Redes siamesas

Índice

Referencias principales

- G. Koch, R. Zemel, R. Salakhutdinov: *Siamese Neural Networks for One-shot Image Recognition*, ICML, 2015.
- F. Schroff, D. Kalenichenko, J. Philbin: *FaceNet: A Unified Embedding for Face Recognition and Clustering*, CVPR, 2015.
- L. Leal-Taixé, C. Canton-Ferrer, K. Schindler: *Learning by tracking: Siamese CNN for robust target association*, CVPR, 2016.
- W. Li, O. H. Jafari, C. Rother, *Deep Object Co-Segmentation*, preprint en ArXiv, 2018.

(Todas las imágenes son originales salvo que se indique lo contrario.)

Tiempo estimado = 1 sesiones de 2 horas (1 semana)

1. Motivación

Hay tareas que, a priori, son ejemplos evidentes de clasificación. El ejemplo más claro es el del reconocimiento facial en puestos automáticos de frontera. La persona debe situarse a una distancia determinada y en unas condiciones de iluminación controladas y conocidas sobre un fondo monocromo. La máquina detecta las dimensiones del rostro y después clasifica a dicha persona en dos grupos: {puede pasar, debe ir a una ventanilla manual}.

Sin embargo, si lo pensamos despacio, la tarea de clasificación no escala bien. ¿Cómo entrenamos la máquina? El conjunto de entrenamiento tendría que tener muestras de individuos que pueden pasar y otros que deben ser revisados, ¿pero qué responde si llega un individuo nuevo? ¿pasar o no pasar es algo que dependa de sus facciones?

Evidentemente no, el individuo es identificado en una base de datos y desviado a la ventanilla. Si entrenamos una red que aprenda a responder cuando dos imágenes son iguales y cuando son diferentes, el problema de permitir el acceso se resolvería comparando la imagen adquirida en el momento con el conjunto de imágenes de una base de datos de gente que debe ser retenida o estudiada por un humando.

Las redes siamesas son el tipo de redes propuesto por Koch en 2015 para esta tarea, es decir para etiquetar como iguales o diferentes dos ejemplos.

2. Arquitectura de las redes siamesas

Una red siamesa en realidad son dos redes, cada una de las cuales recibe un ejemplo y produce una salida. Después ambas salidas se combinan en una nueva red que produce la salida final.

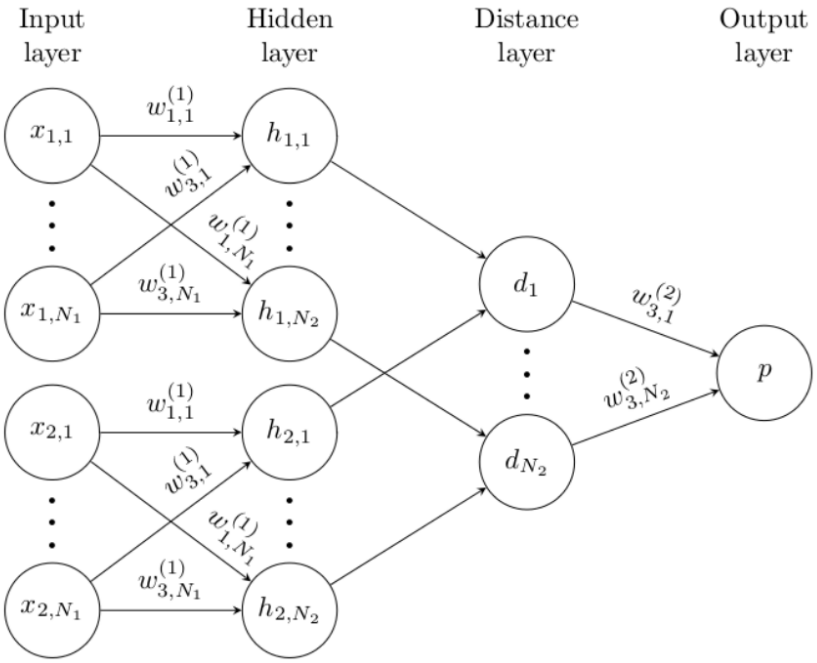

Figura 1: Esquema de una red siamesa. (Fuente: G. Koch, R. Zemel, R.Salakhutdinov: *Siamese Neural Networks for One-shot Image Recognition*, ICML, 2015)

En la Figura 1 se muestra un esquema de arquitectura, organizada en capas de neuronas.

