



Universidad Nacional Autónoma de México

Facultad de Ingeniería

Ingeniería en computación



Inteligencia Artificial (406)

*Profesor: Ing. Jorge Alberto
Hernández Nieto*

Semestre 2025-1

Práctica 3. Red Neuronal

Grupo: 5

Integrantes del Equipo:

Bravo Luna Ivonne Monserrat

Cruz Maldonado Armando

Diaz Gonzalez Rivas Angel Iñaqui

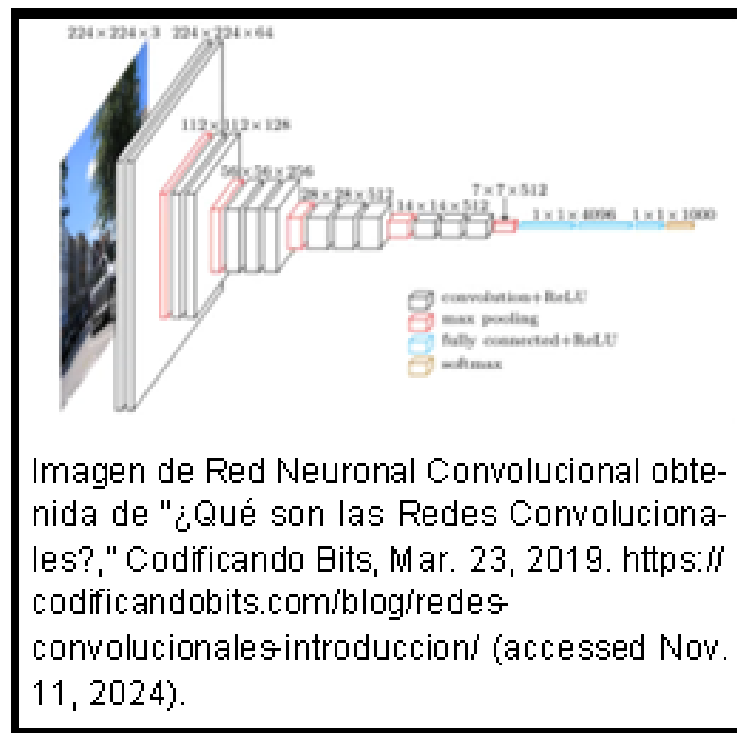
Olivos Jiménez Luis Mario

Siliano Haller Rodrigo

Zarco Romero José Alberto

Índice o contenido

1. Objetivo.....	2
2. Introducción.....	2
3. Desarrollo.....	3
3.1. Red Neuronal. Ejercicio propuesto.....	3
3.1.1. Código.....	4
3.1.2. Captura de la Ejecución.....	12
4. Conclusiones.....	13
5. Bibliografía / Referencias electrónicas.....	14



1. Objetivo

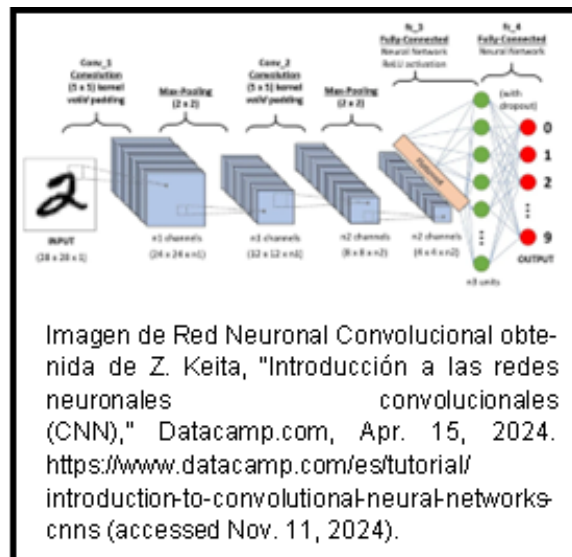
El estudiante desarrollará un sistema de análisis automatizado mediante redes neuronales que permita identificar y clasificar especies de animales en peligro de extinción a partir de videos, por medio de un software de extracción de fotogramas (como VLC) y técnicas de visión por computadora, se busca extraer imágenes clave de los videos, procesarlas e implementarlas en una red neuronal convolucional (CNN) para reconocer dichas especies. Al finalizar, se obtendrá un registro detallado de las especies identificadas.

2. Introducción

Una red neuronal es un método de la inteligencia artificial que enseña a las computadoras a procesar datos de una manera similar a como lo hace el cerebro humano. Se trata de un tipo de proceso de machine learning llamado aprendizaje profundo, el cual utiliza los nodos o las neuronas interconectados en una estructura de capas que se parece al cerebro humano. [1]

Las redes neuronales pueden ayudar a las computadoras a tomar decisiones inteligentes con asistencia humana limitada. Esto se debe a que pueden aprender y modelar las relaciones entre los datos de entrada y salida que no son lineales y que son complejos. [1]

Las redes neuronales se basan en datos de entrenamiento para aprender y mejorar su precisión con el tiempo. Una vez que se ajustan para obtener precisión, son herramientas poderosas en informática e inteligencia artificial, lo que nos permite clasificar y agrupar datos a alta velocidad. [2]



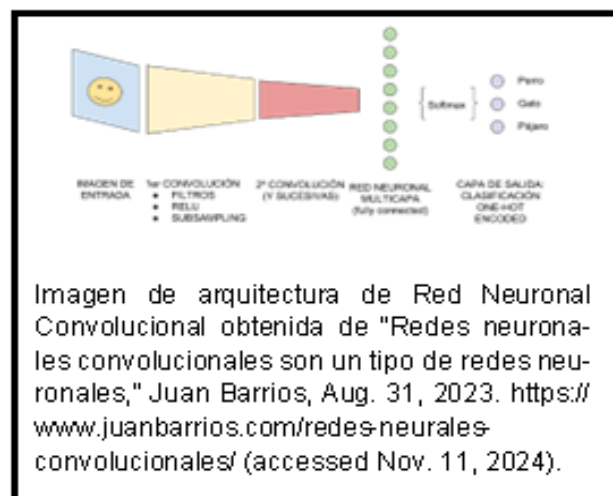
Las redes neuronales son muy útiles para, a partir de un dato de entrada, hacer una clasificación o regresión a partir de un modelo entrenado. Esta misma idea se puede aplicar a una imagen, para obtener su clasificación o detectar un objeto que forme parte de ella. Para ello se recurre a las redes neuronales convolucionales (CNN), las cuales han supuesto una revolución en el sector del reconocimiento de imágenes, ya que usan un procesamiento relativamente pequeño. [3]

En esta práctica, se utilizará una red neuronal convolucional para analizar un video mediante la extracción de imágenes clave o fotogramas con el software VLC. A partir de estos fotogramas, se buscará identificar y clasificar especies de animales en peligro de extinción. Este proceso permitirá explorar el uso de redes neuronales en la clasificación de imágenes y su potencial para el monitoreo de la biodiversidad, creando un registro de los animales detectados en el video y contribuyendo a la preservación de la fauna en riesgo.

