

Visión Por Computador: Ejercicios Prácticos

Ramon Ruiz Dolz

Abril 2019

1 Introducción

Los ejercicios realizados en este trabajo de desarrollan en el contexto de la asignatura de Visión Por Computador (VPC). En este trabajo se va a explicar el procedimiento seguido para la correcta realización de tres ejercicios distintos propuestos. En cada uno de los ejercicios realizados se ponen en práctica distintos conceptos abordados en las clases teóricas, cubriendo así en estos ejercicios gran parte de los conceptos teóricos aprendidos durante el curso.

Este trabajo se estructura de la siguiente forma, en la section 2 se presenta el primer ejercicio realizado, en el cual se ha realizado la implementación de distintas *Residual Networks* [1] con el objetivo de conseguir un modelo sólido para el reconocimiento de géneros a partir de imágenes. En la section 3 se presentará la implementación y comparativa entre distintos modelos Bi-lineales [2] para el reconocimiento de modelos de coches. Finalmente en la section 4 se presentará la experimentación realizada con distintos modelos de redes para *Style Transfer* [3] con distintas imágenes de contenido y estilo.

2 Gender Recognition

En esta primera tarea se nos propone implementar un modelo que sea capaz de reconocer el género de una persona dada una foto en la que aparece una única cara. El *dataset* empleado para este ejercicio es *Labeled Faces in the Wild*¹ (LFW) el cual contiene 13233 fotos de caras en entornos realistas y con expresiones faciales variadas. Cada imagen tiene un tamaño de 100x100 píxeles en el formato RGB.

Esta tarea esta dividida en dos objetivos distintos. El primero consiste en conseguir un modelo que alcance una precisión mayor al 95% en el conjunto de test. El segundo objetivo consiste en alcanzar como mínimo un 90% en el conjunto de test con un modelo que tenga menos de 100K parámetros. Para ello, se ha implementado una *Residual Network* tomando como punto de partida el modelo estudiado en las sesiones teóricas cuya arquitectura podemos observar en la Figure 1.

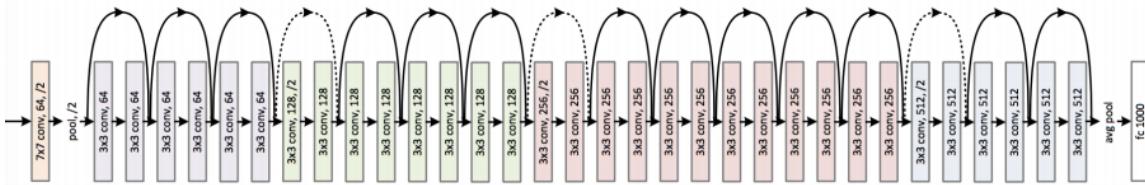


Figure 1: Arquitectura de la ResNet estudiada en las sesiones teóricas.

Por lo tanto, para esta tarea se han implementado dos modelos diferentes. El primero de ellos consiste en un modelo con 1,648,642 de parámetros. Este modelo está compuesto por cuatro bloques convolucionales y un bloque *fully connected* profundo. Cada unidad básica convolucional se compone de un operador convolucional (32 filtros), una *Batch Normalization* y una capa de ruido Gaussiano. Además, cada bloque tiene el doble de unidades básicas convolucionales que el anterior, siendo por lo tanto la sucesión de 1, 2, 4 y 8 unidades convolucionales aquellas que componen la red. Además de esto, a partir del primer bloque,

¹<http://vis-www.cs.umass.edu/lfw/>

los bloques 2, 3 y 4 también disponen de un segundo operador convolucional (32 filtros), un operador de adición que une el final de la unidad básica con el filtro anterior y una función de activación ReLu. Por otra parte, al principio de cada bloque convolucional también se dispone de una capa de *Max Pooling* de 2x2.

