Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Puebla

TC3007C. Inteligencia artificial avanzada para la ciencia de datos
(Grupo 103)

Módulo 2: Implementación de un modelo de deep learning.

Daniel Flores Rodríguez A01734184

4 de Septiembre 2022

Descripción de la situación

Podemos decir que los dinosaurios han sido unas de las criaturas que más nos han impactado en las últimas décadas, han sido tan populares que incluso tienen un lugar en algunas de las grandes industrias, principalmente la de entretenimiento. Conforme se avanza en las investigaciones y descubrimientos de estas criaturas cada dia, el “catálogo” de estos, por así decirlo, empieza a crecer; además de esto, sabemos que muchos de las recreaciones que se creían las verdaderas no lo son y actualmente podemos ver que muchos de los dinosaurios tenían plumas en su cuerpo o simplemente tuvieron características más parecidas a las de un ave de nuestros tiempos. Debido a todo esto y de la mano de la Inteligencia artificial, se ha decidido crear un modelo capaz de identificar a un dinosaurio a través de una imagen o ilustración y así poder lograr más adelante identificar a un dinosaurio y saber si coincide con alguno de las versiones actuales existentes.

Explicación del dataset empleado

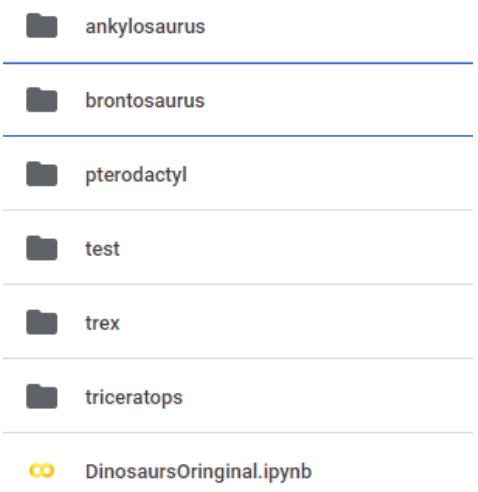
Para este caso, se hizo uso de un set de imágenes de 5 especies distintas de dinosaurios, siendo estas unas de las más conocidas en general, me refiero a el T-rex, Brontosaurio, Pterodactyl, Ankylosaurus y Triceratops. Cada clase o tipo (especie) contiene un conjunto de 40 imágenes, con un total de 200 y 5 clases totales.

Este dataset puede ser encontrado en la página Kaggle, ya que de ahí es donde proviene, a continuación dejare el link a la pagina de donde se extrajo los datasets:

https://www.kaggle.com/datasets/cmglonly/simple-dinosurus-dataset

Configuración del proyecto y manejo de las imágenes

En este caso la estructura y el manejo del dataset se establece de tal manera que las carpetas/clases de las imágenes queden en el mismo nivel del código de python que se realiza, así tenemos el archivo del código y junto a este las carpetas con las imágenes correspondientes.



Ahora bien, para poder trabajar con estas, lo que se realizó fue una clase, con el objetivo de poder transformar y guardar nuestras imágenes, de tal manera que pudiéramos trabajar con librerías de control, como lo son numpy, pandas, etc.

En la misma función se realiza el proceso de extraer la imagen del lugar donde se encuentra a través de la obtención de un ‘path’, es así como terminamos de transformar y redefinir el tamaño de las imágenes en cuestión.

```
# a continuacion se crean los arreglos / matrices en donde guardaremos valores relacionados con las imagenes.

# En la variable x se almacenara la imagen 'descompuesta' de tal manera que podemos trabajar con estas.
X = []

#Se almacenará el indice de las imagenes a manera de posicion y con el nombre de la clase (carpeta) de donde viene dicha/s
Z = []

# Definimos el tamaño de la imagen para poder trabajar y poderles hacer resize mas adelante.
img_size = 225
```

