CC7221: Reconocimiento Visual con Deep Learning
Tarea 2: Redes Siamesas para Sketch-based Image Retrieval

Prof: José M. Saavedra Rondo
Ayudantes: Pablo Torres, Cristóbal Loyola

Abril 2021

1. Objetivo

Entender y aplicar redes siamesas para modelar en un solo espacio de características dominios diferentes como son los sketches y las fotos.

2. Descripción General

Sketch-based Image Retrieval (SBIR) es un caso particular de recuperación de imágenes por contenido, en donde la consulta es un simple dibujo (sketch), generalmente, carente de color y textura. La idea es recuperar imágenes, comúnmente fotos, ordenadas descendentemente de acuerdo con su similitud a la consulta. En la Figura 1 se muestran los primeros resultados generados por un método de recuperación de imágenes, dada una consulta que representa el *Arco del Triunfo*.

Para solucionar el problema anterior, necesitamos generar un espacio de características en donde convivan las representaciones (vectores de características) de los sketches de consulta y de las fotos que se consultan. Los avances en Deep Learning han producido también avances significativos en este problema. Así, una de las formas más efectivas de generar espacios de características es entrenando redes siamesas, esto tiene especial impacto en el contexto de SBIR, pues permite homologar dominios diferentes, como son los sketches y las fotos, en un espacio compartido.

Figura 1: Ejemplo de resultado SBIR. La primera imagen representa la consulta y el resto representa el ranking generado por cierto método.

En este trabajo se pide entrenar y evaluar dos tipos de *losses* para una red siamesa que utiliza ResNet-34 como backbones en el contexto de *sketch-based image retrieval*. La primera usa *contrastive loss* y la segunda *triplet loss*. Para favorecer la convergencia, ambos casos ocupan, además, *cross-entropy loss*.

3. Descripción Detallada

3.1. Backbones

Una arquitectura siamesa se compone de 2 backbones, cada una extraerá información de un dominio determinado. En este caso, tendremos un backbone enfocado en los sketches y otro en las fotos a recuperar. Es altamente recomendable preentrenar los backbones en cada uno de sus dominios. Para este trabajo, se dispondrá de los backbones preentrenados en los dominios subyacentes. Así, el backbone para skeches ha sido entrenado en el conjunto de sketches de Eitz, mientras que para las fotos el backbone ha sido entrenado en ImageNet. Ambos backbones corresponden a una ResNet-34. Un diagrama de esta arquitectura se muestra en Figura 2. Los checkpoints para cada dominio pueden ser descargados desde la sección *Checkpoints* del repositorio `https://github.com/jmsaavedrar/convnet2`.

- **Backbone para fotos**: Ir a ImageNet y presionar *Download*.
- **Backbone para sketches**: Ir a Sketches y presionar *Download*.

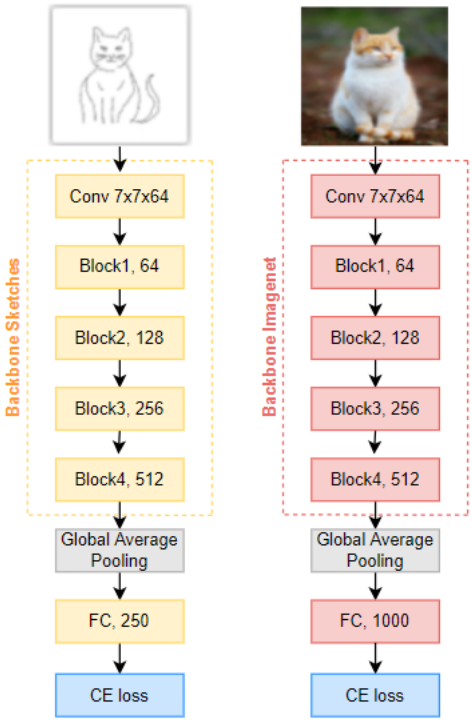

Figura 2: Esquema de la arquitectura utilizada por los backbones de la red siamesa planteada para esta tarea.

3.2. Arquitectura Siamesa

La arquitectura siamesa planteada en este trabajo se extiende de los backbones antes mencionados, agregando 2 capas fully-connected con pesos compartidos entre los dos dominios. La lógica aquí es que los backbones procesan en forma independiente cada dominio, y las capas compartidas finales tratan de homologar las representaciones, de modo que ambas convivan en un solo espacio de características. Esto es, los vectores de características generados por un sketch y una foto que comparten la misma semántica, caen muy cerca en el espacio de características. En esta tarea, se pide entrenar dos versiones de la red siamesa descrita anteriormente. La primera usa *contrative-loss*, mientras que las segunda se entrena en base a tripletas con *triplet-loss*. En ambos casos, el loss generado por los embeddings se complementa con *cross-entropy loss*, de modo que las representaciones generadas mantienen coherencia entre las clases. **Para los experimentos se sugiere ponderar con 0.4 el loss de los embeddings**

y 0.6 para cross-entropy. La Figura 3 muestra la arquitectura para el modelo que usa *contrastive-loss*, mientras que la Figura 4 muestra la arquitectura para *triplet-loss*.

