Curso de Especialización de
Inteligencia Artificial y Big Data (IABD)

Programación de Inteligencia Artificial

UD05. Programación de redes neuronales profundas.
Tarea Online.

JUAN ANTONIO GARCÍA MUELAS

INDICE

Tarea para PIA05

Título de la tarea: Programación de redes neuronales profundas

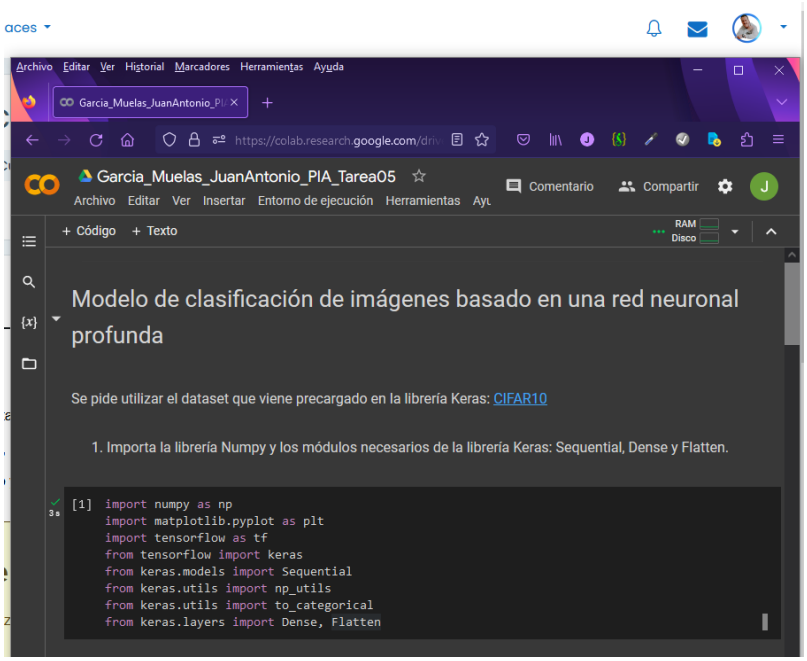
Ciclo formativo y módulo: Curso especialización en Inteligencia Artificial y Big Data - Programación de Inteligencia Artificial

Curso académico: 2022-2023

¿Qué te pedimos que hagas?

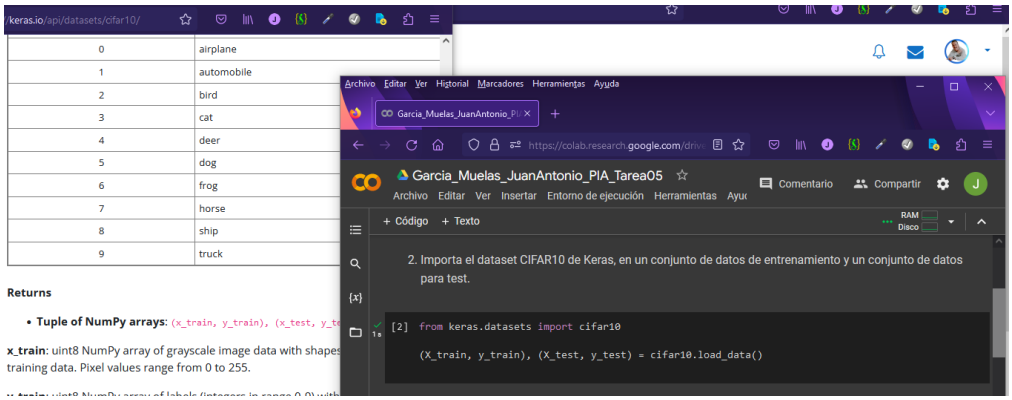
- ✓ **Apartado 1: Carga y explora el dataset CIFAR10**
  - ➢ Inicia un nuevo notebook, preferiblemente en Google Colab. Para guiarte en el proceso, puedes utilizar este cuaderno-guía con algunas sugerencias de fragmentos de código indicados en las celdas de texto, pero tendrás que escribir el código en la celda de código correspondiente y ejecutarlo.
  - ➢ Importa la librerías **Numpy**.
  - ➢ Importa los módulos necesarios para construir una red neuronal profunda: Sequential, Dense y Flatten.

