97493207e-14 6.53533476e-08
  5.24970378e-09 6.51895530e-14]
 [3.28495279e-02 6.09984040e-01 2.11439124e-06 3.19181412e-01
  3.78259830e-02 1.56862923e-04]
 [3.64172533e-02 6.48538730e-07 3.65072815e-03 9.58877802e-01
  1.59134254e-06 1.05198740e-03]
 [9.99992490e-01 6.76408990e-06 6.60458830e-14 2.92565744e-07
  4.27681812e-07 7.33346082e-14]
 [9.50592160e-01 1.59057416e-03 2.11506267e-03 3.94361354e-02
  4.49742982e-03 1.76859600e-03]
 [9.99411345e-01 2.27092787e-05 1.88691857e-12 3.64478031e-10
  5.65881142e-04 6.15232344e-12]
 [9.94027019e-01 1.71328807e-07 2.56578359e-09 5.97281335e-03
  3.09671080e-08 1.31139410e-09]
 [9.99990463e-01 4.83158783e-06 1.19194503e-08 3.58187140e-06
  1.08847087e-06 7.15771176e-09]
 [6.57171418e-04 1.57882008e-04 1.08505830e-01 7.90066086e-03
  6.90908264e-03 8.75869334e-01]
 [7.36915767e-01 5.67539973e-06 2.33370083e-05 2.63020426e-01
  3.03704765e-05 4.34836466e-06]
 [9.99996424e-01 3.45751164e-06 6.53144723e-16 7.25617610e-09
  6.54028796e-08 1.05964825e-13]
 [9.99987841e-01 2.21024578e-07 2.04479051e-07 7.71629311e-06
  3.90485957e-06 1.54696007e-08]
 [9.99439180e-01 1.24983387e-06 1.59450121e-06 5.38021035e-04
  1.82361982e-05 1.76796561e-06]
 [2.12701940e-04 9.94591296e-01 9.92074700e-17 5.19563165e-03
  3.99636406e-07 1.25919206e-14]
 [2.93057383e-05 7.14355394e-07 1.60445040e-07 9.99968290e-01
  1.46239017e-06 9.61708011e-08]
 [1.15754424e-16 1.00000000e+00 4.68861718e-33 9.69367124e-22
  1.99740335e-09 3.51781658e-20]
 [2.22651092e-10 9.99989510e-01 6.49496840e-19 1.77125838e-12
  1.05063018e-05 7.45879053e-14]
 [1.03876278e-06 9.96610582e-01 1.18166227e-10 3.27236869e-08
  3.38773243e-03 6.03075023e-07]
 [4.65722442e-05 1.12532467e-01 1.14124487e-06 1.58625844e-05
  8.70508313e-01 1.68956630e-02]
 [1.35421353e-11 1.00000000e+00 6.33832509e-23 2.92376630e-14
  5.03785680e-09 1.08879076e-15]
 [1.17654537e-07 9.80755329e-01 1.09385759e-14 4.26892723e-11
  1.92445312e-02 2.22914487e-08]
 [4.36780356e-05 3.17906786e-04 7.07400458e-12 3.42579809e-09
  9.99638200e-01 1.93613175e-07]
 [3.00048897e-03 5.09095371e-01 3.50911279e-07 9.62807753e-05
  4.87567723e-01 2.39804358e-04]
 [3.20397485e-05 8.16495195e-02 2.65247690e-09 1.46817541e-07
  9.18304920e-01 1.33051999e-05]
 [3.60182896e-02 1.93860177e-02 2.40601603e-05 1.97993056e-03
  9.40462947e-01 2.12876801e-03]
 [6.29229646e-09 1.00000000e+00 3.00330277e-22 1.41383058e-10
  6.35639041e-11 1.54562538e-15]
 [1.22237764e-03 6.25710845e-01 4.99655206e-09 1.40167531e-05
  3.73008996e-01 4.37472736e-05]
 [1.95509114e-04 9.99782622e-01 1.69612154e-13 1.49185723e-07
  2.18453915e-05 8.15165935e-10]
 [8.11391920e-02 3.80412519e-01 6.11795201e-08 1.80355133e-03
  5.36615789e-01 2.89312466e-05]
 [2.26581059e-02 4.90298271e-01 4.71976513e-10 5.25444875e-06
  4.87035602e-01 2.75808929e-06]
 [2.64602185e-09 1.19711671e-11 9.99997497e-01 1.37072490e-07
  1.82800697e-09 2.43863315e-06]
 [1.11424239e-08 4.26968920e-12 9.99999642e-01 2.80978952e-07
  8.47939600e-12 1.47631653e-07]
 [1.69698522e-09 2.00726827e-12 9.99999642e-01 2.78112026e-07
  1.43839177e-10 6.91667168e-08]
 [9.19146636e-18 1.14836412e-33 1.00000000e+00 6.67790061e-15
  6.11900370e-34 5.00652278e-28]
 [1.18968240e-03 7.01169629e-05 9.55753028e-01 5.94892539e-03
  2.10711989e-03 3.49310972e-02]
 [2.20950938e-07 1.95165941e-15 9.99997973e-01 1.76157209e-06
  1.94530925e-14 2.85247435e-08]
 [2.20604246e-09 1.10306000e-10 8.96625280e-01 1.03374153e-01
  2.67484634e-09 6.69251676e-07]
 [1.55968830e-20 0.00000000e+00 1.00000000e+00 1.93288103e-23
  0.00000000e+00 3.96296009e-30]