3. Desarrollo

3.1. Red Neuronal. Ejercicio propuesto

Utilizando los fotogramas extraídos, se propone entrenar y ajustar una red neuronal para que, a partir de cada imagen, identifique las especies en peligro de extinción que aparezcan. Los resultados se registrarán y se visualizará la aparición de los animales detectados.



3.1.1. Código

Importación de librerías

```
Ejercicio_CNN.ipynb
Ejercicio_CNN.ipynb -Downloads
Ejercicio_CNN.ipynb > Entrenamos el modelo: Aprende a clasificar imágenes > sport_model_fit(train_X, train_label, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(valid_X, valid_label))
+ Code + Markdown + Run All + Restart + Clear All Outputs + Go To | Variables | Outline ...
Python 3.10.11

Importar Librerías

python3 -m pip install numpy
python3 -m pip install tensorflow
python3 -m pip install scikit-learn
python3 -m pip install matplotlib
python3 -m pip install opencv-python

[3] ✓ 3.1s

Requirement already satisfied: numpy in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (2.0.2)
Requirement already satisfied: tensorflow in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (2.18.0)
Requirement already satisfied: absl-py=1.0.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (2.1.0)
Requirement already satisfied: astunparse=1.0.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers=24.3.25 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast=0.5.0,>=0.5.1,!=0.5.2,!=0.2.1 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta=0.1.1 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang=13.0.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum=2.3.2 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /Users/angelinaquidiazgonzalez/Library/Python/3.10/lib/python/site-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,!=6.0.0dev,!=3.20.3 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/si
Requirement already satisfied: requests=3,!=2.21.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (65.5.0)
Requirement already satisfied: six=1.12.0 in /Users/angelinaquidiazgonzalez/Library/Python/3.10/lib/python/site-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor=1.1.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions=3.6.6 in /Users/angelinaquidiazgonzalez/Library/Python/3.10/lib/python/site-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt=1.11.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio=2.0,!=1.24.3 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (1.68.0)
Requirement already satisfied: tensorflow=2.10,!=2.18 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (2.10.0)
Requirement already satisfied: keras=3.5.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (3.6.0)
Requirement already satisfied: numpy>2.1.0,!=3.26.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>3.11.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (3.22.1)
Requirement already satisfied: ml-dtypes=0.5.0,!=0.4.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem=0.23.1 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (0.37.1)
Requirement already satisfied: wheel=1.0,!=0.23.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (0.45.0)
...
Requirement already satisfied: python-dateutil=2.7 in /Users/angelinaquidiazgonzalez/Library/Python/3.10/lib/python/site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: size=1.5 in /Users/angelinaquidiazgonzalez/Library/Python/3.10/lib/python/site-packages (from python-dateutil=2.7->matplotlib) (1.16.0)
Requirement already satisfied: opencv-python in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (4.10.0.84)
```

```
import numpy as np
import os
import re
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

[2] ✓ 0.7s

import tensorflow.keras
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import Sequential, layers, models

[3] ✓ 2.4s
```

Carga del set de imágenes

```
Cargar set de Imágenes

dirname = os.path.join(os.getcwd(), 'animales')
imgpath = dirname + os.sep

images = []
directories = []
dircount = []
prevRoot=''
cant=0

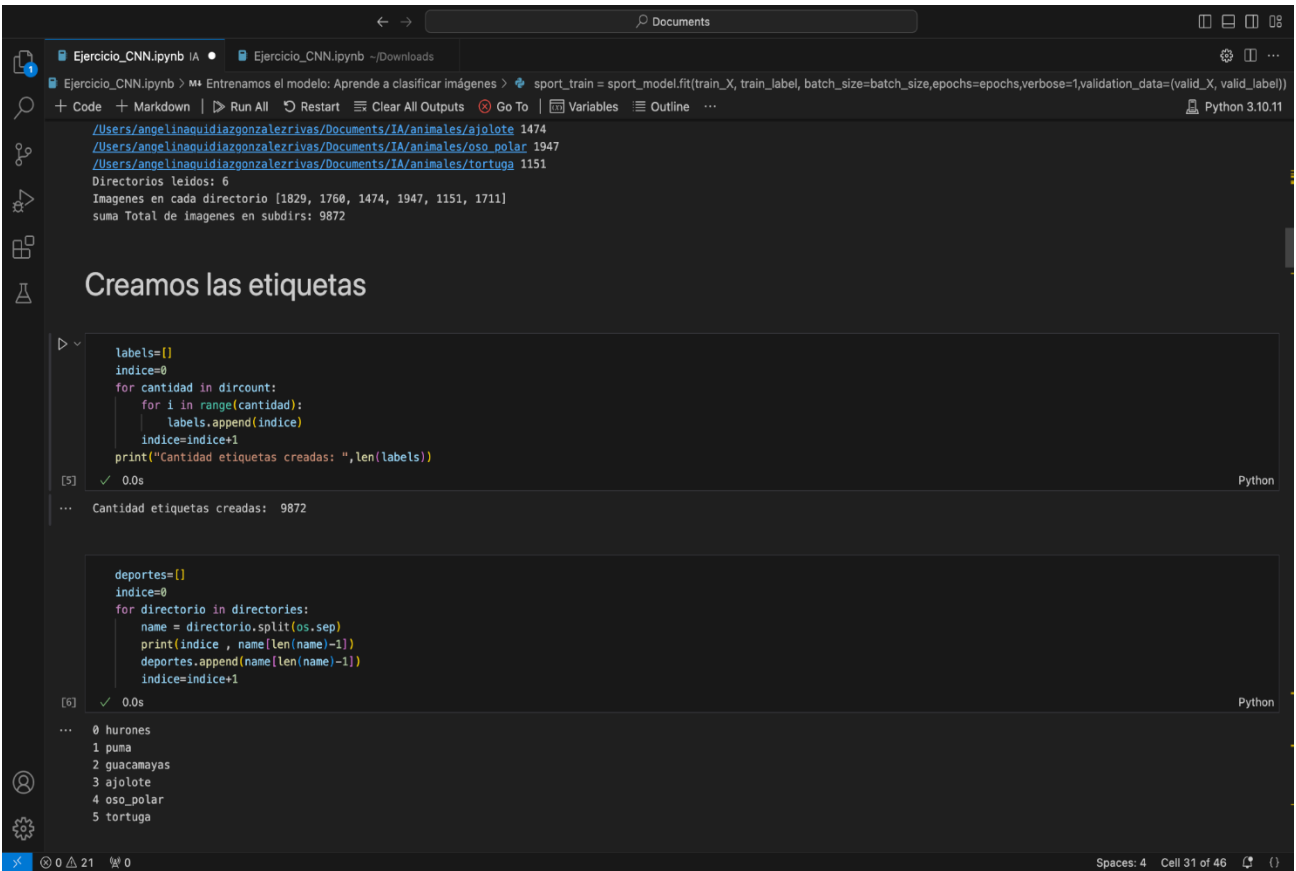
print("leyendo imagenes de ",imgpath)

for root, dirnames, filenames in os.walk(imgpath):
    for filename in filenames:
        if re.search("\.(jpg|jpeg|png|bmp|tiff)$", filename):
            cant=cant+1
            filepath = os.path.join(root, filename)
            image = plt.imread(filepath)
            images.append(image)
            b = "Leyendo..." + str(cant)
            print(b, end="\r")
            if prevRoot !=root:
                print(root, cant)
                prevRoot=root
                directories.append(root)
                dircount.append(cant)
                cant=0
            dircount.append(cant)

dircount = dircount[1:]
dircount[0]=dircount[0]+1
print('Directorios leidos:',len(directories))
print('Imagenes en cada directorio', dircount)
print('suma Total de Imagenes en subdirs:',sum(dircount))

[4] ✓ 1m 4.6s
```