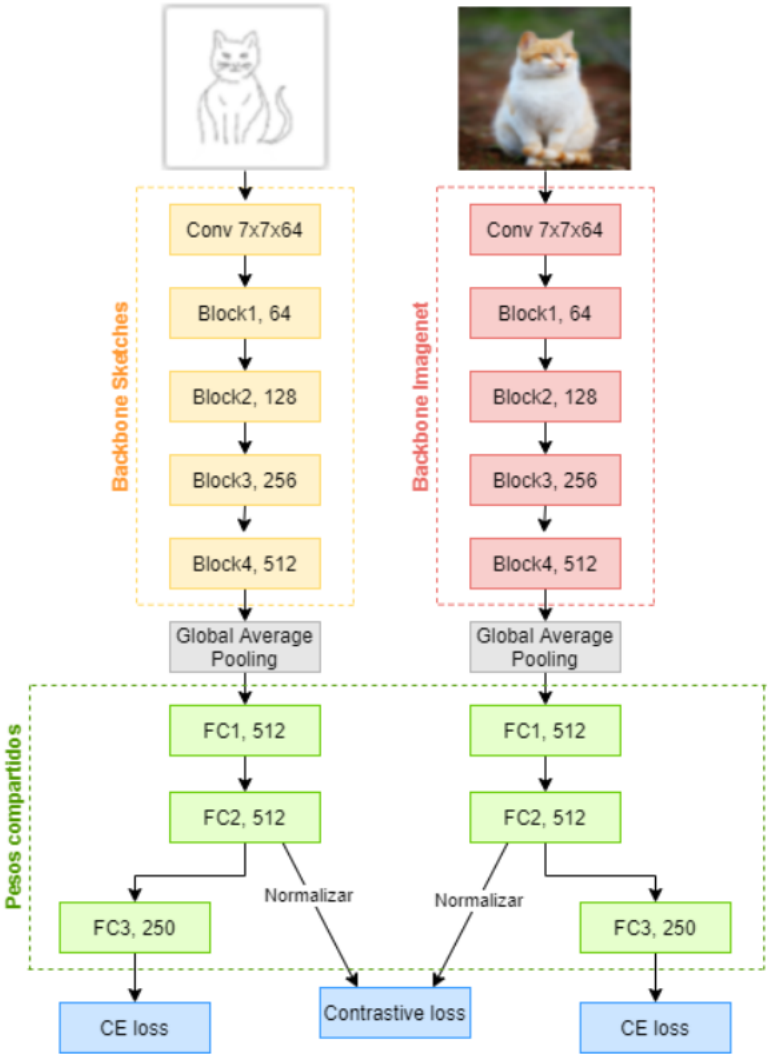

Figura 3: Esquema de la arquitectura de la red siamesa con *contrastive-loss* planteada para esta tarea.

3.3. Datasets

3.3.1. Para Entrenamiento

Para entrenar la red siamesa utilizaremos el dataset de Eitz y Flickr25K, ambos se componen de 250 clases homologadas. Un par positivo se forma con objetos de la misma clase, mientras que un par negativo con elementos de clases diferentes. Los datasets pueden ser descargados de:

- Dataset de sketches (Eitz's): `https://www.dropbox.com/s/ut350iwgby9swk2/Sketch_EITZ.zip?dl=0`
- Dataset de fotos (Flickr25K): `https://www.dropbox.com/s/khbxruh3acq84eg/Flickr25K.zip?dl=0`

3.3.2. Para Evaluación

Para evaluar el desempeño de los modelos utilizaremos el dataset Flick15K que contiene 14501 imágenes a recuperar y 329 sketches. Este dataset puede ser descargado de:

- Dataset Flickr15K: `https://www.dropbox.com/s/q5ew09x4e3rsiht/Flickr15K.zip?dl=0`

