Practica Final Deep Learning

David Pérez Román

July 5, 2025

1 Resumen

En esta practica se pide abordar un problema de visión artificial. Se dispone de un conjunto de imágenes de bandejas que incluyen diferentes comidas y se pide entrenar 3 modelos que sean capaces de determinar el porcentaje de ocupación de comida en las imágenes. Para determinar la calidad de los modelos, se dispone de un baseline RMSE = 4.8.

Para abordar el problema planteado se empleará la estrategia descrita a continuación:

El primer modelo será un modelo simple, sin mejoras en el entrenamiento ni consideraciones para paliar sobreajustes, mientras que el segundo será igual al primero, pero con mejoras aplicadas para mejorar el entrenamiento, evitar sobreajustes, etc.

De este modo, ademas del baseline RMSE propuesto, se podrán comparar los dos modelos para determinar si los cambios realizados han servido para mejorar el modelo o no.

Finalmente, el tercer modelo será un modelo ya existente, el cual se adaptará al problema mediante *Transfer Learning*. El modelo escogido es *DenseNet*, puesto se puede adaptar al problema y porque en *Keras* tiene 3 implementaciones (donde varia la profundidad del modelo), lo que permitirá comparar las 3 implementaciones para escoger la que mejor se adapte al problema (calidad de los resultados, tiempo de entrenamiento, etc).

2 Introducción

El problema que se aborda en este trabajo se trata de un problema de visión artificial.

Se dispone de un conjunto de imágenes, que contiene un total de 1027 imágenes de bandejas que incluyen diferentes comidas, con una dimension espacial de 3264 x 2448 pixeles en RGB. Para cada una de las imágenes, se ha calculado el porcentaje de pixeles que contienen comida. Partiendo de estos pares (imagen, porcentaje), se pide definir y entrenar modelos basados en redes convolucionales (CNNs) que, dada una imagen, proporcionen el porcentaje de pixeles de comida de la misma.

La calidad de los modelos se estimará utilizando el error cuadrático medio (MSE) o la raíz del error cuadrático medio (RMSE). En concreto, se implementaran tres aproximaciones que se compararan con un baseline cuyo RMSE = 4.8.

3 Métodos

Para abordar el problema se implementarán 3 modelos, los cuales se compararán con el baseline propuesto y entre ellos para determinar que modelo se adapta mejor al problema y obtiene mejores resultados.

La estructura de los modelos será la siguiente:

- Modelo 1: Modelo simple, cuyo propósito es servir de base para hacer mejoras y poder observar el efecto de estas mejoras en el problema a tratar.
- Modelo 2: Re-implementación del primer modelo donde, tras analizar los resultados del Modelo 1, se aplicarán mejoras sobre el modelo y en el proceso de entrenamiento.
- Modelo 3: Implementación del modelo DenseNet empleando *Transfer learning*, como la librería *Keras* dispone de 3 implementaciones de DenseNet, donde la diferencia entre ellas radica en la profundidad del modelo, se implementarán las 3 y se escogerá la que mejor se adapte al problema, basando la decisión en la calidad de los resultados, velocidad de entrenamiento, posible sobreajuste, etc.

3.1 Preprocesado de los datos

Antes de definir los modelos y empezar el entrenamiento, se cargan los datos al Notebook y se preprocesan.

Se dividen los datos en particiones train/validation/test y se preprocesan los datos, donde simplemente se normalizan los coeficientes RGB de las imágenes a una escala [0,1].

También se define el tamaño de los batches en 16. Sería posible subirlo a 32, pero es posible que durante el entrenamiento se alcance el limite de memoria RAM disponible en el backend de Google y que se aborte la ejecución, por lo tanto se decide el tamaño del batch en 16.

Finalmente se crean los DataGen de Train, Validation y Test.

3.2 Modelo 1

El modelo 1 es un modelo simple con 5 bloques (2 capas convolucionales con activación ReLu y 1 MaxPooling), siendo el primero el de entrada, seguidos de 1 capa completamente conectada con 1024 neuronas y con activación ReLu y, finalmente, una capa de salida con una única neurona y sin función de activación.

Las capas convolucionales iniciales tienen un numero bajo de filtros, mientras que las capas convolucionales mas profundas tienen un numero de filtros mas elevado, de este modo las primeras capas convolucionales actúan como extractores de características básicas, y las capas más profundas combinan estas características para formar representaciones más complejas.

Se emplea el optimizador Adam en el entrenamiento con un Learning Rate = 0.0001 y se entrena el modelo durante 10 Epochs (este numero de Epochs se mantendrá para los demás modelos).

3.3 Modelo 2

El modelo 2 toma como base el Modelo 1 y se aplican modificaciones para mejorar los resultados obtenidos previamente.

Primero, se decide simplificar el modelo eliminando 1 bloque (2 capas Conv2d y una MaxPooling), la razón es que se pretende que al simplificar el modelo, sea mas difícil que este se sobreajuste al problema, de este modo se debería poder obtener un RMSE en validación mas cercano al de entre-

namiento, también se substituye la capa Flatten por una capa GlobalAveragePooling2D.

