Victoria Beltrán Domínguez

Memoria prácticas visión por computador

En esta memoria se pretende describir brevemente los ejercicios realizados para las prácticas de la asignatura.

1. Gender Recognition (3 points)

Images from "Labeled Faces in the Wild" dataset (LFW) in realistic scenarios, poses and gestures. Faces are automatically detected and cropped to 100x100 pixels RGB.

Training set: 10585 images

Test set: 2648 images

Goals:

- Implement a model with >97% accuracy over test set
- Implement a model with >92% accuracy with less than 100K parameters.

En este ejercicio se presentan dos objetivos diferentes.

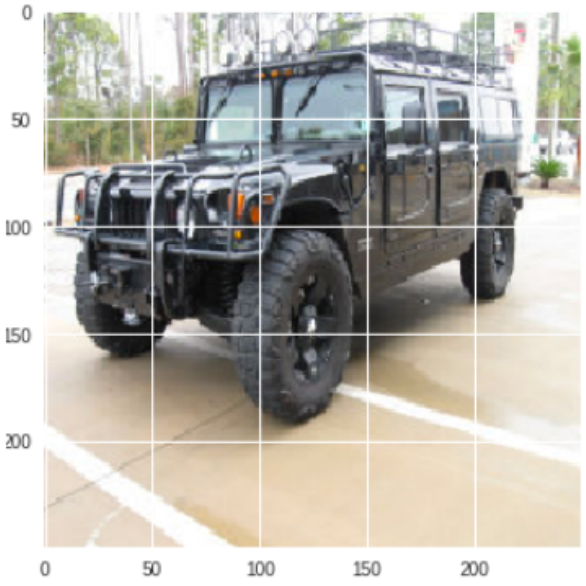
El primer ejercicio se plantea obtener un 97% de accuracy en test. Para conseguirlo, no se ha utilizado ninguna técnica especial. Prácticamente el primer modelo probado ha conseguido esto. En este modelo se define un bloque convolucional en el que hay una convolución con filtros de 3x3, batch normalization, gaussian noise, una relu y un max

pooling. Así, el modelo consiste en una secuencia de bloques con diferentes números de filtro que culminan en una capa densa de 512 neuronas y en otra que ya es la salida de la red. No se han utilizado conexiones residuales y aún así se ha llegado a un accuracy del 98.11%.

En el segundo ejercicio, se plantea implementar un modelo que tenga más de un 92% de precisión pero que tenga menos de 100.000 parámetros. En el paper mencionado, se comenta que existen diferentes formas para reducir el número de parámetros: reducir el tamaño de las imágenes, la profundidad de los filtros, el número de convoluciones, etc. De esta manera, lo primero que hemos hecho ha sido fijar un zoom de 0.4 (pues estoy asumiendo que las imágenes están medianamente centradas). A continuación, he quitado el max Pooling del bloque convolucional y he fijado en la convolución un stride 2. Finalmente, he quitado la capa densa intermedia y he utilizado menos convoluciones y de menor profundidad. El modelo resultante tiene 11.778 parámetros. Aunque son muchos menos de los que tenemos como límite, este modelo consigue un accuracy del 93.46%, por lo que se consiguen ambos objetivos propuestos.

2. Car Model identification with bi-linear models (5 points)

Images of 20 different models of cars.



Training set: 791 images

Test set: 784 images

- Version 1. Two different CNNs:
  Python code: here
- Version 2. The same CNN (potentially a pre-trained model)
  Python code: here

Goals:

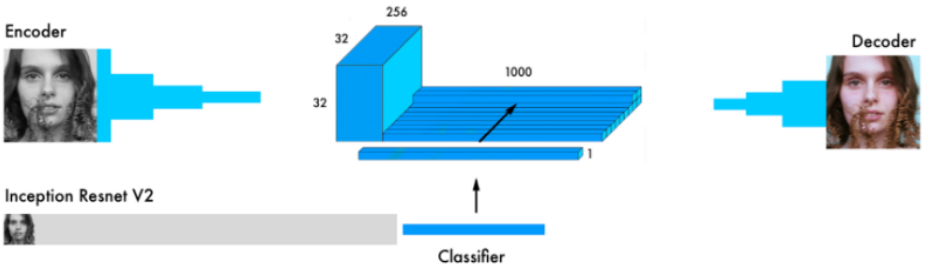
- Understand the above Keras implementations:
  - Name the layers
  - Built several models
  - Understand tensors sizes
  - Connect models with operations (outproduct)
  - Create an image generator that returns a list of tensors
  - Create a data flow with multiple inputs for the model