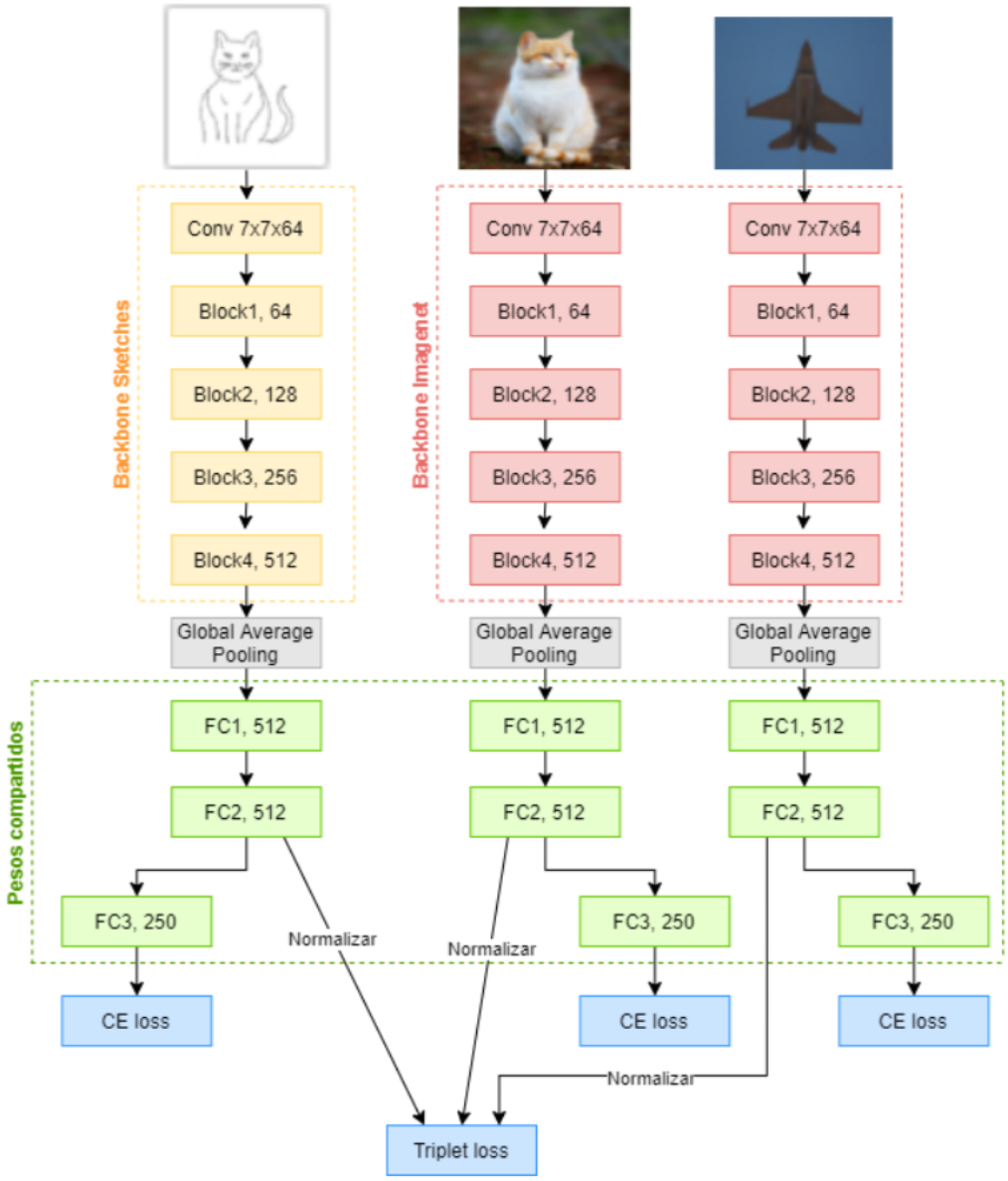

Figura 4: Esquema de la arquitectura de la red siamesa con *triplet-loss* planteada para esta tarea.

3.4. Código Fuente y Parámetros

Para implementar las redes se sugiere seguir el código descrito en `https://github.com/jmsaavedrar/convnet2` que viene con módulos para leer datos y una arquitectura siamesa de ejemplo basada en ResNet-34.

Con respecto a los hiperparámetros, la redes se entrenan con imágenes de tamaño 224×224. Para la optimización deberán utilizar SGD con CosineDecay, como aparece en el código de ejemplo.

3.5. Evaluación

En este trabajo se pide evaluar mAP (mean average precision), recall-presicion y recall ratio (tasa de recuperación para diferente numéro de imágenes recuperadas). Para las 2 últimas mediciones, deberá presentar gráficos.

Adicionalmente, deberá mostrar un ejemplo por cada tipo de sketch (uno por cada clase) del conjunto de evaluación.

4. Informe

1. **Abstract o Resumen**: es el resumen del trabajo.
2. **Introducción**: Aquí se describe el problema y el contexto. Explique qué es SBIR y cómo funcionan las redes siamesas y los *losses* involucrados. [10 %]
3. **Desarrollo**: Aquí se describe el diseño e implementación de los programas necesarios para realizar sus experimentos (40 %).
4. **Resultados Experimentales y Discusión**: Aquí se prensentan los resultados, pero lo más importante es analizarlos. Observe y describa el comportamiento de los modelos en base a las métricas mencionadas anteriormente. ¿Puede genera algunas recomendaciones en base a sus observaciones? [40 %]
5. **Conclusiones** [10 %]

5. Entrega

La entrega se realiza por u-cursos hasta el domingo 12 de mayo, 2021, 23:50 hrs. Se debe incluir:

1. Código fuente (en Python)
2. Informe