 [2.68273567e-07 8.98727393e-19 9.99999404e-01 3.95443408e-07
  1.71642062e-14 2.97544087e-08]
 [1.49129919e-05 1.93642043e-12 9.99245524e-01 7.39603187e-04
  6.95108858e-12 3.12145065e-09]
 [7.68168747e-01 5.37833665e-04 2.21190125e-01 7.06190662e-03
  2.54119560e-03 5.00198570e-04]
 [5.92920464e-04 6.76495119e-07 9.96589899e-01 1.90402311e-03
  1.19340184e-05 9.00593761e-04]
 [1.69802220e-16 2.63209689e-23 1.00000000e+00 2.56332700e-17
  1.72371289e-20 2.37593441e-15]
 [8.15864354e-02 2.71882296e-01 6.66441917e-02 4.67794567e-01
  1.09494723e-01 2.59788963e-03]
 [4.53091682e-32 0.00000000e+00 1.00000000e+00 2.00290804e-28
  0.00000000e+00 4.21091969e-34]
 [2.43692788e-10 8.28380190e-18 1.00000000e+00 2.74013767e-09
  9.79917805e-16 2.32305303e-11]
 [5.14119173e-12 9.67241153e-26 1.00000000e+00 7.01549387e-13
  8.05148595e-24 9.68792362e-18]
 [4.17170720e-03 1.18379367e-05 9.04847264e-01 1.02582749e-03
  5.12324390e-04 8.94310176e-02]
 [7.95405697e-20 3.76093196e-34 1.00000000e+00 8.14330775e-19
  2.64654642e-29 1.34434419e-22]
 [3.21524846e-03 3.20396602e-01 6.65282924e-03 8.44525115e-04
  5.70095003e-01 9.87958089e-02]
 [1.17789702e-02 8.65242982e-05 8.97756219e-01 7.87036642e-02
  1.33579981e-03 1.03389211e-02]
 [6.51992252e-03 1.16394506e-06 1.58382754e-04 9.93317962e-01
  4.55026054e-07 2.01010812e-06]
 [4.14892519e-03 2.58840719e-05 1.81000665e-04 5.38672566e-01
  3.25304898e-03 4.53718543e-01]
 [4.04485872e-05 1.24127278e-10 1.04714521e-08 9.99959588e-01
  3.28776855e-13 1.00344602e-12]
 [5.90693958e-07 1.61144360e-08 2.99106287e-05 9.93613422e-01
  7.74972193e-07 6.35521207e-03]
 [2.29325800e-04 3.98158960e-07 2.48586945e-03 9.64350700e-01
  8.05697312e-07 3.29329744e-02]
 [4.37290013e-01 2.16148607e-03 8.95652629e-04 5.48734903e-01
  7.58671248e-03 3.33123025e-03]
 [2.87211640e-03 2.12024134e-02 7.77499401e-04 2.99530089e-01
  2.64663436e-03 6.72971368e-01]
 [1.95638329e-01 2.51755921e-07 1.32406625e-04 8.04228842e-01
  1.18198798e-07 2.31330972e-08]
 [2.87894718e-02 2.43900623e-03 1.63049642e-02 4.37665656e-02
  1.75912923e-03 9.06940877e-01]
 [9.78305280e-01 1.70234256e-04 4.68943873e-03 3.33108660e-03
  2.46426137e-03 1.10396994e-02]
 [7.69157414e-05 2.63555000e-09 1.46980739e-07 9.99922991e-01
  1.10141141e-09 2.34303821e-09]
 [5.34660671e-07 5.22514323e-11 5.53923974e-06 9.99993920e-01
  2.27400458e-15 3.55533907e-13]
 [4.58072312e-02 4.03722282e-03 2.94447709e-05 5.15762568e-01
  8.37233104e-03 4.25991267e-01]
 [7.35975137e-13 5.71883134e-27 1.18627699e-12 1.00000000e+00
  2.48068944e-27 8.79025315e-18]
 [9.24257189e-02 2.56011321e-04 3.53407562e-01 2.41539888e-02
  1.12882508e-02 5.18468440e-01]
 [9.80340838e-01 3.53258540e-04 1.04169156e-02 4.31804545e-03
  2.09029694e-03 2.48050806e-03]
 [1.10904849e-03 1.39641306e-05 9.02586162e-01 4.06736089e-03
  4.49518819e-04 9.17739719e-02]
 [3.64000052e-01 4.44341637e-03 4.22566518e-05 5.72113276e-01
  5.90774752e-02 3.23552871e-04]
 [1.06608513e-06 1.98509768e-11 3.71536968e-11 9.99998927e-01
  1.90631603e-12 1.73497902e-12]
 [1.04160387e-08 1.55364475e-18 2.11993847e-14 1.00000000e+00
  1.96469445e-19 4.92437615e-20]
 [5.47272444e-01 8.34169760e-02 6.38310166e-06 3.68272662e-01
  1.00312708e-03 2.84260386e-05]
 [3.68204752e-07 9.88177240e-01 1.35071182e-11 2.13230358e-08
  1.18210921e-02 1.36141557e-06]
 [1.73073266e-17 1.00000000e+00 6.21069025e-28 2.30477665e-15
  5.20028638e-12 7.78299893e-21]
 [2.95682879e-12 9.99992132e-01 9.92085043e-24 8.69621419e-14
  7.88326179e-06 1.04807071e-15]
 [2.26445198e-02 3.51328343e-01 3.14343633e-05 4.12600785e-01