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.utils import np_utils
from keras.utils import to_categorical
from keras.layers import Dense, Flatten
```



- ✓ **Apartado 2: Importa el dataset CIFAR10 de Keras, en un conjunto de datos de entrenamiento y un conjunto de datos para test.**
  - ➢ Consulta la documentación de Keras relativa a este dataset para entender cómo están organizados los datos y saber importarlos.

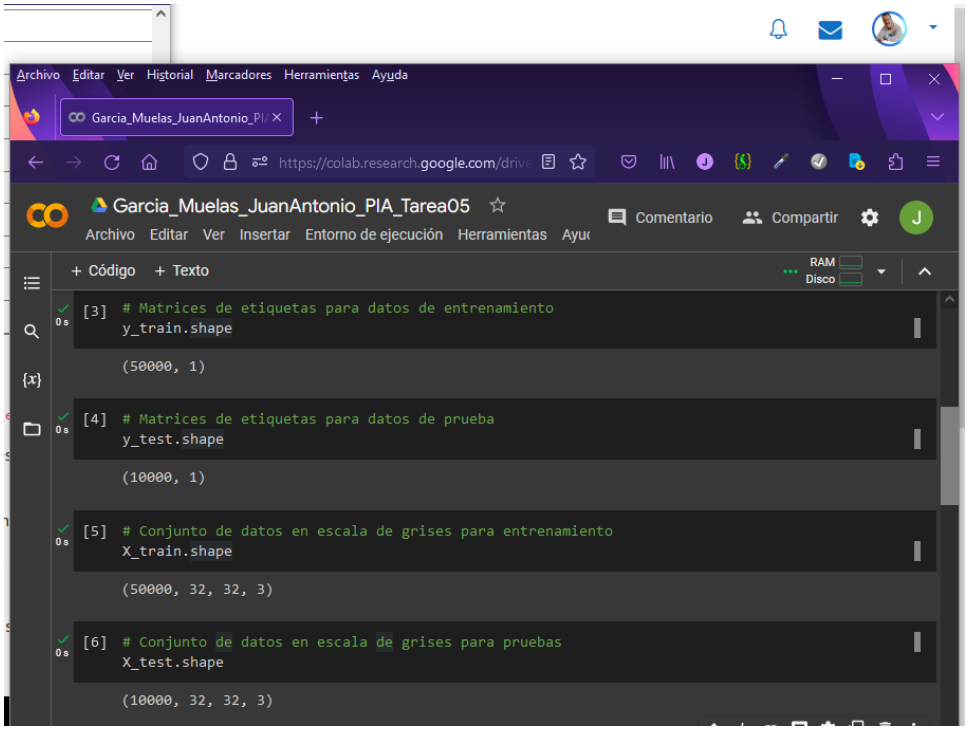
```
from keras.datasets import cifar10
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```



✓ Apartado 3: Explora los datos.

➢ Explora los datos, especialmente, las dimensiones del dataset.

```
# Matrices de etiquetas para datos de entrenamiento
y_train.shape
# Matrices de etiquetas para datos de prueba
y_test.shape
# Conjunto de datos en escala de grises para entrenamiento
X_train.shape
# Conjunto de datos en escala de grises para pruebas
X_test.shape
```



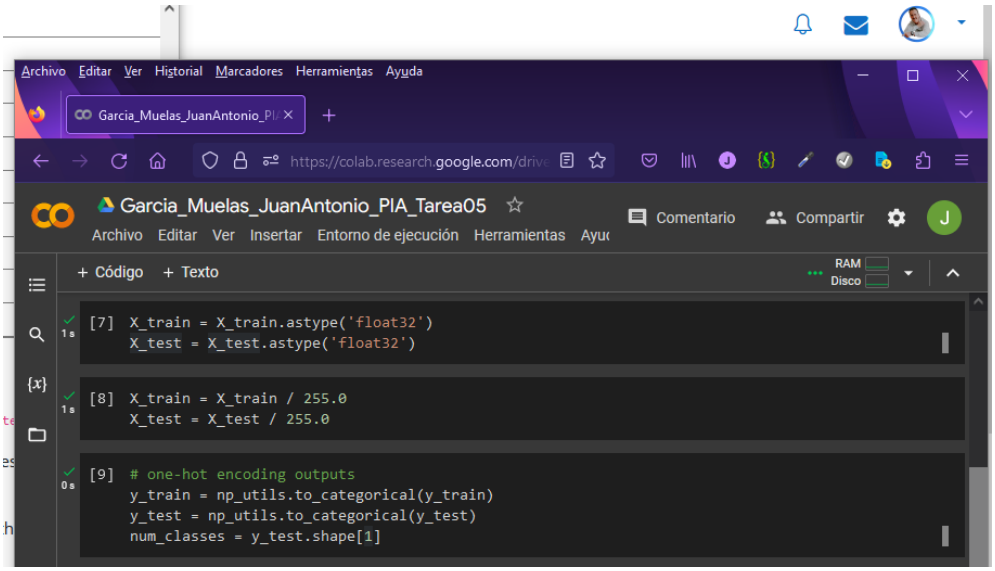
➢ Aplica normalización a los datos de entrada.