Creación de las Etiquetas



The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar indicates the file is 'Ejercicio_CNN.ipynb' and the Python version is 3.10.11. The notebook content includes a code cell that lists file paths for different animal classes and their image counts, followed by a summary of the total number of images and directories. Below this, a section titled 'Creamos las etiquetas' (We create the labels) contains two code cells. The first code cell creates a list of labels based on the number of images in each directory. The second code cell creates a list of sports (deportes) based on the directory names. The output of the first code cell shows the total number of labels created (9872). The output of the second code cell shows the list of sports: hurones, puma, guacamayas, ajolote, oso_polar, and tortuga.

```
#!/usr/bin/env python3
# Ejercicio_CNN.ipynb
# Entrenamos el modelo: Aprende a clasificar imágenes
sport_train = sport_model.fit(train_X, train_label, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(valid_X, valid_label))

/Users/angelinauidiazgonzalezrivas/Documents/IA/animales/ajolote 1474
/Users/angelinauidiazgonzalezrivas/Documents/IA/animales/oso_polar 1947
/Users/angelinauidiazgonzalezrivas/Documents/IA/animales/tortuga 1151
Directorios leídos: 6
Imágenes en cada directorio [1829, 1760, 1474, 1947, 1151, 1711]
suma Total de imágenes en subdirs: 9872
```

Creamos las etiquetas

```
labels=[]
indice=0
for cantidad in dircount:
    for i in range(cantidad):
        labels.append(indice)
        indice=indice+1
print("Cantidad etiquetas creadas: ", len(labels))
```

[5] ✓ 0.0s Python

... Cantidad etiquetas creadas: 9872

```
deportes=[]
indice=0
for directorio in directories:
    name = directorio.split(os.sep)
    print(indice, name[len(name)-1])
    deportes.append(name[len(name)-1])
    indice=indice+1
```

[6] ✓ 0.0s Python

... 0 hurones
1 puma
2 guacamayas
3 ajolote
4 oso_polar
5 tortuga

Spaces: 4 Cell 31 of 46

Creación de sets de entrenamiento y Test



The screenshot shows a Jupyter Notebook interface with a dark theme. The notebook content includes a code cell that creates a numpy array of labels and a list of images resized to 128x128. Below this, a section titled 'Creamos Sets de Entrenamiento y Test' (We create Training and Test Sets) contains a code cell that splits the training and testing data. The output of the first code cell shows the total number of outputs (6) and the output classes (0 1 2 3 4 5). The output of the second code cell shows the training data shape (7897, 128, 128, 3) and the testing data shape (1975, 128, 128, 3).

```
y = np.array(labels)
import cv2
images_resized = [cv2.resize(img, (128, 128)) for img in images] # Redimensionar a 128x128
X = np.array(images_resized, dtype=np.uint8)
```

```
# Find the unique numbers from the train labels
classes = np.unique(y)
nClasses = len(classes)
print('Total number of outputs : ', nClasses)
print('Output classes : ', classes)
```

[7] ✓ 2m 11.2s Python

... Total number of outputs : 6
Output classes : [0 1 2 3 4 5]

Creamos Sets de Entrenamiento y Test

```
train_X, test_X, train_Y, test_Y = train_test_split(X, y, test_size=0.2)
print('Training data shape : ', train_X.shape, train_Y.shape)
print('Testing data shape : ', test_X.shape, test_Y.shape)
```

[8] ✓ 0.3s Python

... Training data shape : (7897, 128, 128, 3) (7897,)
Testing data shape : (1975, 128, 128, 3) (1975,)

Ejercicio_CNN.ipynb IA • Ejercicio_CNN.ipynb ~/Downloads

Ejercicio_CNN.ipynb > M Entrenamos el modelo: Aprende a clasificar imágenes > sport_train = sport_model.fit(train_X, train_label, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(valid_X, valid_label))

+ Code + Markdown | ▶ Run All ⌂ Restart Clear All Outputs Go To Variables Outline ... Python 3.10.11

```
plt.figure(figsize=[5,5])

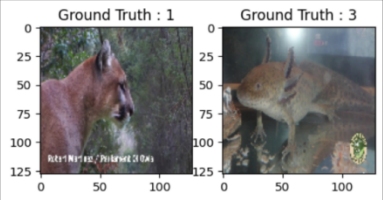
# Display the first image in training data
plt.subplot(121)
plt.imshow(train_X[0,:,:], cmap='gray')
plt.title("Ground Truth : {}".format(train_Y[0]))

# Display the first image in testing data
plt.subplot(122)
plt.imshow(test_X[0,:,:], cmap='gray')
plt.title("Ground Truth : {}".format(test_Y[0]))
```

[9] ✓ 1.5s Python

Text(0.5, 1.0, 'Ground Truth : 3')

...