Suggestion:

- Load a pre-trained VGG16, Resnet... model
- Connect this pre-trained model and form a bi-linear
- Train freezing weights first, unfreeze after some epochs, very low learning rate
- Accuracy >65% is expected

En este ejercicio se pide implementar una red bilineal con un modelo pre entrenado el fin de poder clasificar 20 modelos distintos de coches.
En mi caso, utilizando el código provisto en la versión dos, he reemplazado el modelo convolucional por la última capa convolucional de la vgg16. He definido el modelo como no entrenable y he conectado a su output dos dropouts diferentes, haciendo así que un sólo modelo tenga dos entrenamientos diferentes (simulando el comportamiento de dos redes diferentes aunque sean la misma). Esos dos dropout se los paso al outer product, y entrenamos. Después de 40 epochs de entrenamiento, ponemos la vgg16 como entrenable, y reduciendo el learning rate, volvemos a entrenar toda la red para que el nivel de especificación sea mayor. Obtenemos un accuracy en validación del 73,72%.

3. Image Colorization (3 points)



Goals:

- Understand the above Keras implementations:
  - How to load the inception net
  - How to merge encoder and inception result

Con el fin de desarrollar este ejercicio lo primero que se ha hecho es buscar un conjunto de datos apropiados. En una primera prueba, se escogieron los de gender recognition por proximidad, pero en el entrenamiento final, se ha definido un conjunto de datos de entrenamiento propio.
Particularmente, partiendo del conjunto de datos llamado Image Colorization provisto en Kaggle (que simplemente son imágenes completamente aleatorias coloreadas), hemos escogido las 500 primeras para train y las 50 últimas para validación.
Teniendo este en mente, lo primero que hemos hecho ha sido crear un generador de imágenes. En este generador, cogiendo imagenes aleatoriamente, vamos realizando los siguientes pasos:

1. Cargar el modelo inception (InceptionResNetV2).
2. Normalizamos las imágenes y cambiamos el tamaño de la imagen de entrada a 256 x 256
3. Pasamos la imagen a lab y guardamos la l como una de las entradas a la red.
4. Guardamos ab como test (normalizando por 128 para reducir el exploding gradient).
5. Triplicamos la luminancia de la imagen, y después de cambiar su tamaño se la pasamos a la inception. El resultado de esto será la segunda entrada a la red.

De esta manera, el generador devolverá dos inputs (la luminancia y la salida de la inception) y un output (ab de referencia normalizado).

Una vez entrenamos el modelo con la red dada, el último reto es generar predicciones.
Para ello, tenemos que coger una imagen del generador de test. Este generador te devolverá l, el resultado de la inception y un ab. Como tenemos l y ab podemos reconstruir la imagen original.

A continuación, utilizamos el modelo para recoger el ab predecido y realizamos pasos similares para ver la colorización de la imagen.
Entrenando sólo con 5 epochs y 20 steps en cada epoch obtenemos el siguiente resultado para el conjunto de datos de prueba de gender recognition:

Como se puede observar, como el dataset es bastante homogéneo, aprende rápidamente el color carne. Una vez segura de que el modelo funciona correctamente, le paso el conjunto de datos definido previamente entrenando 300 epochs.
Una vez entrenado, mostramos algunas de las colorizaciones obtenidas:

Como podemos ver, el modelo no coloriza bien. Esto se puede deber tanto a falta de epochs como a la falta de imágenes de train de referencia, pues hemos escogido un conjunto de datos aleatorio, y puede ser que las imágenes de train no coincidan con las de test (aparte de que hay muy pocas). Para solucionarlo, estaría bien encontrar un conjunto de datos mayor y más balanceado y entrenarlo mínimo unos 600 epochs. Sin embargo, dado los recursos y el tiempo disponible, las pruebas realizadas se consideran pertinentes y finalizadas en el ámbito de las prácticas de la asignatura.