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0
```

- Aplica la técnica one-hot encoding al conjunto de datos de salida.

```
# one-hot encoding outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```

- En general, aplica las funciones necesarias para entender cómo son los datos para poder crear el modelo de forma adecuada y entender también los resultados del entrenamiento.

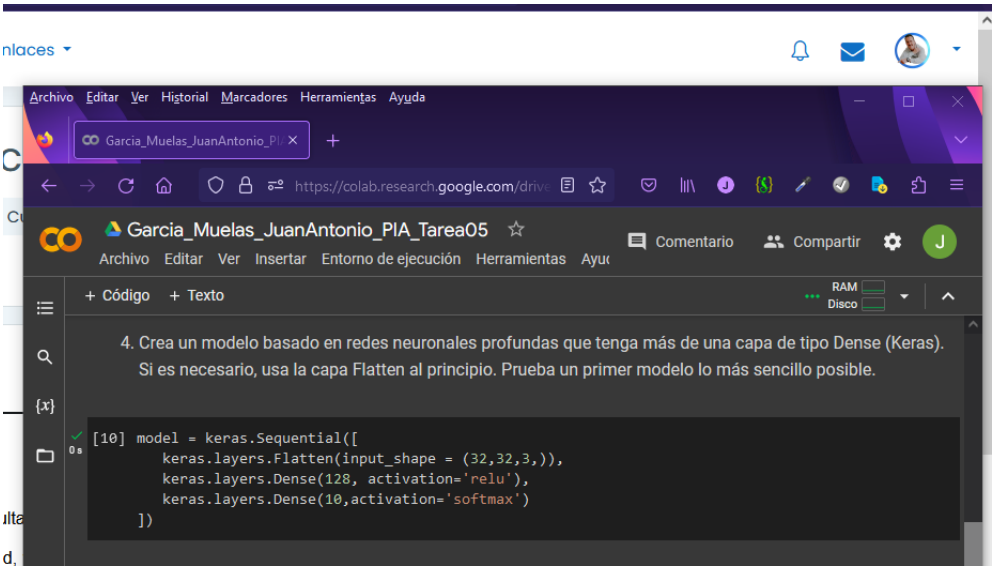


✓ Apartado 4: Crea el modelo.

- Genera un modelo con la clase Sequential.
- Añade el menor número de capas posible, utilizando las clases Dense y Flatten.

Genero un modelo con la capa **`Flatten`**, una capa oculta y una de salida

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape = (32,32,3,)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10,activation='softmax')
   ])
```



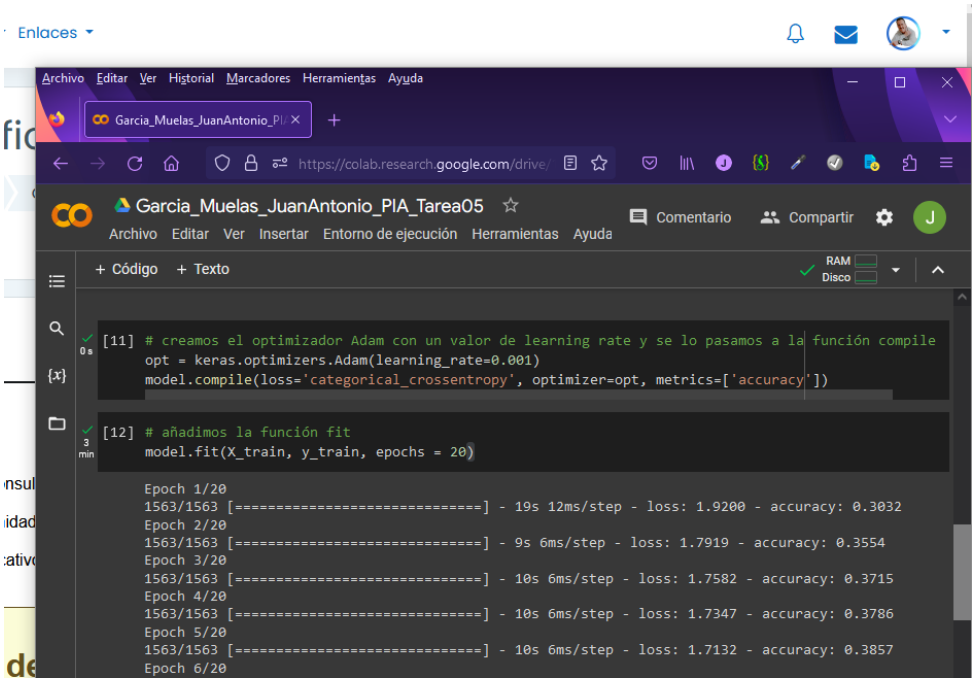
✓ **Apartado 5: Entrena el modelo**.

- Configura el modo de entrenamiento con el método compile.
- Utiliza la función loss = 'categorical_crossentropy'.
- Selecciona el optimizador Adam.
- Utiliza la función fit para entrenar el modelo, con un máximo de 20 epochs.