+ Code + Markdown

Preprocesamos las imágenes

```
train_X = train_X.astype('float32')
test_X = test_X.astype('float32')
train_X = train_X / 255.
test_X = test_X / 255.
```

[10] ✓ 1.8s Python

Spaces: 4 Cell 31 of 46 ()

Ejercicio_CNN.ipynb IA • Ejercicio_CNN.ipynb ~/Downloads

Ejercicio_CNN.ipynb > M Entrenamos el modelo: Aprende a clasificar imágenes > sport_train = sport_model.fit(train_X, train_label, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(valid_X, valid_label))

+ Code + Markdown | ▶ Run All ⌂ Restart Clear All Outputs Go To Variables Outline ... Python 3.10.11

Hacemos el One-hot Encoding para la red

```
# Change the labels from categorical to one-hot encoding
train_Y_one_hot = to_categorical(train_Y)
test_Y_one_hot = to_categorical(test_Y)

# Display the change for category label using one-hot encoding
print('Original label:', train_Y[0])
print('After conversion to one-hot:', train_Y_one_hot[0])
```

[11] ✓ 0.0s Python

Original label: 1
After conversion to one-hot: [0. 1. 0. 0. 0. 0.]

Creamos el Set de Entrenamiento y Validación

```
#Mezclar todo y crear los grupos de entrenamiento y testing
train_X,valid_X,train_label,valid_label = train_test_split(train_X, train_Y_one_hot, test_size=0.2, random_state=13)
```

[12] ✓ 0.7s Python

```
print(train_X.shape,valid_X.shape,train_label.shape,valid_label.shape)
```

[13] ✓ 0.0s Python

(6317, 128, 128, 3) (1580, 128, 128, 3) (6317, 6) (1580, 6)

Creamos el modelo de CNN

```
#declaramos variables con los parámetros de configuración de la red
INIT_LR = 1e-3 # Valor inicial de learning rate. El valor 1e-3 corresponde con 0.001
epochs = 6 # Cantidad de iteraciones completas al conjunto de imágenes de entrenamiento
batch_size = 64 # cantidad de imágenes que se pasan a la vez en cada iteración
```

[14] Python

Spaces: 4 Cell 31 of 46 ()

Creación del modelo de CNN

Ejercicio_CNN.ipynb IA • Ejercicio_CNN.ipynb ~/Downloads

Ejercicio_CNN.ipynb > M4 Entrenamos el modelo: Aprende a clasificar imágenes > sport_train = sport_model.fit(train_X, train_label, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(valid_X, valid_label))

+ Code + Markdown ▶ Run All ↺ Restart ≡ Clear All Outputs ⚠ Go To | Variables Outline ... Python 3.10.11

Creamos el modelo de CNN

```
#declaramos variables con los parámetros de configuración de la red
INIT_LR = 1e-3 # Valor inicial de learning rate. El valor 1e-3 corresponde con 0.001
epochs = 6 # Cantidad de iteraciones completas al conjunto de imágenes de entrenamiento
batch_size = 64 # cantidad de imágenes que se toman a la vez en memoria
```

[14] ✓ 0.0s Python

```
sport_model = Sequential()
sport_model.add(layers.Conv2D(32, kernel_size=(3, 3), activation='linear', padding='same', input_shape=(128, 128, 3)))
sport_model.add(layers.LeakyReLU(alpha=0.1))
sport_model.add(layers.MaxPooling2D((2, 2), padding='same'))
sport_model.add(layers.Dropout(0.5))

sport_model.add(layers.Flatten())
sport_model.add(layers.Dense(32, activation='linear'))
sport_model.add(layers.LeakyReLU(alpha=0.1))
sport_model.add(layers.Dropout(0.5))
sport_model.add(layers.Dense(nClasses, activation='softmax'))
```

[15] ✓ 0.1s Python

... /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim`
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/keras/src/layers/activations/leaky_relu.py:41: UserWarning: Argument `alpha` is deprecated. Use `negati
warnings.warn(

▶ sport_model.summary()

[16] ✓ 0.0s Python

... Model: "sequential"

...

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	896

Spaces: 4 Cell 31 of 46

Ejercicio_CNN.ipynb IA • Ejercicio_CNN.ipynb ~/Downloads

Ejercicio_CNN.ipynb > M4 Entrenamos el modelo: Aprende a clasificar imágenes > sport_train = sport_model.fit(train_X, train_label, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(valid_X, valid_label))

+ Code + Markdown ▶ Run All ↺ Restart ≡ Clear All Outputs ⚠ Go To | Variables Outline ... Python 3.10.11

...

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	896
leaky_re_lu (LeakyReLU)	(None, 128, 128, 32)	0
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
dropout (Dropout)	(None, 64, 64, 32)	0
flatten (Flatten)	(None, 131072)	0
dense (Dense)	(None, 32)	4,194,336
leaky_re_lu_1 (LeakyReLU)	(None, 32)	0
dropout_1 (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 6)	198

... Total params: 4,195,438 (16.00 MB)

... Trainable params: 4,195,438 (16.00 MB)

... Non-trainable params: 0 (0.00 B)

▶ print("Number of images:", len(images))
print("Number of labels:", len(labels))

[17] ✓ 0.0s

... Number of images: 9872
Number of labels: 9872

```
#Asegurar etiquetas enteras
if len(train_label.shape) > 1:
```