Segundo, se añade una capa completamente conectada antes de la salida y se modifica el numero de neuronas de la ya existente para pasar de 1 capa con 1024 neurona a 2 capas con 512 neuronas. Con este cambio se pretende conseguir una mayor capacidad de representación del modelo.

Finalmente, se aplican cambios en el proceso de aprendizaje. Se añade Learning Rate Decay (*ReduceLROnPlateau*) para mejorar el proceso de entrenamiento. Adicionalmente, se implementa early stopping para que el entrenamiento se detenga si el modelo empieza a sobreajustar.

3.4 Modelo 3

El modelo 3 se escogerá de entre las 3 implementaciones de DenseNet. Las tres implementaciones se entrenarán bajo las mismas condiciones.

Primero, se descarga el modelo base pre-entrenado con imágenes de ImageNet, este modelo base se descarga sin la capa completamente conectada (*include_top=False*).

Segundo, se añaden las capas completamente conectadas y de salida del modelo para adaptarlo al problema, además se dejan fijos los parámetros nativos del modelo para unicamente entrenar las capas nuevas.

Finalmente, el proceso de entrenamiento es el mismo que en el modelo 2, es decir, se aplica Learning Rate Decay y Early Stopping.

3.5 Test de los modelos

Al final del notebook se ha añadido un apartado en el que se toma una muestra de n imágenes aleatorias del conjunto de Test y se obtienen los porcentajes de cada modelo para después imprimir por pantalla la imagen original, el porcentaje original y el porcentaje predicho. De este modo, se puede visualizar el comportamiento de los modelos con cada imagen, también se puede ver la diferencia entre ellos.

4 Resultados

Los resultados obtenidos se compararán los el RMSE = 4.8 propuesto originalmente en el problema (EL RMSE del modelo que se empleará para la

comparación es el RMSE en el conjunto de Test), además también se compararán los Modelos 1 y 2 para determinar si los cambios realizados en el segundo modelo han ayudado a mejorar los resultados.

El modelo 1 presenta un RMSE = 4.86, muy cercano al baseline propuesto. Algo a destacar es que, si se observa el plot con la evolución del RMSE en entrenamiento y validación durante el entrenamiento, se puede ver como el RMSE en validación está moderadamente separado del de entrenamiento, también se puede ver como hacia el final empezaba a disminuir el RMSE indicando una posible mejora si hubiese habido mas Epochs.

El modelo 2 presenta un RMSE = 3.4, el cual mejora el baseline propuesto. Se puede observar como los cambios realizados en el modelo han ayudado a acercar el valor de RMSE en validación al de entrenamiento, especialmente al final del proceso de entrenamiento, donde el RMSE en validación se acerca mucho al valor del Train RMSE.

En cuanto al modelo 3, se puede observar como las tres implementaciones de DenseNet han conseguido una mejora sustancial del RMSE propuesto como baseline (2.38, 2.17 y 2.22 respectivamente). De entre las 3 implementaciones, cabe destacar que las 3 presentan una evolución del RMSE en entrenamiento y validación similar, donde la implementación de *DenseNet169* termina antes el entrenamiento debido al Early Stopping. De las 3 implementaciones, se escoge *DenseNet169* que es la que mejor se adapta al problema, además es la que tiene un mejor equilibrio entre calidad de los resultados y tiempo de entrenamiento/complejidad del modelo.

Finalmente, en los test con n imágenes aleatorias se puede observar como el primer modelo no llega a acercarse al porcentaje real en varias imágenes, asimismo el modelo 2 si consigue aproximarse al porcentaje real, mientras que el modelo 3 obtiene las predicciones mas cercanas a los porcentajes reales en cada imagen.

5 Conclusiones

En este trabajo se han entrenado 3 modelos para abordar un problema de visión artificial empleando CNNs.

El primero de los modelos, que ha servido como base para implementar mejoras y presentar el modelo mejorado como el modelo 2, ha presentado la dificultad de diseñar el modelo desde 0. Posibles mejoras en este modelo, que no sean las implementadas en el modelo 2, implicarían un cambio en la

arquitectura del modelo para mejorarlo desde 0.

El segundo modelo ha mejorado los resultados del primero y ha servido para observar el efecto de los cambios y mejoras implementados, donde se ha intentado mejorar el RMSE y reducir cualquier sobreajuste. Como el modelo 1 no presentaba mucho sobreajuste, no se han implementado todas las posibles técnicas para reducirlo, pero no se descarta la posibilidad de añadir técnicas como la Regularización L2 o el Dropout para reducir cualquier sobreajuste posible en un futuro, además también se podría mejorar acelerar el entrenamiento si se emplease inicializaron de pesos.

El modelo 3 ha permitido observar como afecta la profundidad del modelo DenseNet en el problema haciendo uso de las 3 implementaciones disponibles, escogiendo finalmente DenseNet169. El RMSE obtenido podría mejorarse si, una vez entrenado el modelo, se desbloqueasen los parámetros del modelo y, empleando un Learning Rate muy pequeño, se hiciese Fine Tuning de parámetros.

Finalmente, se han implementado los 3 modelos y se ha hecho una comprobación con 5 muestras aleatorias del conjunto de test, donde se han comprobado los porcentajes predichos con los reales para ver, de forma empírica, la calidad de los resultados obtenidos de los modelos.