```
# creamos el optimizador Adam con un valor de learning rate y se lo p
asamos a la función compile

opt = keras.optimizers.Adam(learning_rate=0.001)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics
=['accuracy'])

# añadimos la función fit
model.fit(X_train, y_train, epochs = 20)
```



✓ **Apartado 6: Mejora el modelo**.

- Crea un nuevo modelo con más capas y mayor número de neuronas.

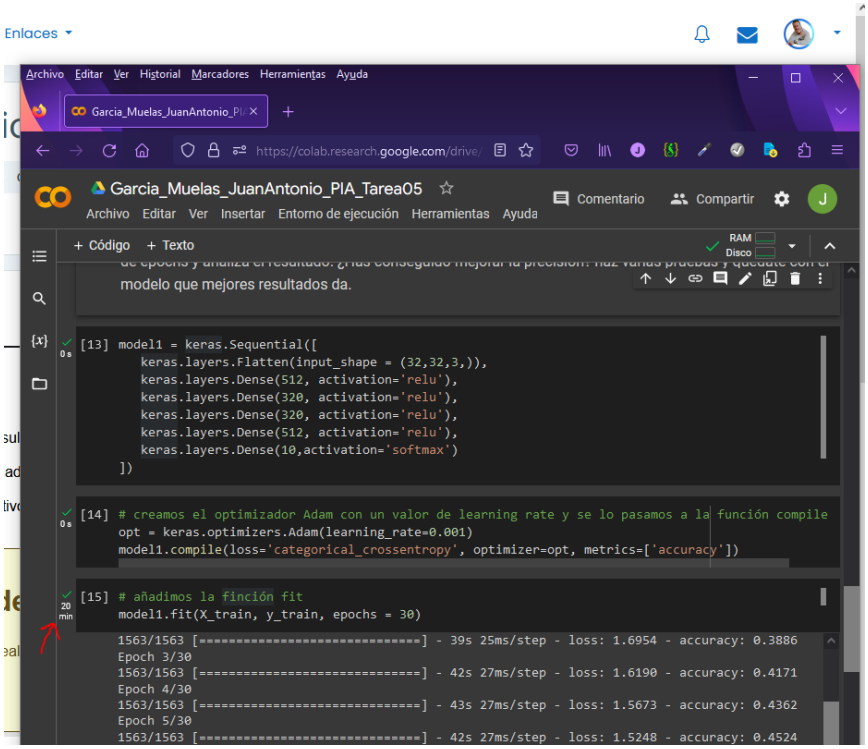
```
model1 = keras.Sequential([
   keras.layers.Flatten(input_shape = (32,32,3,)),
   keras.layers.Dense(512, activation='relu'),
   keras.layers.Dense(320, activation='relu'),
   keras.layers.Dense(320, activation='relu'),
   keras.layers.Dense(512, activation='relu'),
   keras.layers.Dense(10,activation='softmax')
])
```

- Entrénalo utilizando un número mayor de epochs y analiza el resultado.

```
# creamos el optimizador Adam con un valor de learning rate y se lo p
asamos a la función compile
opt = keras.optimizers.Adam(learning_rate=0.001)

model1.compile(loss='categorical_crossentropy', optimizer=opt, metric
s=['accuracy'])
```