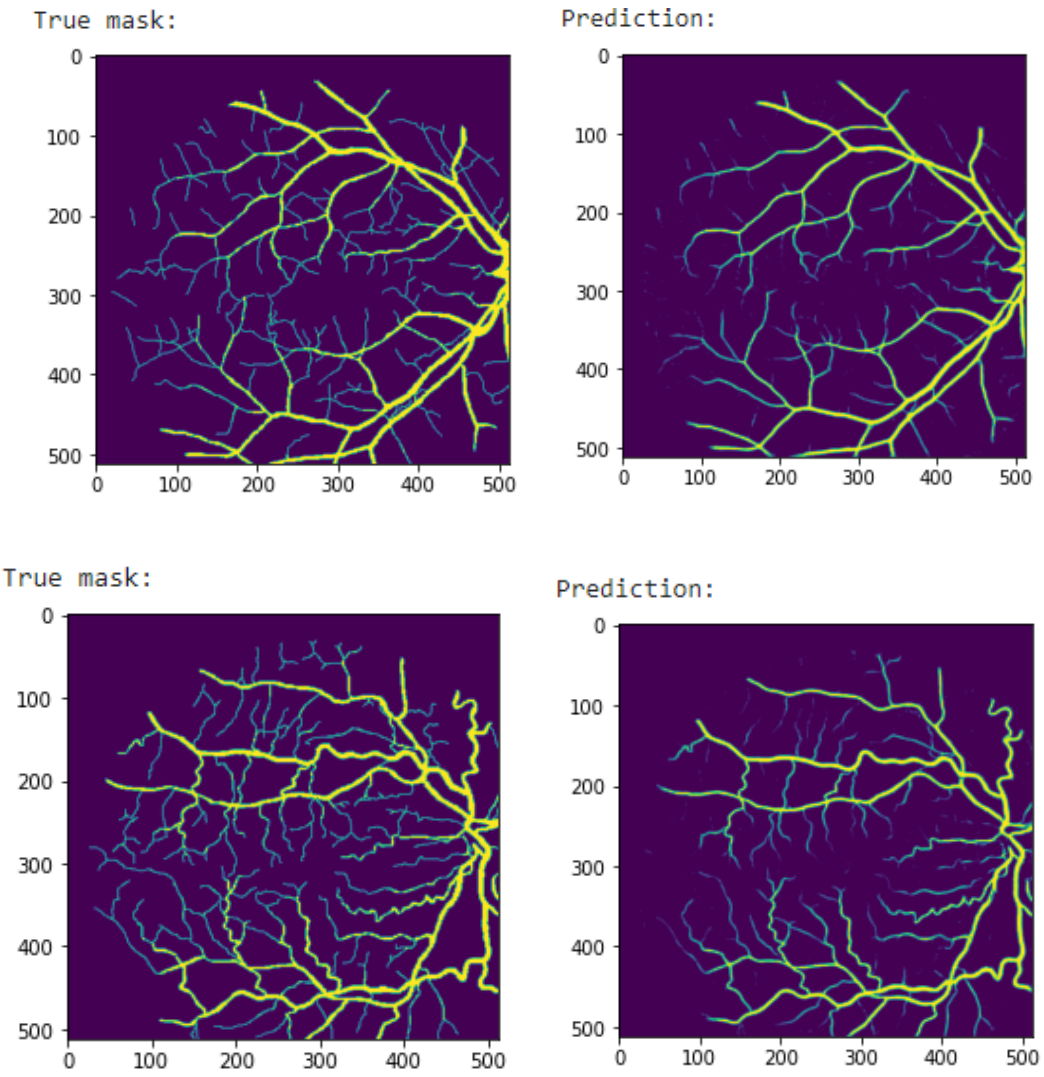
4. Image Segmentation (4 points)

Goals:

- Exercise: implement a UNET.

Para este ejercicio, se ha implementado una U-net siguiendo la estructura presentada en las diapositivas (con diferentes dimensiones). En el propio código se ha especificado cada convolución junto con su resultado.

Una vez claro el modelo y después de entrenarlo 300 epochs, obtenemos un MSE de 0.1. Algún ejemplo de las predicciones hechas podrían ser:



Como podemos ver, las predicciones son bastante similares, aunque aún les falta algo de detalle, sobre todo en las terminaciones azules. Si quisiéramos mejorar los resultados, podríamos entrenar el modelo aún más epochs con el fin de refinarlo.