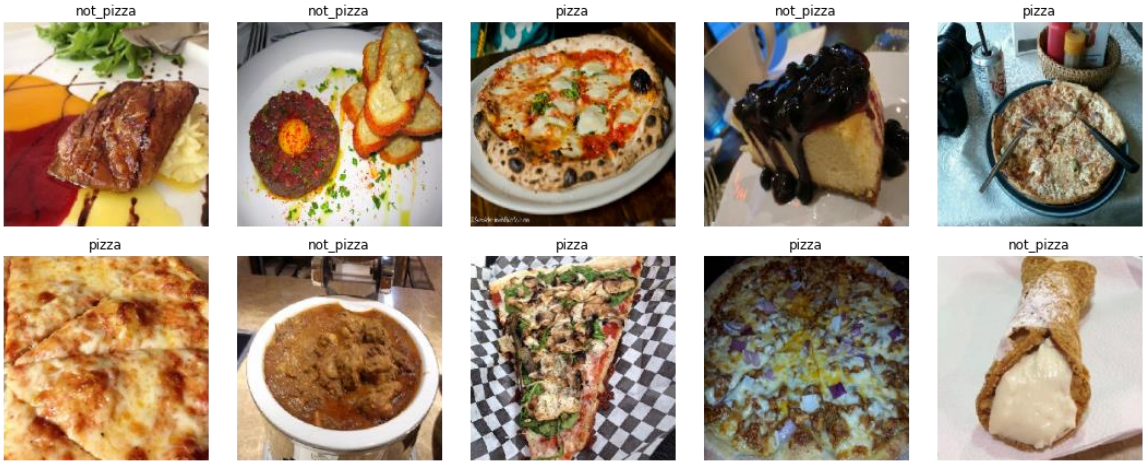
Módulo 2 Implementación de un modelo de deep learning.

Keyuan Zhao A01366831

Contexto del dataset

El dataset fue obtenido en la plataforma de Kaggle, en donde tiene almacenado un total de 1966 de imágenes en formato JPG, las cuales 983 son de pizzas y 983 no son de pizzas.

Para la aplicación de framework se utilizó las librerías de Tensorflow con Keras y algunos modelos de aplicación para procesamiento de capas.



Modelo 1

El primer modelo se usó 4 funciones para procesar las capas, en donde: 2 son de Conv2D para dividir en pequeñas capas de 2D, 1 de Flatten para aplanar las capas en 1D y el último con Dense para conectar las capas.

```
model_1 = Sequential([
    Input(shape = (224, 224, 3)),
    Conv2D(filters = 32, kernel_size = 2, padding = 'valid', activation = 'relu'),
    Conv2D(filters = 32, kernel_size = 2, padding = 'valid', activation = 'relu'),
    Flatten(),
    Dense(1, activation = 'sigmoid')
])
```

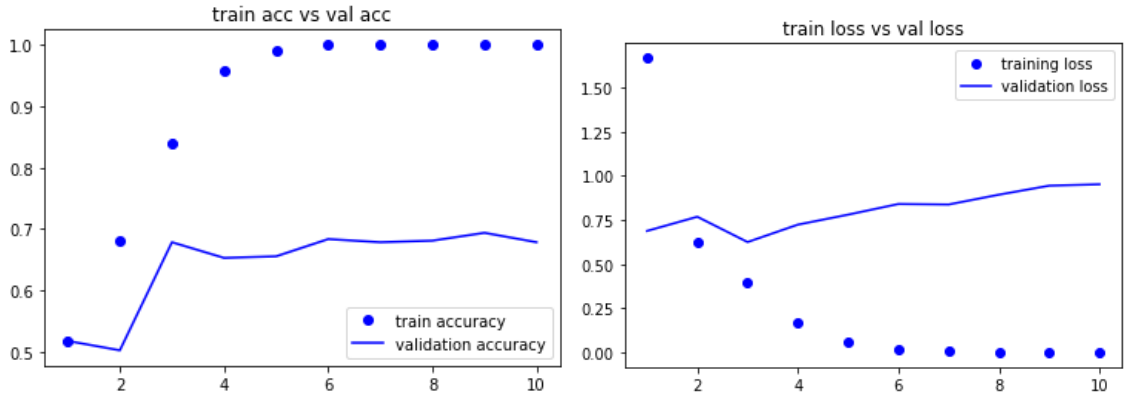
Posteriormente para compilar el primer modelo, se aplicó la función de BinaryCrossentropy como parámetro de loss, ya que es útil cuando queremos predecir para tener respuestas binarias (Sí y No); también se aplicó la función de Adam para la optimización, ya que esta función nos permite ajustar automáticamente el learning reta dependiendo de las veces.

```
model_1.compile(loss = BinaryCrossentropy(),
                optimizer = Adam(),
                metrics = ['accuracy'])
```