```
# añadimos la función fit
model1.fit(X_train, y_train, epochs = 30)
```



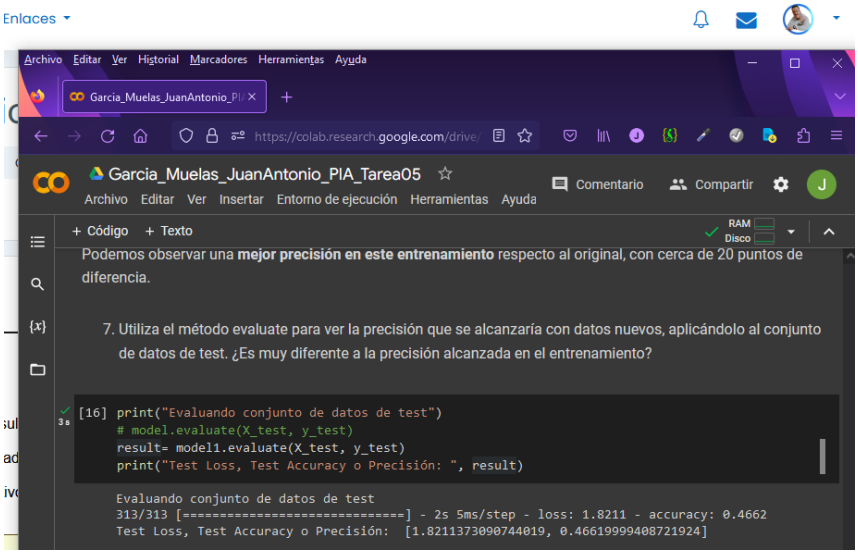
➢ ¿Has conseguido mejorar la precisión? haz varias pruebas y quédate con el modelo que mejores resultados da.

Encontramos una mayor precisión de en torno a 20 puntos respecto al primer intento.

✓ **Apartado 7: Evalúa el nuevo modelo**.

➢ Utiliza el método evaluate para ver la precisión que se alcanzaría con datos nuevos, aplicándolo al conjunto de datos de test.

```
print("Evaluando conjunto de datos de test")
result= model1.evaluate(X_test, y_test)
print("Test Loss, Test Accuracy o Precisión: ", result)
```



- ¿Es muy diferente a la precisión alcanzada en el entrenamiento?

  La precisión baja considerablemente en la evaluación del conjunto de datos.

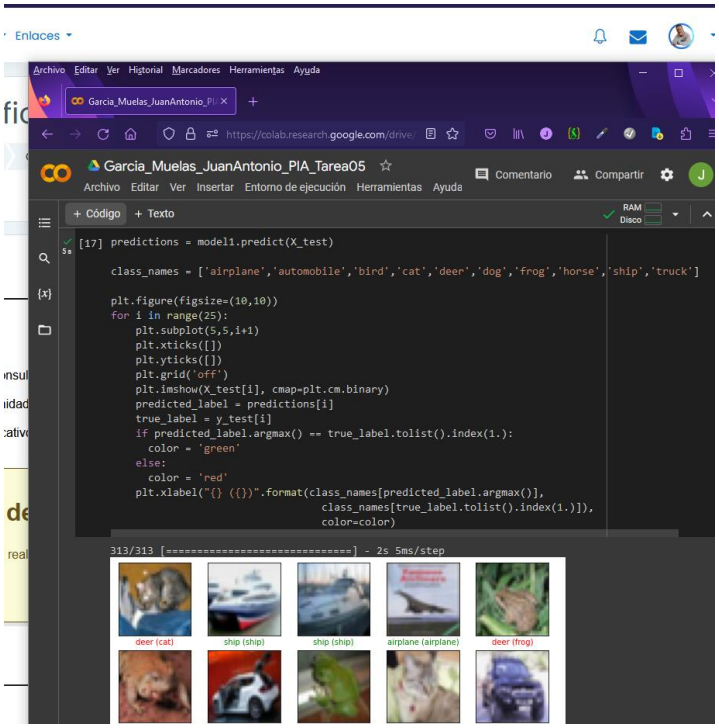
✓ **Apartado 8: Visualiza las predicciones.**

- Explora de forma visual la precisión que se consigue, representando las primeras 25 imágenes del conjunto de datos de test, y comparando la etiqueta real con la de la predicción.
- En la guía tienes un script sugerido para ayudarte con el código.

```
predictions = model1.predict(X_test)

class_names = ['airplane','automobile','bird','cat','deer','dog','fro
g','horse','ship','truck']
plt.figure(figsize=(10,10))

for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid('off')
    plt.imshow(X_test[i], cmap=plt.cm.binary)
    predicted_label = predictions[i]
    true_label = y_test[i]
    if predicted_label.argmax() == true_label.tolist().index(1.):
      color = 'green'
    else:
      color = 'red'
    plt.xlabel("{} ({})".format(class_names[predicted_label.argmax()]
,
                                class_names[true_label.tolist().index(1.)]),
                                color=color)
```



ENLACE al [notebook en colab]