Spaces: 4 Cell 31 of 46


```
Ejercicio_CNN.ipynb IA Ejercicio_CNN.ipynb ~/Downloads
IA > Ejercicio_CNN.ipynb > M+ Creamos el modelo de CNN > # Asegurar etiquetas enteras
+ Code + Markdown | ▶ Run All ⌂ Restart ⌂ Clear All Outputs ⌂ Go To | ⌂ Variables ⌂ Outline ... Python 3.10.11

# Asegurar etiquetas enteras
if len(train_label.shape) > 1:
    train_label = np.argmax(train_label, axis=1)
if len(valid_label.shape) > 1:
    valid_label = np.argmax(valid_label, axis=1)

# Configurar el modelo
sport_model.compile(
    optimizer='adam',
    loss=tensorflow.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)

# Verificar dimensiones de validación
print(f"train_X: {train_X.shape}, train_label: {train_label.shape}")
print(f"valid_X: {valid_X.shape}, valid_label: {valid_label.shape}")

# Entrenar el modelo
sport_train = sport_model.fit(
    train_X, train_label,
    batch_size=batch_size,
    epochs=epochs,
    verbose=1,
    validation_data=(valid_X, valid_label)
)

[59] ✓ 2m 42.6s Python

... train_X: (6317, 128, 128, 3), train_label: (6317,)
valid_X: (1580, 128, 128, 3), valid_label: (1580,)
Epoch 1/10
198/198 ————— 17s 83ms/step - accuracy: 0.7146 - loss: 1.1609 - val_accuracy: 0.3494 - val_loss: 80.5951
Epoch 2/10
198/198 ————— 15s 77ms/step - accuracy: 0.8734 - loss: 0.3060 - val_accuracy: 0.1949 - val_loss: 191.6255
Epoch 3/10
198/198 ————— 15s 76ms/step - accuracy: 0.9629 - loss: 0.1159 - val_accuracy: 0.1696 - val_loss: 356.7969
Epoch 4/10
198/198 ————— 15s 77ms/step - accuracy: 0.9877 - loss: 0.0531 - val_accuracy: 0.1791 - val_loss: 255.2773
Epoch 5/10
198/198 ————— 16s 79ms/step - accuracy: 0.9909 - loss: 0.0338 - val_accuracy: 0.1804 - val_loss: 206.0779
Epoch 6/10
198/198 ————— 16s 81ms/step - accuracy: 0.9943 - loss: 0.0244 - val_accuracy: 0.4190 - val_loss: 97.0851
...
Cell 28 of 46 {}
```

```
Ejercicio_CNN.ipynb IA Ejercicio_CNN.ipynb ~/Downloads
IA > Ejercicio_CNN.ipynb > M+ Creamos el modelo de CNN > # Asegurar etiquetas enteras
+ Code + Markdown | ▶ Run All ⌂ Restart ⌂ Clear All Outputs ⌂ Go To | ⌂ Variables ⌂ Outline ... Python 3.10.11

# Entrenar el modelo
sport_train = sport_model.fit(
    train_X, train_label,
    batch_size=batch_size,
    epochs=epochs,
    verbose=1,
    validation_data=(valid_X, valid_label)
)

[59] ✓ 2m 42.6s Python

... train_X: (6317, 128, 128, 3), train_label: (6317,)
valid_X: (1580, 128, 128, 3), valid_label: (1580,)
Epoch 1/10
198/198 ————— 17s 83ms/step - accuracy: 0.7146 - loss: 1.1609 - val_accuracy: 0.3494 - val_loss: 80.5951
Epoch 2/10
198/198 ————— 15s 77ms/step - accuracy: 0.8734 - loss: 0.3060 - val_accuracy: 0.1949 - val_loss: 191.6255
Epoch 3/10
198/198 ————— 15s 76ms/step - accuracy: 0.9629 - loss: 0.1159 - val_accuracy: 0.1696 - val_loss: 356.7969
Epoch 4/10
198/198 ————— 15s 77ms/step - accuracy: 0.9877 - loss: 0.0531 - val_accuracy: 0.1791 - val_loss: 255.2773
Epoch 5/10
198/198 ————— 16s 79ms/step - accuracy: 0.9909 - loss: 0.0338 - val_accuracy: 0.1804 - val_loss: 206.0779
Epoch 6/10
198/198 ————— 16s 81ms/step - accuracy: 0.9943 - loss: 0.0244 - val_accuracy: 0.4190 - val_loss: 97.0851
Epoch 7/10
198/198 ————— 16s 82ms/step - accuracy: 0.9894 - loss: 0.0322 - val_accuracy: 0.3734 - val_loss: 129.9861
Epoch 8/10
198/198 ————— 16s 83ms/step - accuracy: 0.9981 - loss: 0.0139 - val_accuracy: 0.7570 - val_loss: 29.6241
Epoch 9/10
198/198 ————— 17s 80ms/step - accuracy: 0.9913 - loss: 0.0200 - val_accuracy: 0.6608 - val_loss: 50.7945
Epoch 10/10
198/198 ————— 17s 88ms/step - accuracy: 0.9972 - loss: 0.0142 - val_accuracy: 0.7430 - val_loss: 36.9029

sport_model.compile(loss=tensorflow.keras.losses.categorical_crossentropy, optimizer=tensorflow.keras.optimizers.Adagrad(INIT_LR, INIT_LR / 100), metrics=['accu

[50] ✓ 0.0s Python

Cell 28 of 46 {}
```