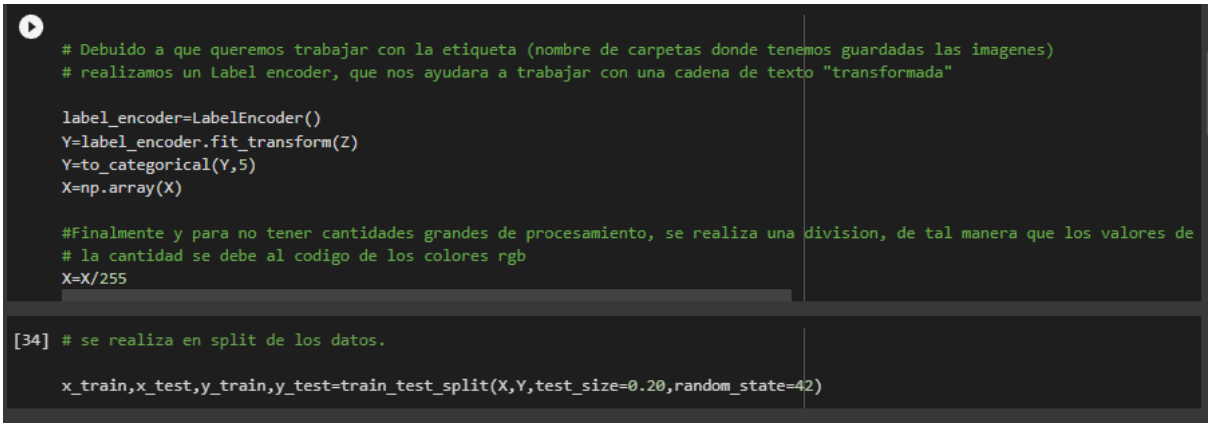
```
[30] #definimos una funcion que nos ayuda a poder convertir la imagen en matriz y ademas creamos el indice de estas.

def train_test_img(especie, PD):
  for img in tqdm(os.listdir(PD)):
    nombre = especie
    path = os.path.join(PD, img) #obtenemos el path del archivo.
    img = cv2.imread(path, cv2.IMREAD_COLOR) #leemos la imagen
    img = cv2.resize(img, (img_size, img_size)) #reajustamos el tamaño de la imagen

    #se van agregando a los arreglos que se declararon anteriormente
    X.append(np.array(img))
    Z.append(str(nombre))
```

```
[31] # en esta parte invocamos la funcion que acabamos de crear y le enviamos sus respectivos parametros.
for dino in dinosaurs:
  dir = './'+dino
  train_test_img(dino, dir)
```

Ya teniendo estos arreglos con la información correspondiente extraída de los archivos de las imágenes, tenemos que seguir transformando ciertas cosas para que podamos entrenar nuestro modelo, con esto, me refiero a la transformación de las etiquetas/nombres de las clasificaciones/especies de dinosaurios. Para esto deberemos aplicar Label Encoding, esto con el objetivo de asignar un valor numérico a los nombres de las especies.

Al término de lo anterior, se realizó una transformación de “escala”, donde básicamente estamos reduciendo valores rgb a un valor menor 1 para poder trabajar de manera más rápida y óptima, esto se debe a que el código rgb solo llega hasta 255, por lo tanto al dividir el número sobre el máximo, obtenemos un valor menor.

Posteriormente, ya que habíamos completado todo lo referente a la limpieza y transformación del dataset, se procedió a realizar el split del test y del train de nuestro dataset, esto con ayuda de la librería de scikit learn que ya hemos ocupado en trabajos anteriores.



```
# Debuido a que queremos trabajar con la etiqueta (nombre de carpetas donde tenemos guardadas las imagenes)
# realizamos un Label encoder, que nos ayudara a trabajar con una cadena de texto "transformada"

label_encoder=LabelEncoder()
Y=label_encoder.fit_transform(Z)
Y=to_categorical(Y,5)
X=np.array(X)

#Finalmente y para no tener cantidades grandes de procesamiento, se realiza una division, de tal manera que los valores de
# la cantidad se debe al codigo de los colores rgb
X=X/255
```

