IES Pere Maria Orts

Modelos de Inteligencia Artificial

Transferencia del conocimiento

Autor:
Kenny Berrones

Profesor:
David Campoy Miñarro

iesperemariaorts

GENERALITAT
VALENCIANA

Índice

1. Introducción

Transfer Learning es un enfoque de Deep Learning que utiliza un modelo entrenado para una tarea como punto de partida para otro modelo que realiza una tarea similar. Actualizar y volver a entrenar una red con Transfer Learning es más fácil y rápido que entrenarla desde cero. Transfer Learning se utiliza para clasificación de imágenes, detección de objetos, reconocimiento de voz, y otras aplicaciones.

En esta práctica se pretende aplicar está técnica para que a partir de un modelo keras especializado en reconocer coches podamos entrenarlo para reconocer camiones.

2. Experimentos

2.1. Código Original

Con el código que se nos adjunta obtenemos el siguiente resultado:

Figura 1: Resultado del código original para el caso de los coches

Como se aprecia en la imagen anterior, vemos que para el conjunto de pruebas el acierto es de 0.92. Pero al probar con el conjunto de los coches vemos que tiene una precisión un tanto dispar, esto se debe a que hemos entrenado el modelo por solo una época. Antes de pasar a modificar este código vamos a ver los resultados para los casos de no coches:

Figura 2: Resultado del código original para el caso de los no coches

En la figura anterior, vemos que en este caso funciona muy bien debido a que entrenamos con más datos de clases que no son coches.

Además, vamos a ver como se distribuyen los datos en la siguiente imagen:

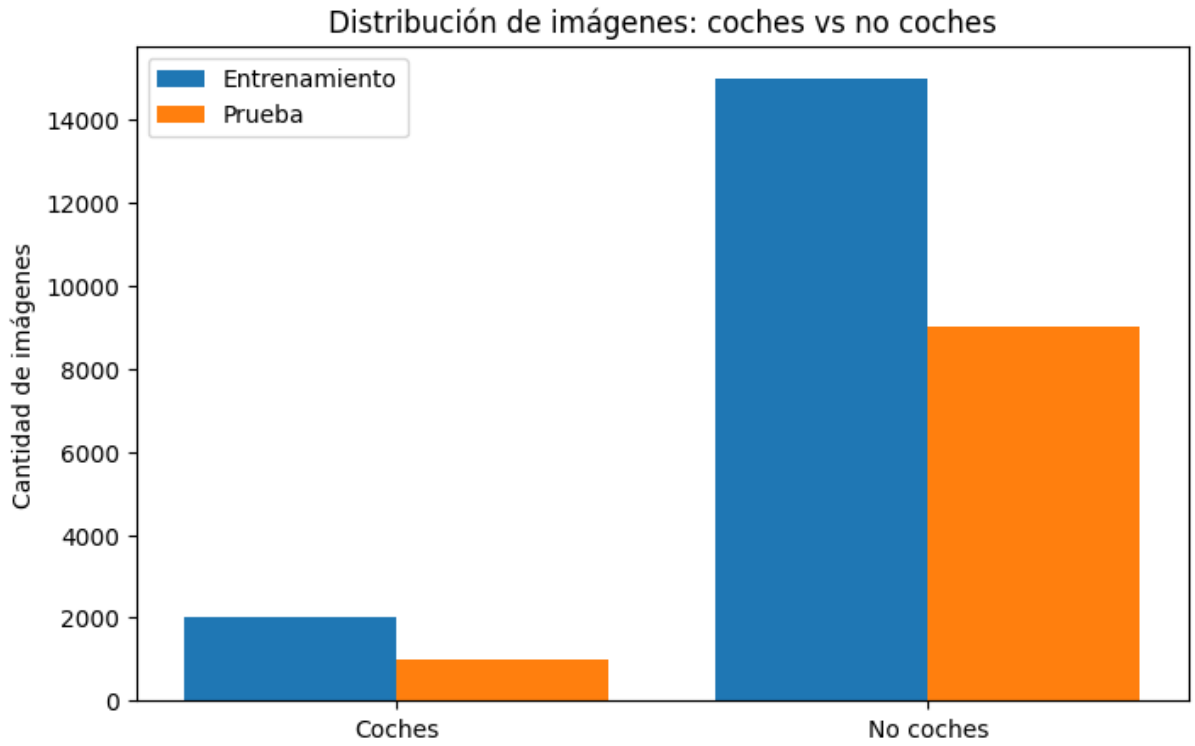

Figura 3: Distribución original de la cantidad de datos

2.2. Cambio en las épocas

Como hemos podido apreciar, este código entrenaba tan solo por una época, lo que hemos realizado es que vamos a aumentar este valor, en la siguiente imagen podemos apreciar como aumenta la tasa de acierto a lo largo de las distintas épocas:

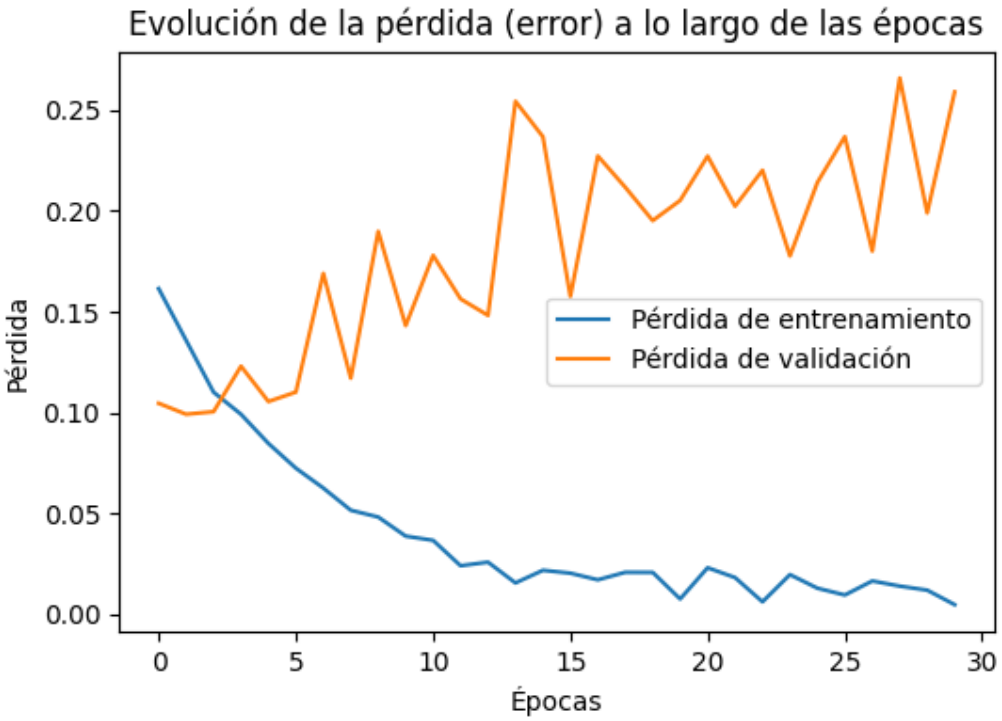

Figura 4: Tasa de error tras entrenar el modelo durante 30 épocas

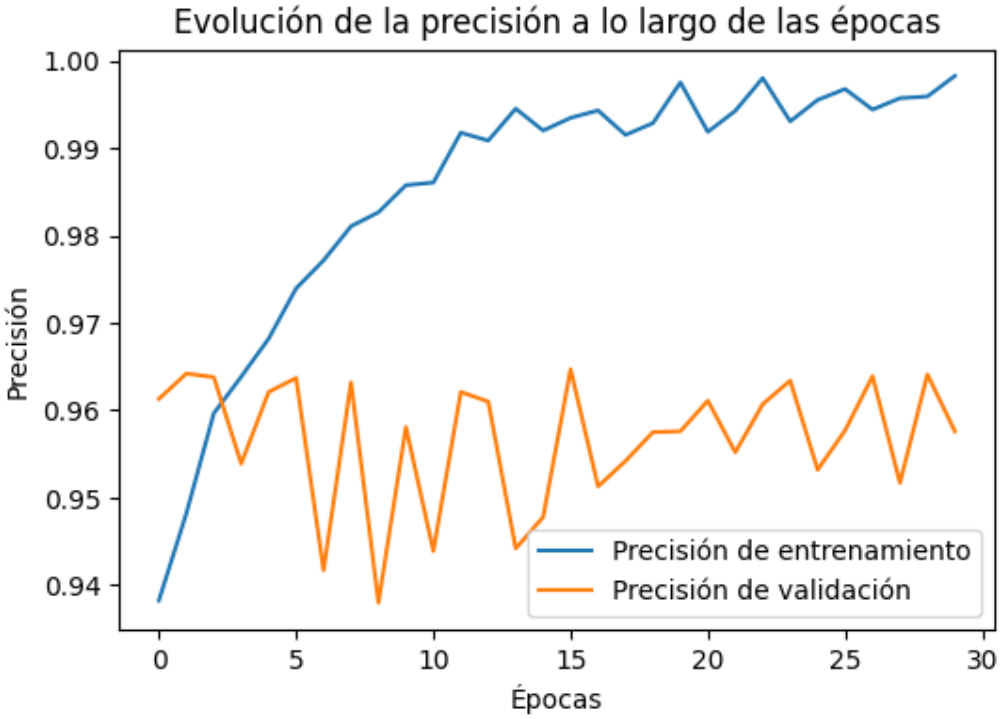

Figura 5: Tasa de acierto tras entrenar el modelo durante 30 épocas

Como se puede apreciar en las imágenes anteriores, la tasa de acierto en el conjunto de entrenamiento va aumentando mientras que la del conjunto de validación fluctúa. Esto tiene que ver con la perdida, si apreciamos la primera gráfica vemos que mientras la

perdida del conjunto de entrenamiento va tendiendo a cero la perdida del conjunto de validación está aumentando, esto se podría deber a muchos factores como podría ser un sobre entrenamiento, muchas épocas, entre otros.

Con este código modificado obtenemos una mejor tasa de acierto para el conjunto de pruebas, concretamente de un 95 %. Además, obtenemos los siguientes resultados:

Figura 6: Resultado del código original para el caso de los coches

Figura 7: Resultado del código original para el caso de los no coches

Vemos que ahora los resultados mejoran sobre todo para los casos de los coches, ya que ahora en la gran mayoría de casos el modelo está seguro al 100 % de que se trata de un coche.

3. Transfer Learning

Ahora procedemos con la técnica de Transfer Learning, aquí partiremos de un modelo ya entrenado, concretamente el modelo que reconoce entre coches y no coches.

Aquí tendremos que sacar las imágenes de coches, camiones y otras clases para poder entrenar. Lo haremos con el siguiente código:

```
# Filtrar imágenes de coches, camiones y otras clases
car_indices = np.where(train_labels == 1)[0][:5000]
truck_indices = np.where(train_labels == 9)[0][:5000]