Evaluación de la red

```
Ejercicio_CNN.ipynb IA • Ejercicio_CNN.ipynb ~/Downloads
IA > Ejercicio_CNN.ipynb > * Creamos el modelo de CNN > # Asegurar etiquetas enteras
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs ⌂ Go To 📄 Variables 📄 Outline ... Python 3.10.11

# guardamos la red, para reutilizarla en el futuro, sin tener que volver a entrenar
sport_model.save("animales.keras")

[ ] Python

Evaluamos la red

# Sincronizar test_X y test_Y_one_hot
min_samples = min(test_X.shape[0], test_Y_one_hot.shape[0])
test_X = test_X[:min_samples]
test_Y_one_hot = test_Y_one_hot[:min_samples]

# Confirmar dimensiones
print("test_X shape:", test_X.shape)
print("test_Y_one_hot shape:", test_Y_one_hot.shape)

# Evaluar el modelo
test_eval = sport_model.evaluate(test_X, test_Y_one_hot, verbose=1)
print("Test evaluation:", test_eval)

test_eval = sport_model.evaluate(test_X, test_Y_one_hot, verbose=1)

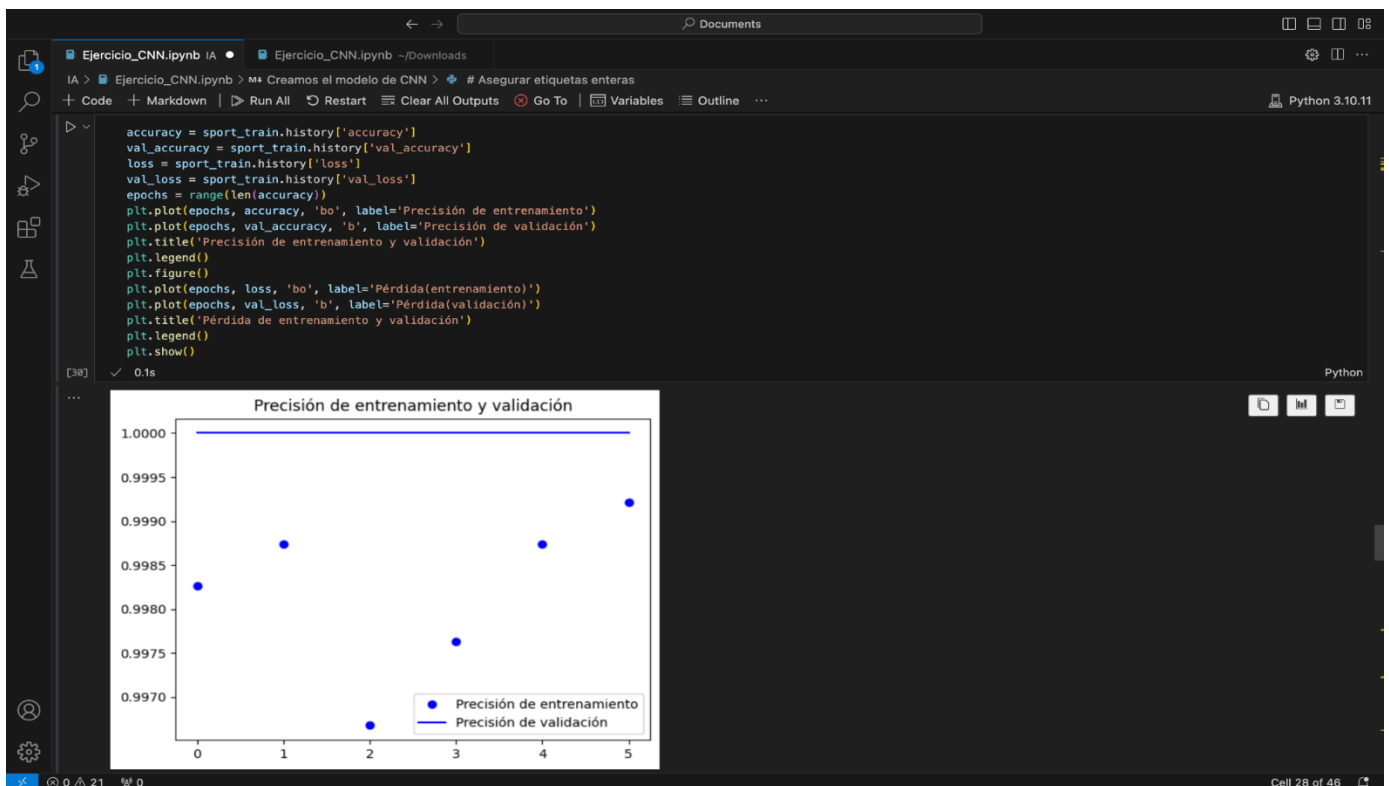
[28] ✓ 3.3s Python

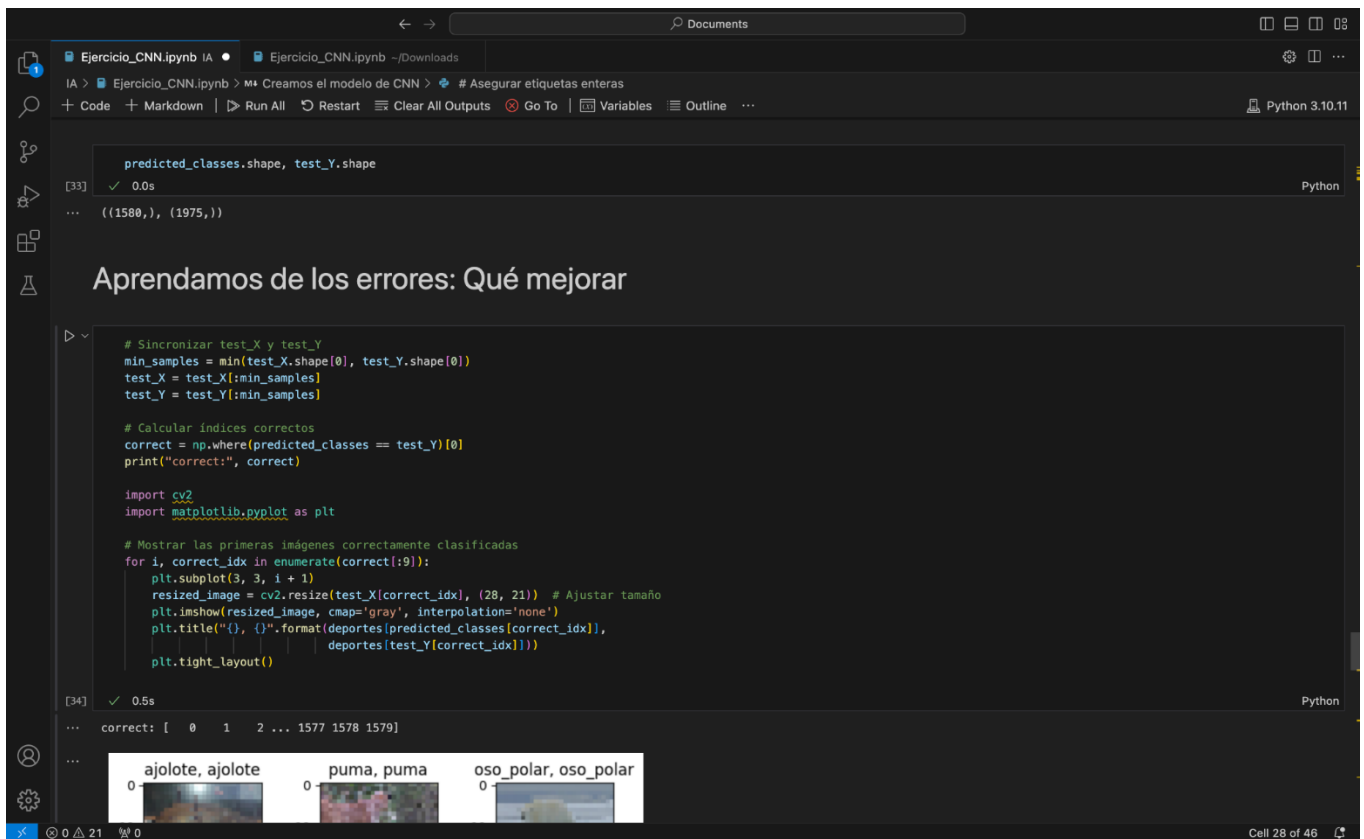
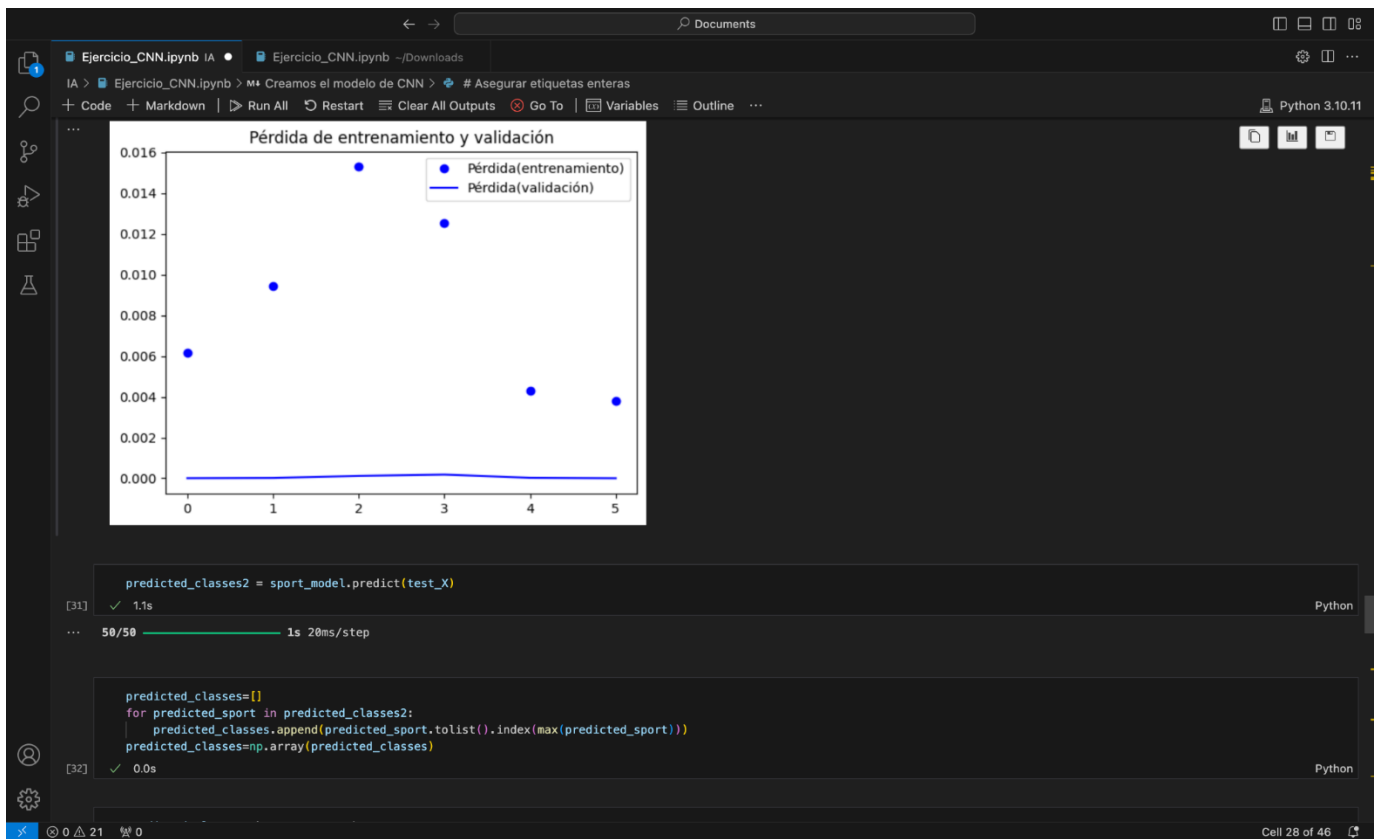
... test_X shape: (1580, 128, 128, 3)
test_Y_one_hot shape: (1580, 6)
50/50 ----- 2s 20ms/step - accuracy: 0.9987 - loss: 0.0936
Test evaluation: [0.0961417704820633, 0.998734176158905]
50/50 ----- 1s 19ms/step - accuracy: 0.9987 - loss: 0.0936

print('Pérdida(Prueba):', test_eval[0])
print('Precisión(Prueba):', test_eval[1])

[29] ✓ 0.0s Python

... Pérdida(Prueba): 0.0961417704820633
Precisión(Prueba): 0.998734176158905
```





