Introducción

El objetivo de este proyecto es implementar una Red Neuronal Convolutiva (CNN, por sus siglas en inglés), para la categorización de imágenes de piezas de LEGO.

Se utilizó un Dataset de 6,414 elementos (imágenes .PNG) divididos en 16 categorías, que corresponden a 16 tipos diferentes de piezas LEGO. La separación de datos para el entrenamiento y prueba del modelo fue de 80%/20%, considerando que 1280 imágenes serán suficientes para validar el correcto funcionamiento del modelo, y aprovechando el resto para un entrenamiento robusto y completo.

Escalamiento de Imágenes

En cuanto a las técnicas de escalamiento, se decidió utilizar las siguientes:

***Zoom*:** Como su nombre lo indica, es la función encargada de hacer zoom a las imágenes, de forma aleatoria dentro del rango establecido del 0 al 30%. Esto sirve para que el modelo no se quede con un solo tamaño de la pieza, y sea capaz de identificarlas sin importar que tan cercana o lejana ha sido tomada la imagen.
***width_shift_range*:** Se refiere al cambio en el rango de ancho de la imagen. En otras palabras, "mueve" la imagen hacia la izquierda o derecha, en este caso dentro de un rango del 0 al 20% de su tamaño hacia cada lado, para que el modelo sea capaz de identificar piezas de LEGO no solo centradas en la imagen, sino ubicadas en distintas zonas de la misma.
height_shift_range: Es la función encargada de "mover" la imagen, al igual que *width_shift_range*, pero de forma horizontal, en un rango del 0 al 20% tanto hacia arriba como hacia abajo. Su objetivo es muy similar, el de ubicar las piezas de LEGO en distintas áreas dentro de la matriz de la imagen, para que el modelo no busque objetos únicamente en el centro de la imagen.
Se decidió no utilizar otras funciones como la ***rotación***, o el ***flip horizontal***, debido a que el dataset se encuentra ya muy completo en cuanto a los ángulos en que fueron tomadas las imágenes, por lo que dichas técnicas serían de poca utilidad al entrenar el modelo.

Es importante mencionar que el escalamiento de imágenes se lleva a cabo como parte del proceso de entrenamiento, con la misma RAM que se utiliza para correr la red. Las imágenes generadas en ningún momento se almacenan en el disco, sino que la función es utilizada únicamente una vez que se comienza a entrenar el modelo.

Red Neuronal Convolutiva

Se utilizó como referencia la implementación propuesta por ***Alex Krizhevsky, Ilya Sutskever y Geoffrey E. Hinton*** en su paper **"ImageNet Classification with Deep Convolutional Neural Networks",** y se adaptó a las capacidades de cómputo con que se disponía para la ejecución de este proyecto, así como las diferencias en dimensiones entre el dataset utilizado para su red *(1.2 millones de imágenes de alta resolución, divididas en 1,000 clases),* y el nuestro. A continuación se describe en detalle el diseño de la red.

Se decidió implementar un ***modelo secuencial***, es decir, una red en la que la salida de una capa se convierte directamente en la entrada de la siguiente. El orden de las capas del modelo es el siguiente:

Input --> Conv2D --> Conv2D --> Conv2D --> Conv2D --> Conv2D --> Flatten --> Dense --> Dense --> --> Dense --> Output

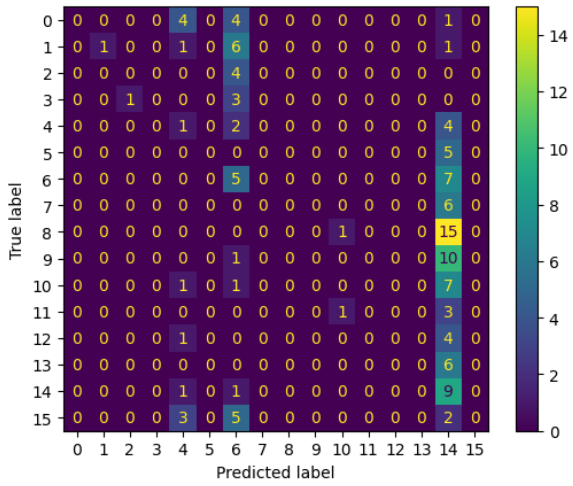
- 5 capas convolutivas de 2 dimensiones, en las que el tamaño de los filtros se va reduciendo, pero la cantidad de filtros a aplicar va aumentando proporcionalmente, y se aplica la función de activación ReLU (Rectified Linear Unit) en las 5.
  La implementación que se utilizó de guía disminuye de igual forma el tamaño de los filtros a aplicar, pero a una escala bastante mayor *(aplicando 96 filtros de 11 x 11 x 3, y llegando hasta 256 filtros de 3 x 3 x 192).* Como se mencionó anteriormente, en nuestro modelo se redujo el tamaño de dichos filtros, y la cantidad de los mismos, para una correcta adaptación al poder de cómputo con que se cuenta para el entrenamiento.
- Una capa "Flatten", para "aplanar", o convertir la salida de la última capa convolutiva 2D, en un vector de una sola dimensión. Hacemos esto para que la siguiente capa (densa) funcione correctamente.
- 1 capa densa de 128 neuronas. Esta capa es también conocida como "completamente conectada" ya que todas las neuronas se conectan entre ellas.
- otra capa densa, pero esta de 64 neuronas.
- Y por último, una capa densa con activación *softmax,* adecuada para la categorización de imágenes, y 16 neuronas, una para cada categoría del dataset con el que se está trabajando.

```
Epoch 1/30
80/80 [==============================] - 692s 9s/step - loss: 2.1565 - acc: 0.2269
Epoch 2/30
80/80 [==============================] - 734s 9s/step - loss: 2.0924 - acc: 0.2562
Epoch 3/30
80/80 [==============================] - 699s 9s/step - loss: 2.0374 - acc: 0.2674
Epoch 4/30
80/80 [==============================] - 702s 9s/step - loss: 1.9550 - acc: 0.2848
Epoch 5/30
80/80 [==============================] - 699s 9s/step - loss: 1.9113 - acc: 0.3012
Epoch 6/30
80/80 [==============================] - 704s 9s/step - loss: 1.8624 - acc: 0.3114
Epoch 7/30
80/80 [==============================] - 698s 9s/step - loss: 1.7889 - acc: 0.3402
Epoch 8/30
80/80 [==============================] - 704s 9s/step - loss: 1.8072 - acc: 0.3443
Epoch 9/30
80/80 [==============================] - 699s 9s/step - loss: 1.7609 - acc: 0.3416
Epoch 10/30
80/80 [==============================] - 697s 9s/step - loss: 1.7581 - acc: 0.3449
```