El segundo modelo implementado se ha construido a partir del modelo descrito anteriormente. La estructura modular es equivalente, sin embargo tiene dos diferencias fundamentales respecto al modelo anterior. Este segundo modelo debe tener una cantidad menor a 100K parámetros. Concretamente, este modelo consta de 89,090 parámetros. Para conseguir reducir el número de parámetros de más de 1M a esta cifra se ha reducido por una parte el número de bloques convolucionales a la mitad (2) y también se ha reducido el número de filtros en los operadores convolucionales de 32 a 8. De esta forma se ha conseguido un modelo capaz de lidiar con esta tarea con un número de parámetros inferior al máximo establecido en el ejercicio.

Finalmente, en la Table 1 se pueden observar los resultados obtenidos por ambos modelos para esta primera tarea.

Modelo	Convergencia (Epoch)	Train (Acc)	Test (Acc)
Model1	100	0.9539	0.9641
Model2	100	0.9152	0.9396

Table 1: Resultados obtenidos en la tarea de reconocimiento de genero.

Tal y como podemos observar, ambos modelos cumplen con los objetivos establecidos en esta tarea, el Model1 alcanza una precisión mayor al 95% (96.41%) y el Model2, con un número inferior a los 100K parámetros, es capaz de superar el 90% de precisión (93.96%) en el reconocimiento de genero a partir de caras.

3 Car Model Identification with Bi-linear Models

En la segunda tarea, la propuesta consiste en implementar un modelo capaz de diferenciar entre 20 modelos de coche diferentes. En este caso, el *dataset* empleado consiste en 1575 imágenes de distintos tipos de coche de 250x250 píxeles en formato RGB.

Para la realización de esta tarea se ha hecho uso de redes convolucionales bi-lineales [2]. Partiendo del concepto estudiado en las sesiones de teoria, una red bi-lineal es aquella que consta de dos *flujos* convolucionales independientes cuya salida es combinada con la otra formando un único vector, tal y como se puede observar en la Figure 2.

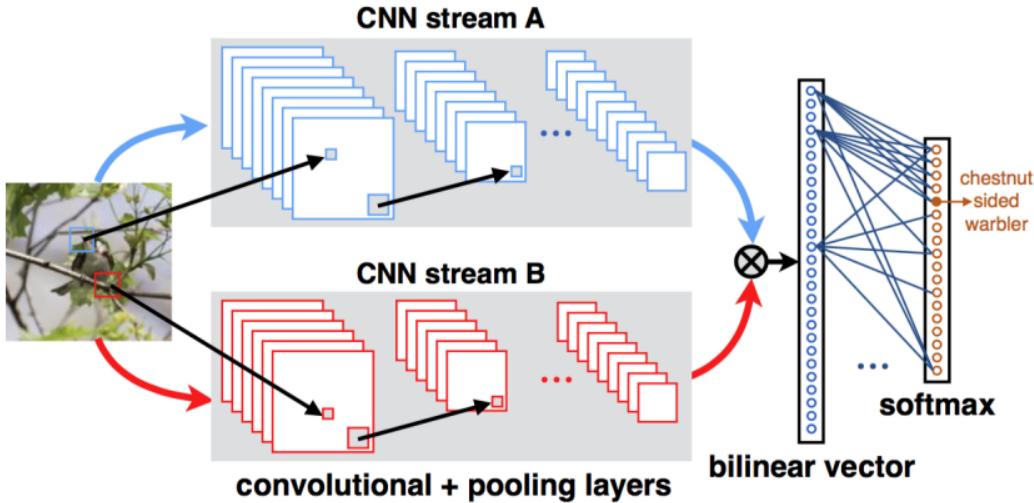


Figure 2: Estructura típica de un modelo bi-lineal.

Mi propósito en la realización de esta tarea ha consistido en analizar el comportamiento de estos modelos en función de las redes convolucionales empleadas en cada *flujo*. Para ello he realizado dos experimentos distintos, en el primer experimento se ha probado el desempeño del modelo bi-lineal haciendo uso de dos redes convolucionales distintas. Sin

embargo, en el segundo experimento se ha tratado de observar el comportamiento del modelo bi-lineal haciendo uso de dos redes VGG16 entrenadas previamente. A continuación se detallan las configuraciones realizadas en cada uno de estos experimentos.