  2.08962649e-01 4.43226937e-03]
 [4.53479707e-01 5.34986913e-01 4.75989373e-06 4.04228253e-04
  1.10405786e-02 8.37631160e-05]
 [9.67442095e-01 1.24508620e-03 9.35651272e-08 3.12914215e-02
  1.80532224e-05 3.23088943e-06]
 [2.22021318e-03 8.34199369e-01 1.92895428e-07 7.89625556e-05
  1.61506921e-01 1.99431856e-03]
 [2.96615195e-02 6.50945783e-01 6.29238315e-08 5.91106655e-05
  3.19062948e-01 2.70557503e-04]
 [1.32005230e-01 1.40541986e-01 3.17634828e-02 2.49242038e-01
  3.34136069e-01 1.12311162e-01]
 [4.72443253e-06 4.79267128e-02 3.23824799e-17 4.15999069e-10
  9.52068627e-01 8.00588129e-09]
 [1.45732262e-03 9.58783329e-01 1.54835652e-04 4.48836666e-03
  2.21632626e-02 1.29528418e-02]
 [9.77398455e-01 1.24242418e-02 8.40524539e-10 1.14801616e-04
  1.00613907e-02 1.24000337e-06]
 [1.33362948e-04 9.96940136e-01 1.57149193e-14 8.37115532e-09
  2.92640808e-03 1.46066512e-07]
 [4.70787170e-04 8.95174921e-01 1.25800426e-07 1.22151745e-04
  1.04228348e-01 3.62825290e-06]
 [4.90566522e-01 9.80201587e-02 5.45606126e-05 8.62635002e-02
  3.23781788e-01 1.31342199e-03]
 [1.57630502e-03 9.32735333e-04 9.35269728e-09 2.22527410e-06
  9.97461915e-01 2.67329360e-05]
 [9.04605666e-04 9.51160036e-05 4.32355795e-04 2.87140859e-03
  1.43770538e-02 9.81319547e-01]
 [6.54707401e-05 1.35779346e-03 2.55247322e-03 8.47875178e-01
  4.59641014e-05 1.48103118e-01]
 [1.90105184e-05 1.13032968e-06 9.09840763e-01 7.76879824e-05
  3.18504885e-06 9.00582150e-02]
 [2.45624232e-08 1.39323122e-08 1.35045440e-07 2.68203507e-06
  1.11085070e-08 9.99997139e-01]
 [2.93578023e-10 1.19617614e-08 1.89189211e-06 3.32811425e-07
  4.37382859e-08 9.99997735e-01]
 [8.61106921e-13 3.15756807e-13 2.14156748e-10 1.51890532e-07
  3.82489570e-12 9.99999881e-01]
 [6.31169223e-06 1.74241571e-03 2.30724528e-03 1.49210787e-03
  1.64757064e-03 9.92804348e-01]
 [7.23099802e-03 6.45172549e-05 2.17793695e-06 1.41272585e-05
  9.91759837e-01 9.28308931e-04]
 [9.86647283e-05 2.95988229e-07 5.09444042e-04 6.16957186e-05
  8.11291335e-04 9.98518527e-01]
 [5.93271196e-11 3.44119733e-10 2.09812145e-09 1.90121288e-08
  1.78450121e-09 1.00000000e+00]
 [2.92282905e-02 2.05213059e-04 1.37776020e-04 2.07015313e-02
  8.93385768e-01 5.63413985e-02]
 [1.10796182e-05 5.26092461e-17 4.27204072e-01 5.72774351e-01
  8.04962989e-14 1.04976380e-05]
 [1.62520064e-05 8.11752579e-06 7.72443818e-05 1.12432579e-04
  8.73218305e-05 9.99698639e-01]
 [5.58296465e-13 1.56519505e-13 4.51025634e-10 1.02974268e-10
  2.62026147e-11 1.00000000e+00]
 [4.04762337e-03 3.40769626e-03 1.71856925e-01 9.19754803e-03
  2.74761647e-01 5.36728501e-01]
 [1.87235866e-02 2.96520186e-04 4.10190638e-04 9.74176764e-01
  3.57988453e-03 2.81306636e-03]
 [3.10458677e-06 7.43742108e-01 3.06913421e-06 1.32355164e-03
  1.61488559e-02 2.38779232e-01]
 [6.11534622e-03 3.73029662e-03 3.32068908e-03 2.76080985e-02
  1.17849121e-02 9.47440743e-01]
 [1.12704327e-03 4.80365088e-06 9.78093803e-01 9.54776246e-04
  3.03427572e-03 1.67852286e-02]
 [3.16863543e-06 1.72338090e-07 6.11345386e-05 4.66329783e-01
  9.39750316e-06 5.33596337e-01]
 [1.03722448e-02 2.21675232e-01 1.93171867e-10 4.56370035e-05
  7.67906427e-01 4.92209551e-07]]
```

Out[ ]:
```
(112, 6)
```

In [ ]:
```python
print(prediccion[0]) # Ejemplo de predicción para el primer elemento del test.
predicho = np.argmax(prediccion[0])  # Nos quedamos con la posición del valor máximo de las estimaciones de probabilidad para cada una de las 6 clases.
print(predicho)
print(MAP_PIECE[predicho]) # Acceso al diccionario para mostrar el nombre de la pieza predicha.
```