Después de compilar se llevará a cabo el entrenamiento del modelo, por lo tanto se aplicó con sólo 10 épocas y la cantidad de pasos por cada época es dependiendo la cantidad de los datos.

```
history_1 = model_1.fit(train_data,
                        epochs= 10,
                        steps_per_epoch = len(train_data),
                        validation_data = val_data,
                        validation_steps = len(val_data))
```

Una vez entrenado el modelo, graficamos la comparación del accuracy y loss de nuestro train y test. En la gráfica de “train acc vs val acc” observamos que nuestro accuracy para train fue favorable, sin embargo, para test no fue favorable ya que sólo tiene cerca de 70% de exactitud de predecir una imagen. La gráfica de “train los vs val loos” observamos que el modelo trian se entrenó correctamente y fue disminuyendo loss conforme las épocas, a cambio para test fue lo contrario.



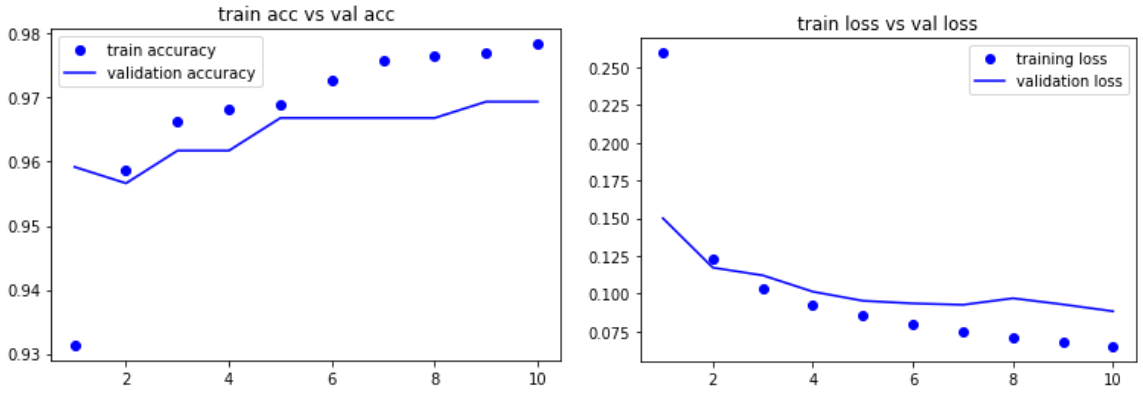
Después de analizar nuestras gráficas, determinamos que el modelo sólo tiene 70% de accuracy para la predicción, eso quiere decir que no es favorable y puede resultar error cuando queremos algo preciso, por lo tanto, debemos de cambiar otro tipo de modelo para que aumente el accuracy.

Modelo 2

Para el segundo modelo se aplicó una arquitectura de Keras llamado EfficientNetB7 y la capa de Dense para conectar las capas.

```
model_2 = Sequential([
    feature_extractor_layer,
    Dense(1, activation = 'sigmoid')
])
```

Los ajustes de compilación y la cantidad de épocas para el entrenamiento fueron las mismas que el modelo 1. Después del entrenar el modelo, graficamos las mismas gráficas de accuracy y loss train y test.



La gráfica “train acc vs val acc” del modelo 2 a comparación del modelo 1 visualmente mejoró bastante, ya que el accuracy de test subió a 97%, por lo tanto, es muy predictivo para nuestro objetivo.

Prueba

Para comprobar si nuestro modelo es efectivo, cargamos unas imágenes para que el modelo determina si la imagen es pizza o no.

El modelo tiene 97% de efectividad, por lo tanto, existe 3% de probabilidad en el que el modelo falle. Así como en la siguiente imagen, el modelo reconoce el pan de muerto vista desde arriba y lo interpreta como pizza. Una forma de solucionarlo y eliminar estos tipos de error es aplicar otra arquitectura y agregar más capas para aumentar la eficiencia del modelo.

Fuente

Link de Kaggle: https://www.kaggle.com/datasets/carlosrunner/pizza-not-pizza