other_indices = np.where((train_labels != 1) & (train_labels != 9))[0][:10000]
```

Luego las etiquetas las pondremos con un valor para cada clase a entrenar:

```
# Convertir etiquetas: 1 -> coche, 9 -> camión, otras -> otras clases (0)
train_labels = np.array([1 if label == 1 else (2 if label == 9 else 0)
    for label in train_labels])
```

- **0:** Otras clases.
- **1:** Coches.

- **2:** Camiones.

Luego tendremos que congelar todas las capas del modelo excepto la última para que no se reentrenen, lo haremos con este código:

```
# Congelar las capas del modelo base excepto la última capa
for layer in base_model.layers[:-1]:
    layer.trainable = False
```

Tras esto tendremos que crear el nuevo modelo, para esto copiaremos todas las capas del modelo original excepto la última, y además, añadiremos una capa nueva para que el modelo pueda diferenciar entre las 3 clases. Y cambiaremos la función de activación de **sigmoid** a **softmax**, en resumen, usaremos la función sigmoid para clasificación binaria mientras que la función softmax se usará para clasificación multiclase, esto se hace con el siguiente código:

```
model = Sequential(base_model.layers[:-1])
model.add(Dense(3, activation='softmax'))
```

Y a la hora de compilar el modelo también tendremos que cambiar la función de perdida, en el modelo que diferenciaba entre coches y no coches usábamos la función **binary_crossentropy**, ahora tendremos que emplear la función de perdida **categorical_crossentropy**.

Luego tendremos que reentrenar el modelo y ya podremos realizar las predicciones para las 3 clases. Cuando hemos hecho las predicciones hemos obtenido los siguientes resultados:

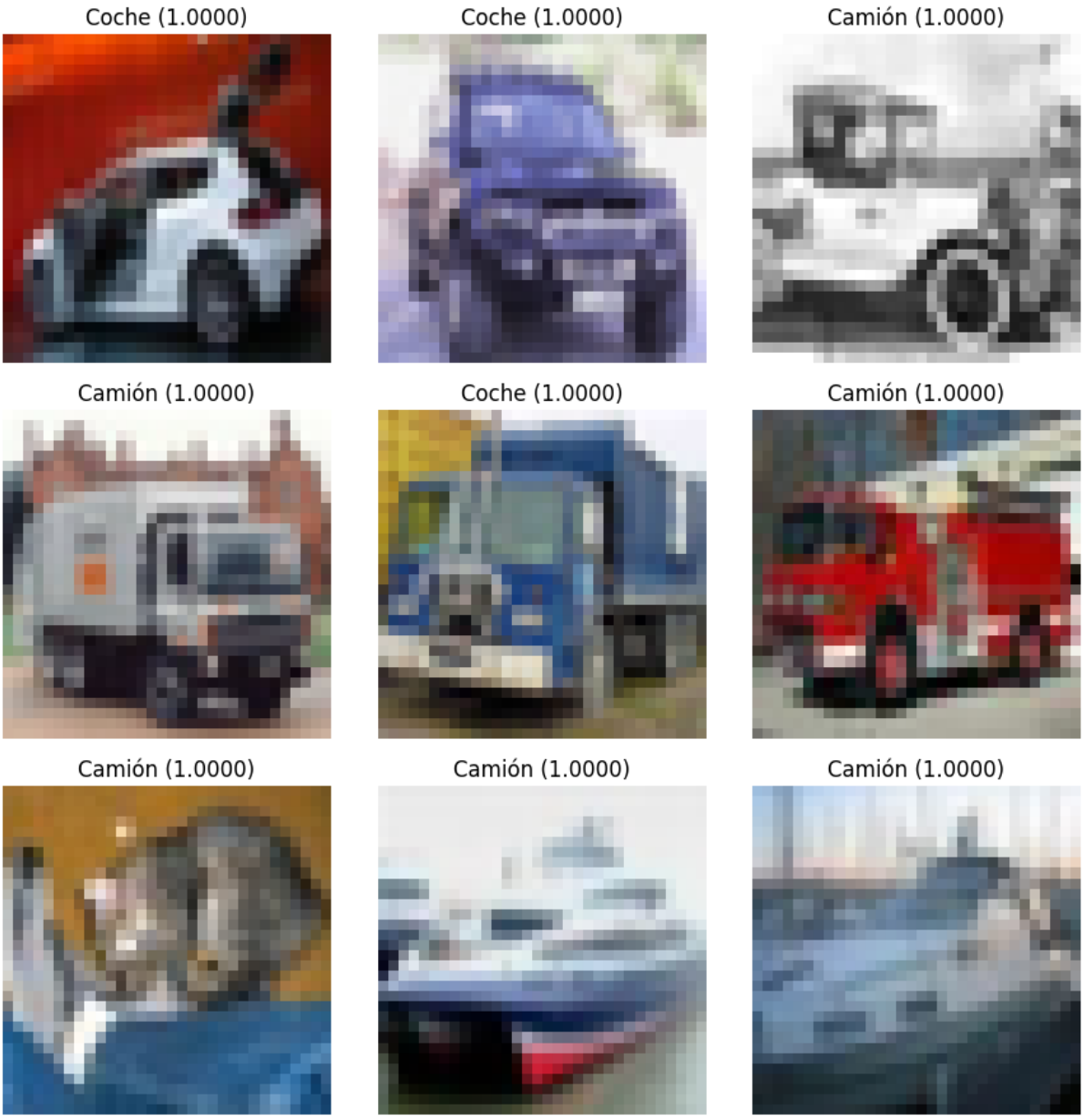

Figura 8: Resultados del modelo que diferencia entre coches, camiones y otras clases

4. Programa que reconozca las 10 clases

Para lograr tener un modelo de IA capaz de reconocer entre las 10 clases podemos realizar dos enfoques, realizar un entrenamiento desde cero para todas las clases del dataset o realizar la técnica de transfer learning a partir del modelo que reconoce coches y no coches o a partir del modelo que reconoce entre coches, camiones y no coches.