- La primera capa es la de entrada de datos y se divide en dos.
  - ▷ Las neuronas $x_{1,1}$ a x_{1,N_1} reciben las N_1 características de los ejemplos que se introducen por la red #1 (la de arriba).
  - ▷ Las neuronas $x_{2,1}$ a x_{2,N_1} reciben las N_1 características de los ejemplos que se introducen por la red #2 (la de abajo).

  Evidentemente los ejemplos introducidos por cada una de las redes siamesas tienen el mismo número de características (misma resolución por ejemplo).
- Sigue una capa oculta con N_2 neuronas, tanto en la red #1 como en la red #2. Pero si nos fijamos en la figura, podemos apreciar que los pesos de conexión son los mismos para ambas redes (*weight tying*). En la práctica esto es tan sencillo como utilizar el mismo modelo de red con diferentes entradas.

  De esta manera dos imágenes iguales no acaban mapeadas por sus respectivas redes en lugares diferentes del espacio de características ya que las dos realizan la misma función.
- A continuación se añade una nueva capa con tantas neuronas como la última capa de las dos redes previas, que en la Figura 1 es N_2, donde se calcula la distancia L1 entre las salidas generas por sendas redes. Es decir calcula $d_i = |h_{1,i} - h_{2,i}|$ para $i = 1, \ldots, N_2$.

  Por tanto el resultado es un vector de N_2 características.
- Finalmente un cabezal clasificador estima si las dos imágenes introducidas son la misma o no. Esta estimación se compara con la etiqueta real.

En la Figura 2 se muestra con más detalle la estructura convolucional de las redes #1 y #2 propuesta por los autores.

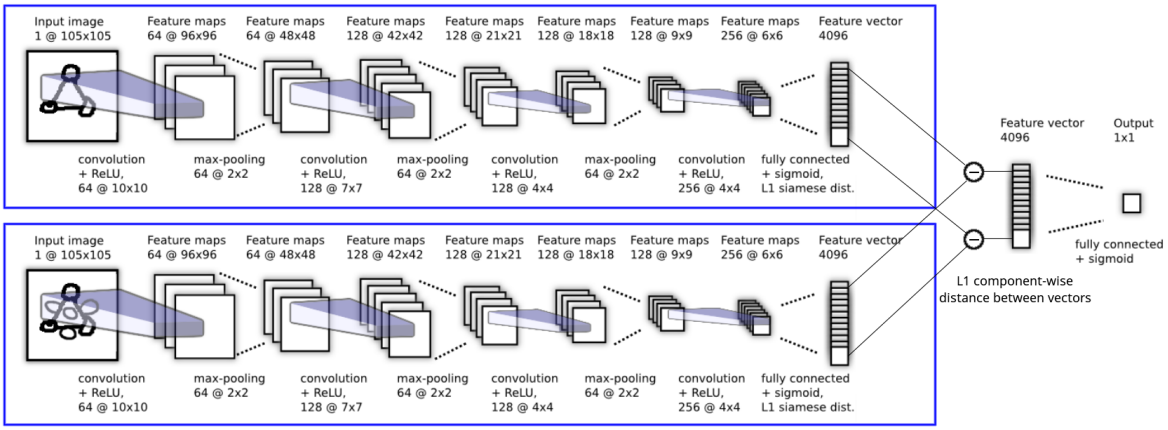

Figura 2: Detalle de la arquitectura de una red siamesa.

3. Entrenamiento de las redes siamesas

La salida de una red siamesa estima si las dos imágenes introducidas son la misma o no; y como es una tarea supervisada, nosotros sabemos la respuesta. Por tanto se trata de un problema de clasificación binario, con etiquetas {"son iguales", "no son iguales"}; así que la entropía cruzada binaria (*binary cross entropy*) es una función de pérdida apropiada.

De modo que, dado el i-ésimo par de imágenes $\{x_1^{(i)}, x_2^{(i)}\}$, su pérdida es:

$$\ell^{(i)}\left(x_1^{(i)}, x_2^{(i)}\right) = y \log p(\hat{y}) + (1-y)\log(1-p(\hat{y})) + \lambda^\intercal \|\mathbf{w}\|^2, \tag{1}$$

donde $y = 1$ si son iguales y 0 en caso contrario, e igualmente $\hat{y} = 1$ si la red estima que son iguales y 0 en caso contrario. Además la red estima la etiqueta con una cierta probabilidad $p(\hat{y})$. Por último los autores añaden un término de regularización con un vector de hiperparámetros λ.