Testing y Métricas

Luego de haber entrenado el modelo hasta su estancamiento en el 34% de *accuracy* dentro del set de entrenamiento, se procedió a probar con el set de Testing, obteniendo un 9% de accuracy, apenas 3% por encima de lo que se obtendría si se categorizaran de forma aleatoria.

Haciendo un análisis más profundo, buscando el porqué de estos resultados, se obtuvieron datos interesantes, muy útiles para dar los siguientes pasos en el mejoramiento del modelo. Al construir una matriz de confusión con los datos obtenidos en las pruebas, se observó que la gran mayoría de las predicciones que hace el modelo, son de las categorías 14, 6 y 4. Es decir, está “confundiendo” varias categorías de piezas y simplificando su clasificación casi en su totalidad, a estas 3 categorías.



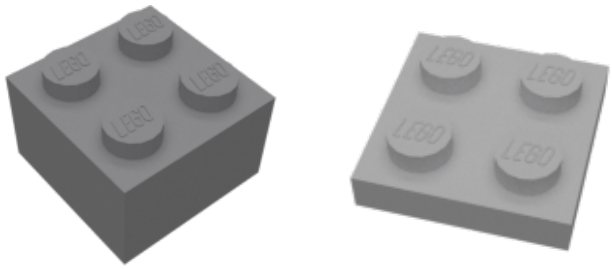
Se podría decir que es un problema de *overfitting*, interpretando que el modelo “memorizó” ciertos patrones. Sin embargo, al tener un porcentaje tan bajo de accuracy en el entrenamiento, es muy poco probable que este sea el caso.

Siguientes Pasos

Para obtener un modelo con un mayor porcentaje de precisión, se proponen dos soluciones:

La primer opción tiene que ver con el dataset. Se propone simplificar categorías dentro de éste, es decir, unir categorías con piezas muy similares, como la 3003 y la 3022, o la 11214 y la 18651. Esto ayudaría a que el modelo no confundiera dichas piezas, y se obtendría así un porcentaje más alto de *accuracy*

3003 (izquierda) y 3022 (derecha)



11214 (izquierda) y 18651 (derecha)

La Segunda opción es Implementar un modelo llamado ***Xception,*** que utiliza redes ya establecidas como punto de partida de su entrenamiento, en lugar de comenzar con valores meramente aleatorios. Existen ejemplos de modelos entrenados para datasets muy similares a los que se están utilizando en este proyecto, con un porcentaje muy alto de precisión, lo que sugiere que se podría obtener algo similar si se decide utilizar para resolver este reto.

Referencias

Dataset - [Joost Hazelzet]. ([2021]). [Images of LEGO Bricks], [Version 4]. Retrieved [May 15th] from [https://www.kaggle.com/datasets/joosthazelzet/lego-brick-images].

Paper de referencia – A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097-1105.

Paper Xception - X. Wu, R. Liu, H. Yang and Z. Chen, "An Xception Based Convolutional Neural Network for Scene Image Classification with Transfer Learning," 2020 2nd International Conference on Information Technology and Computer Application (ITCA), Guangzhou, China, 2020, pp. 262-267.

link a Google Drive del Dataset:
https://drive.google.com/drive/folders/1N3XUCOKy_GJDuUURXu-Upi_2kAdC32ds?usp=sharing

_link al Notebook:
https://colab.research.google.com/drive/1Ot38XblgfjPriJ4eujBNpSJjBmvzH5Om?usp=sharing