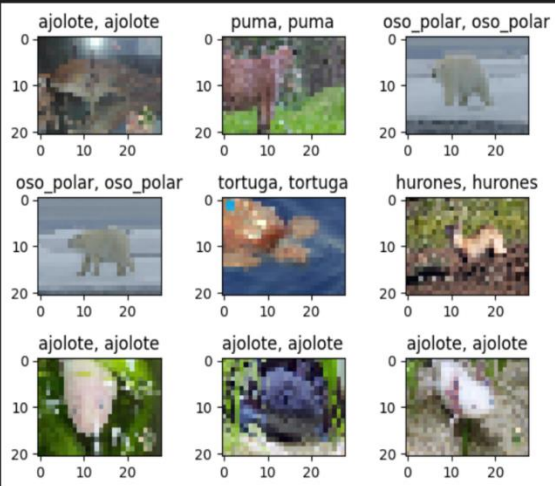
Ejercicio_CNN.ipynb IA • Ejercicio_CNN.ipynb ~/Downloads

IA > Ejercicio_CNN.ipynb > M4 Creamos el modelo de CNN > # Asegurar etiquetas enteras

+ Code + Markdown | Run All | Restart | Clear All Outputs | Go To | Variables | Outline ... Python 3.10.11

```
... correct: [ 0 1 2 ... 1577 1578 1579]
```

...



```
# Sincronizar test_X y test_Y
min_samples = min(test_X.shape[0], test_Y.shape[0])
test_X = test_X[:min_samples]
test_Y = test_Y[:min_samples]

# Calcular índices correctos
incorrect = np.where(predicted_classes != test_Y)[0]
print("incorrect:", incorrect)

import cv2
import matplotlib.pyplot as plt
```

[35] 0.0s

Cell 28 of 46

Ejercicio_CNN.ipynb IA • Ejercicio_CNN.ipynb ~/Downloads

IA > Ejercicio_CNN.ipynb > M4 Creamos el modelo de CNN > # Asegurar etiquetas enteras