3.1 Experimento 1: Different CNNs bi-linear model

Tal y como se ha comentado, el primer experimento se ha realizado con dos redes distintas en cada uno de los *flujos* del modelo bi-lineal. Para la realización de este primer experimento se ha partido del código proporcionado para la realización de esta tarea. De esta forma ha sido posible entender cómo traducir la Figure 2 a *Keras*. Algunas modificaciones menores han sido necesarias para el correcto funcionamiento del código (definición del optimizador, LRS, compilación del modelo, gestión de los datos de validación, etc.). Ya con el código completamente funcional se ha lanzado este primer experimento alcanzando una precisión de 44% (0.4401). Este valor se tomará como *baseline* para los futuros experimentos.

3.2 Experimento 2: Two pre-trained VGG16 bi-linear model

A diferencia del experimento anterior, en este segundo experimento se han empleado dos modelos VGG16 [4] pre-entrenados. Debido al funcionamiento de Keras, algunas consideraciones han tenido que ser tomadas en cuenta, como por ejemplo renombrar las distintas capas, puesto que al ser dos modelos idénticos, inicialmente contaban con los mismos nombres y esto derivaba en un error. Además, se han lanzado dos experimentos dentro del contexto de este segundo experimento. Uno en el cual se han *congelado* los pesos de las redes VGG y otro con los pesos sin *congelar*. Los resultados alcanzados en este segundo experimento son de 60.9% (0.6097) para el modelo con los pesos sin *congelar* y de 3.32% (0.0332) para el modelo con los pesos *congelados*.

3.3 Resultados

Finalmente, en esta sección se va a realizar una comparativa del comportamiento de los distintos modelos bi-lineales, con el objetivo de observar cual de ellos muestra un mayor rendimiento para la tarea de identificación de modelos de coche. En la Table 2 se pueden observar los valores de precisión alcanzados por cada uno de los 3 modelos explicados anteriormente.

Modelo	Convergencia (Epoch)	Train (Acc)	Test (Acc)
EX1	350	0.9572	0.4401
EX2a	230	0.9937	0.6097
EX2b	150	0.0365	0.0332

Table 2: Resultados obtenidos en la tarea de identificación de modelos de coche.

Como podemos observar, el modelo que mejor comportamiento ha mostrado en la tarea de identificación de modelos de coche es aquel con dos VGG16 pre-entrenadas cuyos pesos están *congelados*. Alcanzando casi un 61% de precisión es con diferencia el modelo bi-lineal con mayor precisión de los tres experimentos que se han realizado en el contexto de esta tarea.

4 Style Transfer

Finalmente, en esta última tarea se han realizado una serie de experimentos entorno a la técnica de *Style Transfer*. El proceso de *Style Transfer*, descrito de forma breve con el objetivo de entender los conceptos fundamentales que se van a manejar en este ejercicio, consiste en obtener una imagen nueva a partir de dos imágenes, una de contenido y otra de estilo. Se podría considerar que la imagen de contenido guiará la estructura de la imagen resultante, mientras que la de estilo se encargará de orientar artísticamente el modelo. Este proceso se puede observar en las siguientes figuras.