```
[9.9998999e-01 1.2376545e-06 1.4354398e-10 8.6540172e-07 7.8653866e-06
 1.4399563e-10]
0
Bishop
```

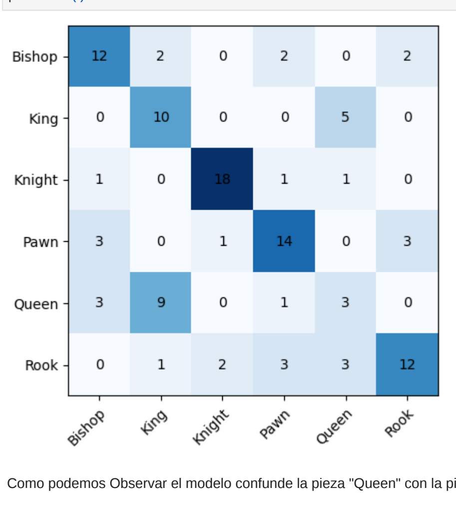
In [ ]:
```python
y_t.shape
```

Out[ ]:
```
(112, 6)
```

In [ ]:
```python
from sklearn.metrics import confusion_matrix, classification_report
LIST_PIECE = ['Bishop','King','Knight','Pawn', 'Queen', 'Rook']
informe = classification_report(np.argmax(y_t,axis=1), np.argmax(prediccion,axis=1), target_names= LIST_PIECE)
print(informe)
```