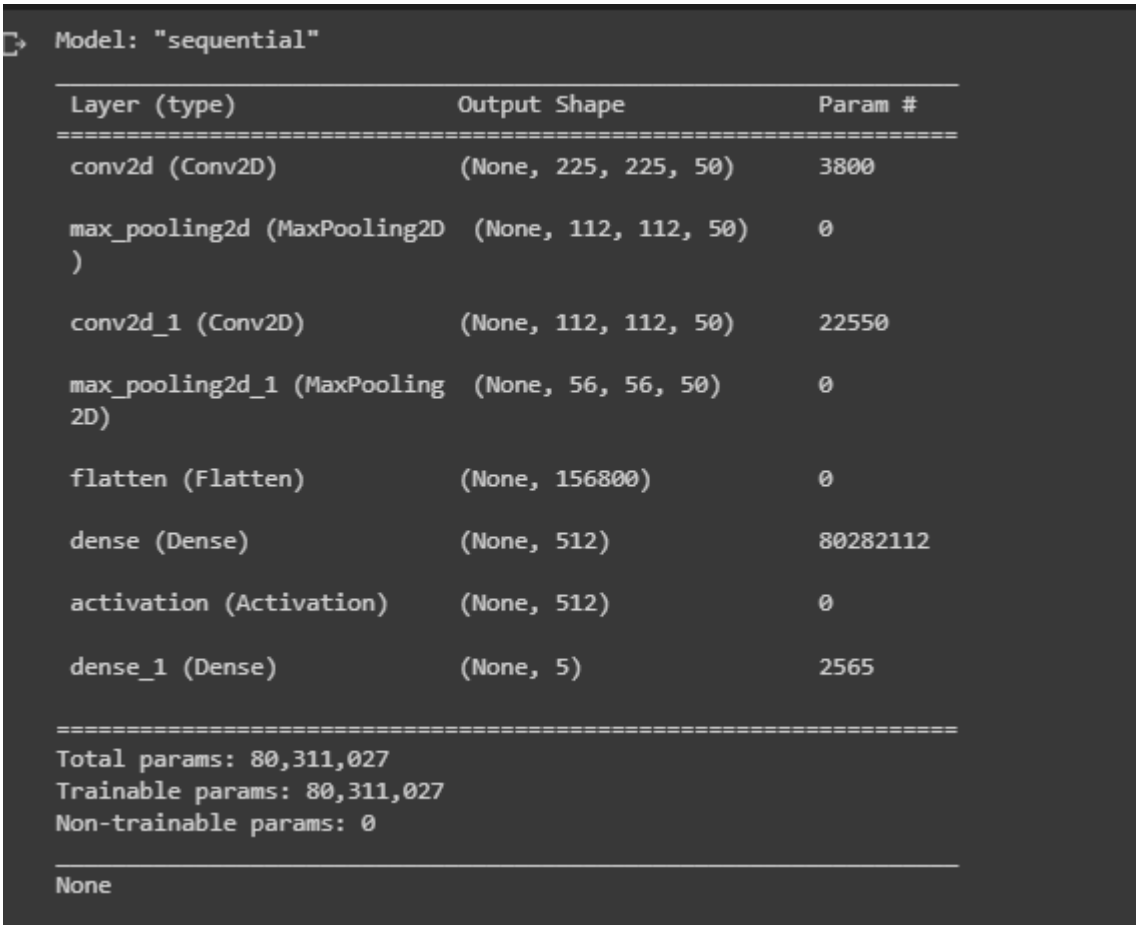
```
[34] # se realiza en split de los datos.

x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,random_state=42)
```

Sobre el modelo usado

El modelo implementado para este caso fue una red neuronal convolutiva también conocida como CNN, podemos decir que es la más adecuada para este problema pues aprende directamente de la información dada, sin necesidad de tener que extraer alguna característica extra.
Son usadas ampliamente para poder encontrar patrones de imágenes, reconocimiento de objetos, caras, formas, etc.
Se podría decir que la base o núcleo de este modelo son las convoluciones, la cual consiste en tomar un conjunto de unidades cercanas entre sí y a partir de estos ir realizando operaciones matemáticas de producto escalar contra una matriz más pequeña conocida como kernel.
Durante la realización del modelo por primera vez lo que se realizó fue armar una red de 2 capas la primera de entrada y otra profunda, algunos de los hiperparametros que podemos encontrar al momento de definir estas capas son, para el caso de las Conv2D, el filtro, se refiere a la cantidad de kernels a emplear en esa capa en específico, para este caso se establecen 50 y 50, por otro lado encontramos el tamaño del kernel, para este caso la dejaremos como 5x5, la funcion de activacion es Relu.

Aunado a esto, después de cada capa, se agrega un MaxPooling2D, la cual es una operación que nos permite analizar el contenido de la imagen en

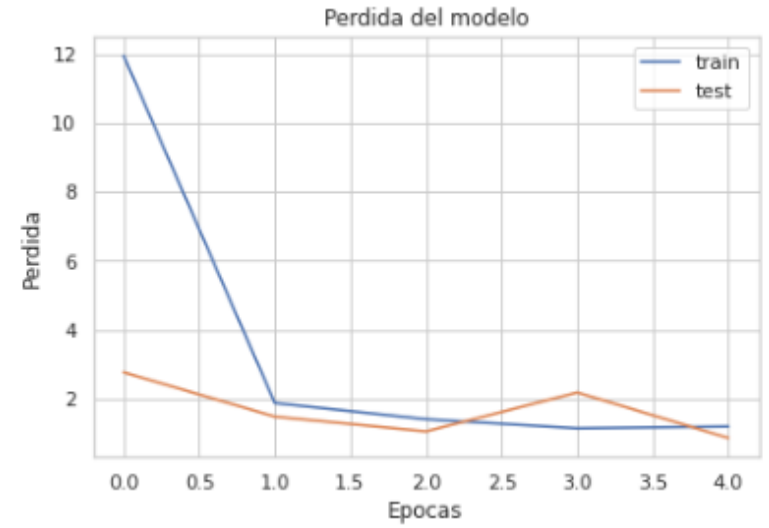
cuestión pero por regiones, se define de 2x2. Finalmente se realiza/aplica una función de flatten, a continuación podemos observar un resumen del modelo después de haberlo compilado.