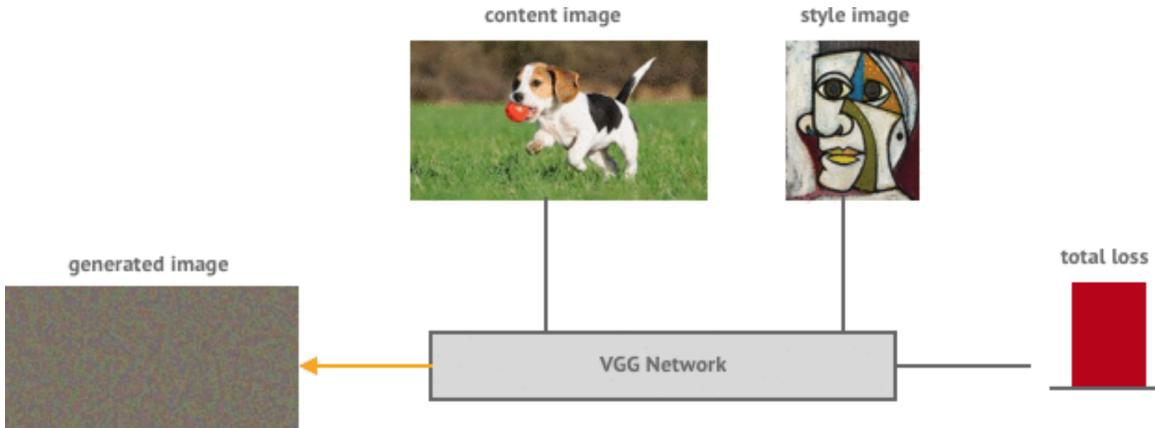


Figure 3: Diagrama elemental de un sistema de ST.

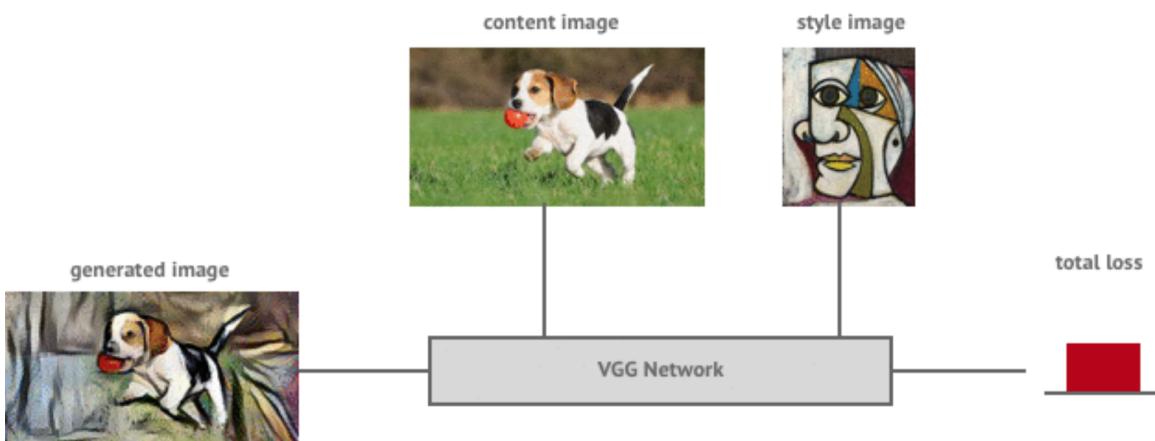


Figure 4: Diagrama elemental de un sistema de ST.

Concretamente, la Figure 3 muestra el estado del sistema al iniciar el proceso de transferencia, mientras que en la Figure 4 se puede apreciar el estado del sistema una vez el modelo ha convergido. Como podemos observar, la función de pérdida se ve reducida a la vez que la imagen resultante adquiere mayor similitud a ambas imágenes de entrada. Esto es posible debido a que realmente, la función de pérdida viene definida como la combinación de dos funciones diferentes, una relacionada con el contenido y otra relacionada con el estilo. A continuación se explican de forma concisa los dos experimentos realizados entorno a este sistema y sus funciones de pérdida.

Para la realización de los experimentos, se ha tomado el código proporcionado en GitHub² como punto de partida. Además, todos los experimentos han sido realizados con la Figure 5 como imagen de contenido y la Figure 6 como imagen de estilo con el objetivo de observar las distintas variaciones en el resultado.