+ Code + Markdown | Run All | Restart | Clear All Outputs | Go To | Variables | Outline ... Python 3.10.11

```
# Sincronizar test_X y test_Y
min_samples = min(test_X.shape[0], test_Y.shape[0])
test_X = test_X[:min_samples]
test_Y = test_Y[:min_samples]

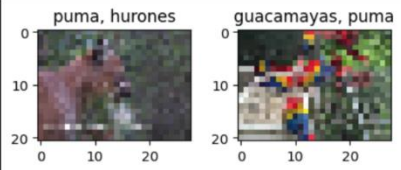
# Calcular índices correctos
incorrect = np.where(predicted_classes != test_Y)[0]
print("incorrect:", incorrect)

import cv2
import matplotlib.pyplot as plt

# Mostrar las primeras imágenes incorrectamente clasificadas
for i, incorrect_idx in enumerate(incorrect[:9]):
    plt.subplot(3, 3, i + 1)
    resized_image = cv2.resize(test_X[incorrect_idx], (28, 21)) # Ajustar tamaño
    plt.imshow(resized_image, cmap='gray', interpolation='none')
    plt.title("{} {}".format(deportes[predicted_classes[incorrect_idx]],
                             deportes[test_Y[incorrect_idx]]))
    plt.tight_layout()

incorrect: [ 273 1164]
```

[35] ✓ 0.0s Python



```
target_names = ["Clase {}".format(i) for i in range(nClasses)]
print(classification_report(test_Y, predicted_classes, target_names=target_names))
```

[36] ✓ 0.0s Python

precision recall f1-score support

Cell 28 of 46

The screenshot shows a Jupyter Notebook interface with two code cells. The first cell contains a line of code to format target names and a call to `print(classification_report)`. The output is a classification report for six classes (Clase 0 to Clase 5) and overall metrics (accuracy, macro avg, weighted avg). The second cell contains a code snippet to list files in the current directory, showing `['.DS_Store', 'huron.jpg']`.

```
target_names = ["Clase {}".format(i) for i in range(nClasses)]
print(classification_report(test_Y, predicted_classes, target_names=target_names))
```

```

...
              precision    recall  f1-score   support

   Clase 0       1.00        1.00        1.00         289
   Clase 1       1.00        1.00        1.00         265
   Clase 2       1.00        1.00        1.00         247
   Clase 3       1.00        1.00        1.00         309
   Clase 4       1.00        1.00        1.00         185
   Clase 5       1.00        1.00        1.00         285

 accuracy       1.00        1.00        1.00        1580
  macro avg       1.00        1.00        1.00        1580
 weighted avg       1.00        1.00        1.00        1580

```

```
import os
print("All files in current directory:", os.listdir('test'))
```

```

...
All files in current directory: ['.DS_Store', 'huron.jpg']

```

Prediccion de una nueva imagen

```
python3 -m pip install scikit-image
from skimage.transform import resize

images=[]
# AQUI ESPECIFICAMOS UNAS IMAGENES
filenames = ['test/huron.jpg']

for filepath in filenames:
    image = plt.imread(filepath,0)
```

3.1.2. Captura de la Ejecución

The screenshot shows a Jupyter Notebook interface with a code cell for image prediction. The code includes installing `scikit-image`, loading an image, resizing it, and using a model to predict classes. The output shows the predicted class for the image 'huron.jpg' as 'guacamayas'. Below the code cell, a detailed list of dependencies and their versions is displayed.

```
python3 -m pip install scikit-image
from skimage.transform import resize

images=[]
# AQUI ESPECIFICAMOS UNAS IMAGENES
filenames = ['test/huron.jpg']

for filepath in filenames:
    image = plt.imread(filepath,0)
    image_resized = resize(image, (128, 128), anti_aliasing=True, clip=False, preserve_range=True)
    images.append(image_resized)

X = np.array(images, dtype=np.uint8) #convierto de lista a numpy
test_X = X.astype('float32')
test_X = test_X / 255.

predicted_classes = sport_model.predict(test_X)

for i, img_tagged in enumerate(predicted_classes):
    print(filenames[i], deportes[img_tagged.tolist().index(max(img_tagged))])
```

```

[54] ✓ 0.6s
...
Requirement already satisfied: scikit-image in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (0.24.0)
Requirement already satisfied: numpy>=1.23 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from scikit-image) (2.0.2)
Requirement already satisfied: scipy>=1.9 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from scikit-image) (1.14.1)
Requirement already satisfied: networkx>=2.8 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from scikit-image) (3.4.2)
Requirement already satisfied: pillow>=9.1 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from scikit-image) (11.0.0)
Requirement already satisfied: imageio>=2.33 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from scikit-image) (2.36.0)
Requirement already satisfied: tifffiles>=2022.8.12 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from scikit-image) (2024.9.20)
Requirement already satisfied: packaging>=21 in /Users/angelinaquidiazgonzalezrivas/Library/Python/3.10/lib/python/site-packages (from scikit-image) (24.2)
Requirement already satisfied: lazy-loader>=0.4 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from scikit-image) (0.4)
1/1 ✓ 0s 11ms/step
test/huron.jpg guacamayas

```

4. Conclusiones

En conclusión, mediante la realización de esta práctica sobre redes neuronales convolucionales se demostró su aplicación en la identificación de animales en peligro de extinción por medio de imágenes extraídas de fotogramas con VLC, empleando técnicas de aprendizaje profundo que permiten a las computadoras reconocer patrones complejos en grandes volúmenes de datos visuales, como lo visto en clases teóricas.

Del mismo modo, La implementación de las redes neuronales convolucionales y el uso de modelos entrenados en imágenes de animales contribuye a la precisión en la detección, mientras que el análisis de resultados proporciona una base para futuras mejoras. Este proceso que se llevo a cabo permite procesar grandes volúmenes de datos visuales de manera rápida y eficiente, demostrando que las CNN son herramientas poderosas en tareas de clasificación y detección.

Finalmente, durante el desarrollo de esta práctica tuvimos algunos problemas en el desarrollo del código, puesto que era algo nuevo para nosotros el uso de esos programas, sin embargo, mediante una correcta organización e investigación pudimos cumplir satisfactoriamente con lo solicitado.

5. Bibliografía / Referencias electrónicas

- [1] “¿Qué es una red neuronal? - Explicación de las redes neuronales artificiales - AWS,” Amazon Web Services, Inc., 2022. <https://aws.amazon.com/es/what-is/neural-network/> (accessed Nov. 11, 2024).
- [2] IBM, “neural networks,” Ibm.com, May 16, 2024. <https://www.ibm.com/mx-es/topics/neural-networks> (accessed Nov. 11, 2024).

[3] desarrolllosidn, "Deep Learning: clasificando imágenes con redes neuronales | LIS Data Solutions," LIS Data Solutions, May 25, 2020. <https://www.lisdatasolutions.com/es/blog/deep-learning-clasificando-imagenes-con-redes-neuronales/> (accessed Nov. 11, 2024).