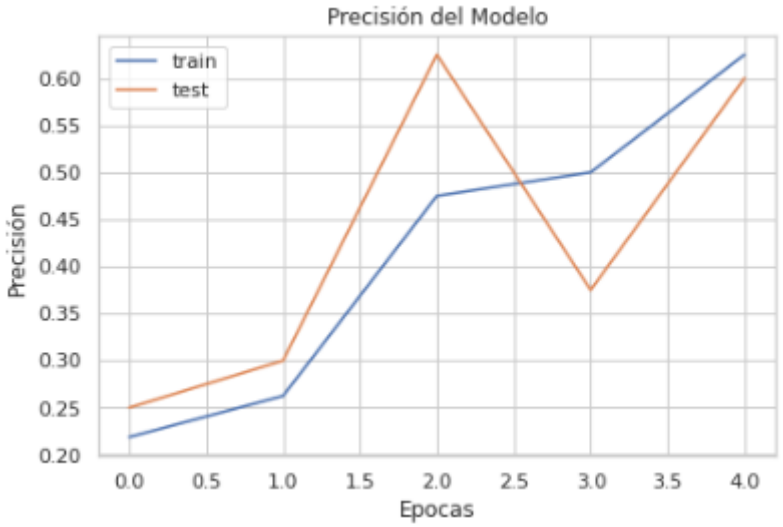


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 225, 225, 50)      3800

 max_pooling2d (MaxPooling2D  (None, 112, 112, 50)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 50)      22550

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 50)       0
 2D)

 flatten (Flatten)           (None, 156800)            0

 dense (Dense)               (None, 512)               80282112

 activation (Activation)     (None, 512)               0

 dense_1 (Dense)             (None, 5)                 2565

=================================================================
Total params: 80,311,027
Trainable params: 80,311,027
Non-trainable params: 0
_________________________________________________________________
None
```

Al tener nuestro modelo, lo que se realiza como paso siguiente es el entrenamiento del modelo, aquí cabe mencionar que dos parámetros importantes: las épocas y el número del batch. Para nuestro caso, se definieron como batch_size: 35 y epochs: 5.

Al momento de realizar el entrenamiento y tras haber tardado unos minutos en cargar, el resultado muestra un comportamiento de overfitting, lo que nos quiere decir que el modelo puede ser mucho mejor para los datos de train, sin embargo en los datos de prueba es donde se ve que falla. Otra característica que podemos notar aquí es que el comportamiento de las gráficas es muy similar en muchas ocasiones, como se puede observar a continuación:





Finalmente podemos decir que el modelo obtuvo un porcentaje de precisión del 60%, un valor algo bajo, en comparación con otros modelos. Así mismo, podemos mencionar la matriz de confusión del modelo respecto a las 5 especias/clases que se tienen.