Figure 5: Imagen de contenido empleada en los experimentos



Figure 6: Imagen de estilo empleada en los experimentos

²<https://github.com/RParedesPalacios/ComputerVisionLab/blob/master/Exercises/style.ipynb>

El primer experimento realizado ha consistido en probar el comportamiento de la red modificando los pesos asociados a cada función de perdida (contenido / estilo). Inicialmente, se ha probado a lanzar la red con los pesos predefinidos (contenido = 0.05 / estilo = 5.0) tal y como se puede observar en la Figure 9. A continuación he tratado observar que sucedía igualando ambos pesos (contenido = estilo) cuyo resultado se puede apreciar en Figure 7. Como se puede apreciar, igualando los pesos, a duras penas se puede apreciar el impacto del nuevo estilo sobre la imagen original. Es por esto que se ha ido incrementando el valor proporcional del peso del estilo en las siguientes pruebas. Concretamente se han ido probando con las relaciones de 10 en Figure 8, 500 en Figure 10, 1000 en Figure 11 y 5000 en Figure 12 de peso del estilo respecto al contenido. Mediante este experimento ha sido posible observar perfectamente la progresión y el impacto de los pesos sobre las imágenes de entrada. Empezando a apreciar las características principales del estilo escogido en la relación de 100 de peso del estilo respecto al contenido y llegando a la cúspide en la relación de 1000. A partir de este punto el estilo adquiere demasiada relevancia, perdiendo así las mínimas características importantes del contenido que permitan apreciar la forma de la imagen original. Es por ello que para los siguientes experimentos se aplicará la relación de 500 de peso del estilo respecto al contenido.



Figure 7: Imagen resultante con pesos iguales.



Figure 10: Imagen resultante con el peso del estilo 500 veces mayor.

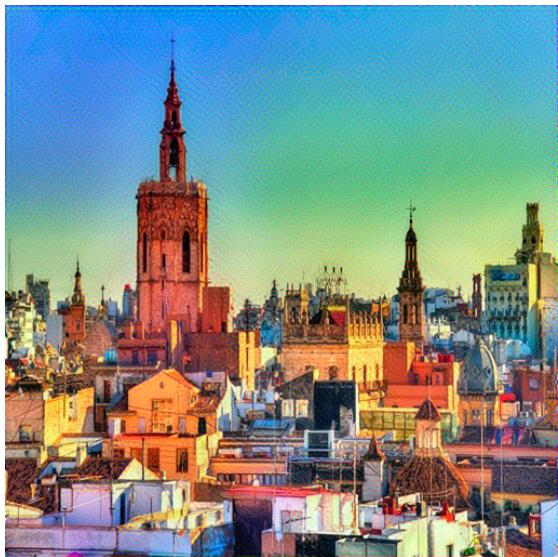


Figure 8: Imagen resultante con el peso del estilo 10 veces mayor.

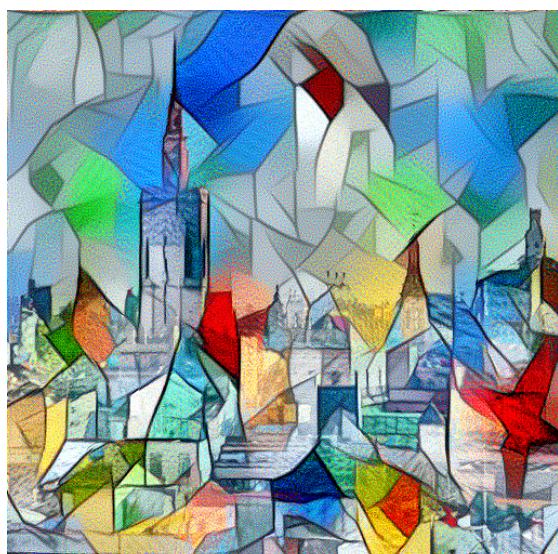


Figure 11: Imagen resultante con el peso del estilo 1000 veces mayor.