Al final la opción que he decidido implementar es realizar un entrenamiento desde cero para las 10 clases, para este caso no hemos tenido que hacer muchos cambios respecto al código original, solo tendremos que transformar las etiquetas originales a one-hot encoding. Además, tendremos que cambiar la última capa de la red neuronal a una neurona con activación softmax para que pueda diferenciar entre las 10 clases. Tras realizar el entrenamiento hemos obtenido estos resultados:

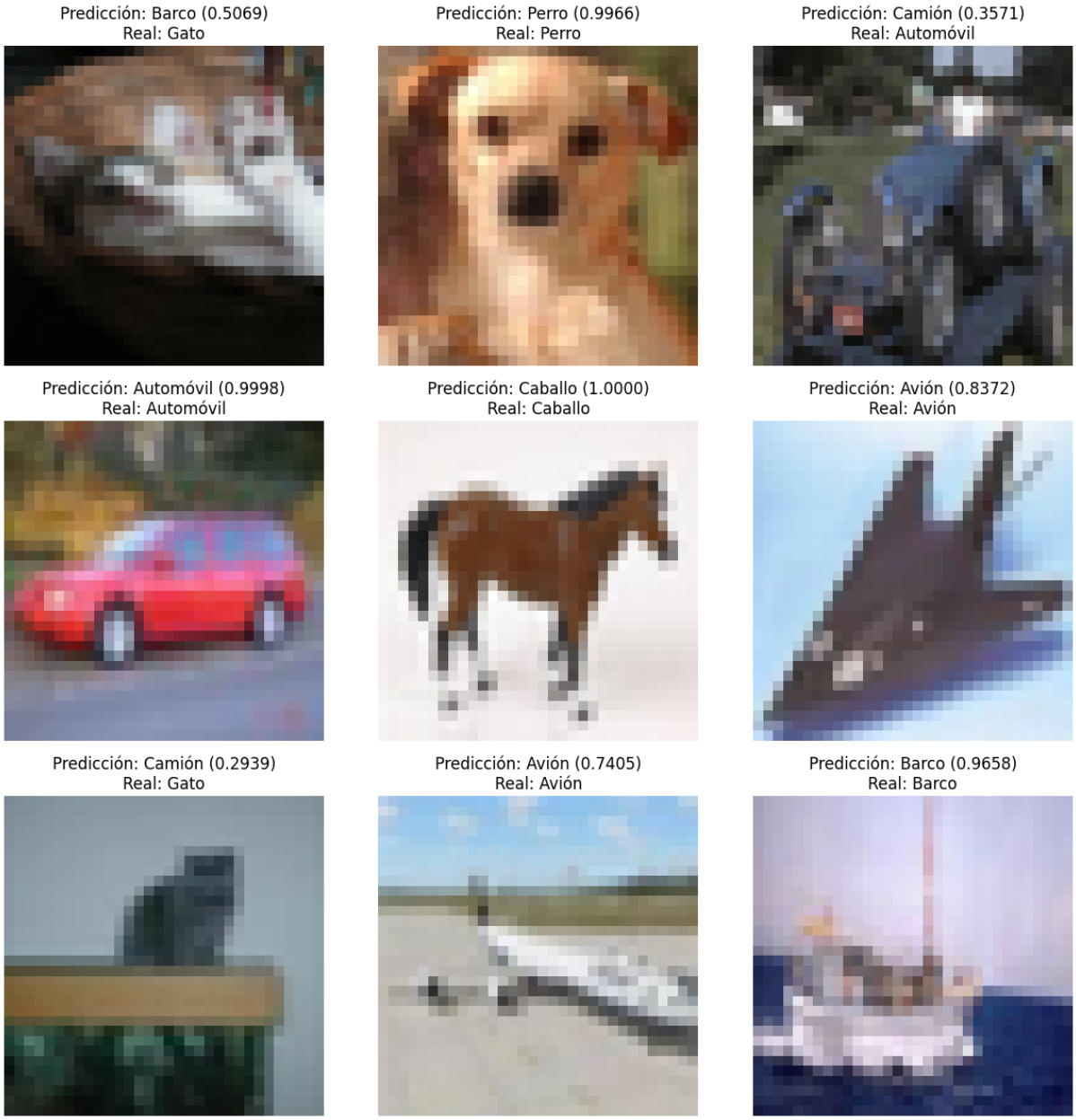

Figura 9: Resultados de las predicciones para las 10 clases

5. Conclusiones

En conclusión, la práctica demuestra que el Transfer Learning es una técnica efectiva para reutilizar un modelo previamente entrenado, facilitando y acelerando el entrenamiento en nuevas tareas sin empezar desde cero. A lo largo de los experimentos, se observó que aumentar las épocas de entrenamiento mejora la precisión, aunque es esencial evitar el sobreajuste para no perder generalización. Las modificaciones al modelo, como ajustar las capas y cambiar la función de activación, permitieron expandir la clasificación de binaria a multiclase con buenos resultados, y el enfoque propuesto muestra que el Transfer Learning es una herramienta útil para escalar modelos a nuevas tareas de clasificación con eficiencia y precisión.