```
perdida de Modelo 1: 0.8581182360649109
precision de Modelo 1: 0.6000000238418579
```

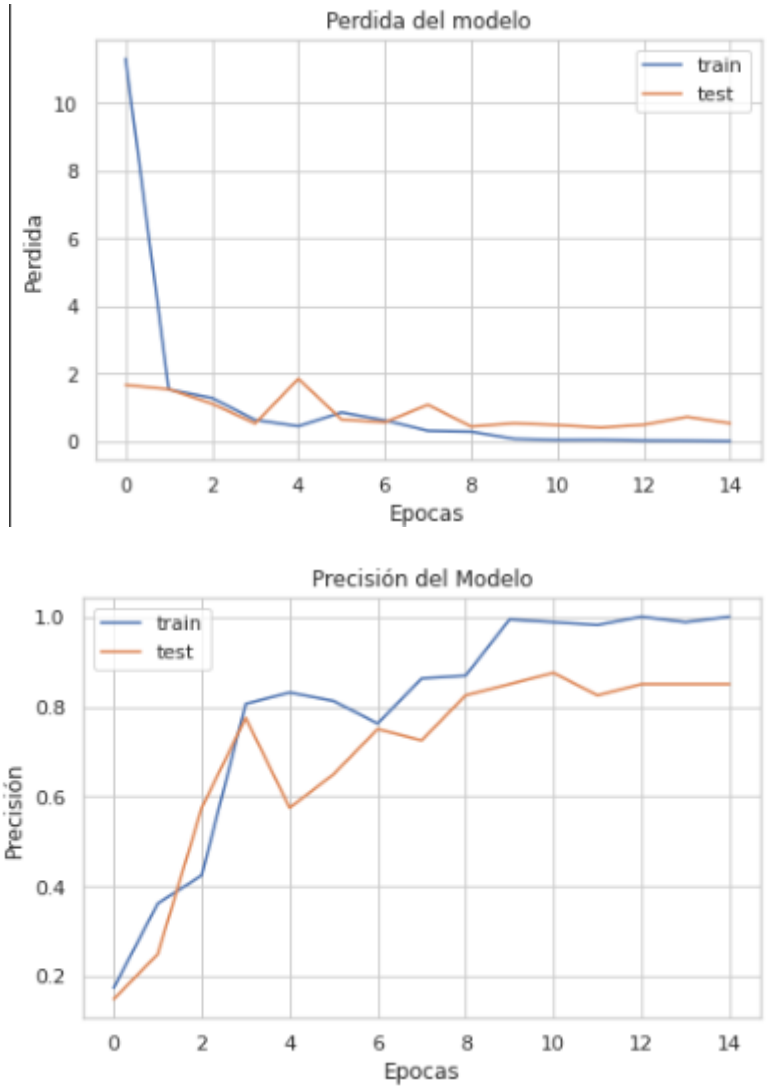
```
array([[5, 0, 0, 0, 0],
       [3, 8, 1, 0, 0],
       [6, 0, 0, 0, 0],
       [0, 1, 0, 9, 0],
       [5, 0, 0, 0, 2]])
```

Segundo modelo

Para el nuevo modelo, el cual se debía mejorar la precisión del modelo realizado anteriormente, se hizo una prueba respecto a la configuración del modelo que depende principalmente en el número de batches, épocas y capas 2D del modelo, quedando de la siguiente manera: batch_size: 30 y epochs: 15. Adicionalmente a esto, se añadieron capas extra al modelo, la que se agregó fue una capa exterior, quedando definidas de tal manera que la primera, es decir la de entrada con 32 kernels, la segunda con 62 kernels y la última con 120. Estos cambios fueron suficientes para obtener un mejor resultado en el entrenamiento, dandonos estos resultados:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 225, 225, 32)      2432

 max_pooling2d (MaxPooling2D  (None, 112, 112, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 128)      0
 2D)

 flatten (Flatten)           (None, 100352)            0

 dense (Dense)               (None, 512)               51380736

 activation (Activation)     (None, 512)               0

 dense_1 (Dense)             (None, 5)                 2565

=================================================================
Total params: 51,478,085
Trainable params: 51,478,085
Non-trainable params: 0
_________________________________________________________________
```

podemos ver el resumen del modelo compilado y listo para ser entrenado, al momento de aplicar la función de fitting, se obtienen las siguientes gráficas:



Claramente podemos observar que las gráficas se comportan de tal manera que el resultado de esta ocasión es overfitting, ya que como se observa las líneas tienen un comportamiento similar, es decir, en la mayoría de las “épocas”, las líneas se mantienen casi al mismo nivel , y solo varían un poco en ocasiones.

Para poder comprobar el mejoramiento del modelo, procedemos a realizar un cálculo del error y la percusión del modelo, así como la matriz de confusión del modelo, la cual se puede interpretar leyendo la diagonal, sabiendo que esos valores son los que se acertaron correctamente, así mismo podemos

observar menos dispersión respecto a las especias/clases que no fueron precedidas de manera correcta respecto al primer modelo.

```
perdida de Modelo 2: 0.5383797883987427
precision de Modelo 2: 0.8500000238418579
```

```
array([[4, 0, 0, 0, 1],
       [1, 8, 3, 0, 0],
       [0, 0, 6, 0, 0],
       [0, 1, 0, 9, 0],
       [0, 0, 0, 0, 7]])
```

Podemos concluir que hubo un cambio notorio entre ambos modelos gracias principalmente a la modificación de los hiperparámetros críticos para el modelo usado, lo que nos dice la importancia que debemos tener al momento de estar modelando y probando Deep Learning.

Link al video demostración:
https://youtu.be/XTMflmaUVvM