Figure 9: Imagen resultante con el peso del estilo 100 veces mayor.

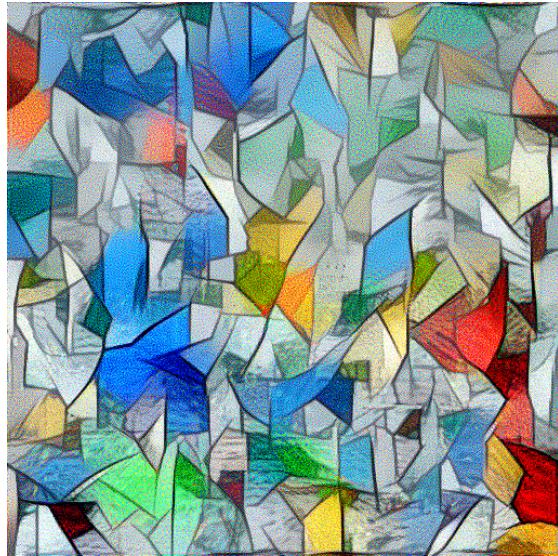


Figure 12: Imagen resultante con el peso del estilo 5000 veces mayor.

En segundo lugar, se ha decidido probar a realizar modificaciones sobre las funciones de pérdida. Concretamente sobre las capas convolucionales tomadas en cuenta a la hora de calcular la pérdida tanto de estilo como de contenido. Se han diseñado dos pruebas diferentes en este segundo experimento, la primera de ellas con el objetivo de observar el comportamiento del modelo obteniendo las características de contenido desde las últimas capas de la red VGG y las características de estilo desde las primeras capas de la red neuronal. En contraposición, en la siguiente prueba se ha hecho justo totalmente al contrario, las características de contenido se han extraído de las primeras capas de la red, mientras que las características de estilo han sido obtenidas a partir de las últimas capas. Estas modificaciones tienen repercusión total sobre la función de pérdida final calculada. Los resultados obtenidos se pueden observar a continuación.

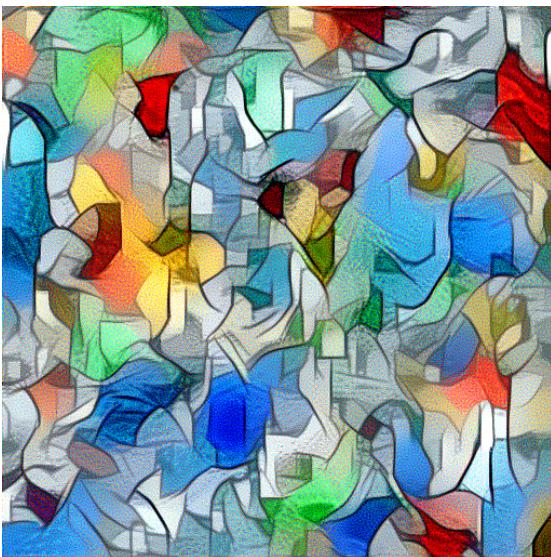


Figure 13: Imagen resultante al obtener las características en la primera prueba.



Figure 14: Imagen resultante al obtener las características en la segunda prueba.

Tal y como se puede apreciar, ambas imágenes han salido muy polarizadas, o bien con muchos rasgos de estilo y pocos de contenido, o bien con muchos rasgos de contenido y pocos de estilo. Concretamente, en la Figure 13 se puede observar como el contenido es a penas perceptible. Por otra parte, en la Figure 14 se puede apreciar el efecto contrario.

En conclusión, no es únicamente importante la estructura en sí de la red, si no también realizar un ajuste con mayor detenimiento de ciertos parámetros que pueden definir el comportamiento del sistema. En este ejercicio se han analizado dos de estos parámetros y se ha mostrado la repercusión que puede tener sobre el resultado una variación sobre el ajuste de estos.

References

- [1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [2] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015.
- [3] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.
- [4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.