```
              precision    recall  f1-score   support

      Bishop       0.63      0.67      0.65        18
        King       0.45      0.67      0.54        15
      Knight       0.86      0.86      0.86        21
        Pawn       0.67      0.67      0.67        21
       Queen       0.25      0.19      0.21        16
        Rook       0.71      0.57      0.63        21

    accuracy                           0.62       112
   macro avg       0.59      0.60      0.59       112
weighted avg       0.62      0.62      0.61       112
```

In [ ]:
```python
# Calcular la matriz de confusión para clasificación multi-label
matriz_confusion = confusion_matrix(np.argmax(y_t,axis=1), np.argmax(prediccion,axis=1))

# Crear una figura y ejes
fig, ax = plt.subplots()
im = ax.imshow(matriz_confusion, cmap='Blues')

# Mostrar las etiquetas en los ejes
ax.set_xticks(np.arange(len(LIST_PIECE)))
ax.set_yticks(np.arange(len(LIST_PIECE)))
ax.set_xticklabels(LIST_PIECE)
ax.set_yticklabels(LIST_PIECE)

# Rotar las etiquetas y ajustar la posición
plt.setp(ax.get_xticklabels(), rotation=45, ha="right", rotation_mode="anchor")

# Mostrar los valores en cada celda
for i in range(len(LIST_PIECE)):
    for j in range(len(LIST_PIECE)):
        text = ax.text(j, i, matriz_confusion[i, j], ha="center", va="center", color="black")

plt.show()
```



Como podemos Observar el modelo confunde la pieza "Queen" con la pieza "King", habria que presentarle más imagenes de estas dos piezas para que pueda distinguir más facilmente entre ambas.