Función de pérdida triple. Una función de pérdida más elaborada y popular es la *triplet loss*, propuesta por Schroff et al. en el CVPR de 2015. Esta pérdida requiere de una estructura un poco diferente, y de un entrenamiento adaptado a esta estructura, mostrada en la Figura 3.

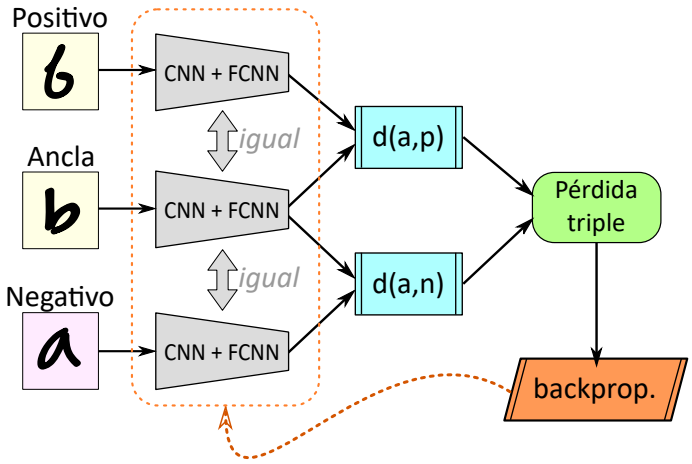

Figura 3: Estructura de una red siamesa que utilice la función de pérdida triple.

1. Tomamos una imagen, a la que denominaremos "ancla", y dos imágenes más, una similar ("positiva") al ancla y otra diferente ("negativa").
2. Cada una de las imágenes es evaluada por una red neuronal. Sin embargo, igual que ocurría antes, estas tres redes comparten los pesos, de modo que cuando actualicemos los de una, actualizaremos los de las otras dos.

   Como resultado de esta etapa *feedforward* obtenemos un vector de características por cada una de las redes.
3. Calculamos la distancia entre el ancla (a) y la positiva (p), y la distancia entre el ancla y la negativa (n), nombradas como $d(a,p)$ y $d(a,n)$ en la Figura 3.
4. Del mismo modo que la función de pérdida (1) se definía para el i-ésimo par de imágenes, la pérdida triple se define ahora para el i-ésimo trío $(a^{(i)}, p^{(i)}, n^{(i)})$.

   El objetivo de la pérdida es lograr que la distancia desde el ancla a la positiva sea pequeña mientras que la distancia a la negativa sea grande. Esto es equivalente a decir que

   $$d(a^{(i)}, p^{(i)}) \leq d(a^{(i)}, n^{(i)})$$

   Para evitar que esta desigualdad se satisfaga con la solución trivial $0 = 0$, se impone que exista una diferencia de $d(a^{(i)}, n^{(i)})$ a $d(a^{(i)}, p^{(i)})$ tenga que ser siempre mayor o igual que

un cierto margen m, que nosotros fijamos. En definitiva, imponemos que

$$d(a^{(i)}, p^{(i)}) - d(a^{(i)}, n^{(i)}) + m \leq 0.$$

Finalmente, dado que queremos vamos a minimizar la pérdida, forzamos a que sólo ajustaremos los parámetros cuando no se cumple la última desigualdad, es decir cuando el término de la derecha sea superior a cero; si es menor o igual no necesitamos ajustarlos. Por tanto la pérdida triple queda:

$$\ell^{(i)}\big(a^{(i)}, p^{(i)}, n^{(i)}\big) = \text{máx}\{d(a^{(i)}, p^{(i)}) - d(a^{(i)}, n^{(i)}) + m; 0\} \tag{2}$$

Función de coste. Tanto si se utiliza la función de pérdida (1) como si es la (2), ésta sólo se aplica a un par o un trío de ejemplos. Para un lote de N_s pares o tríos tendríamos $\mathcal{L} = \sum_{i=1}^{N_s} \ell^{(i)}$. Por tanto, a partir el conjunto de datos se deben construir conjuntos de pares de ejemplos positivos y de pares negativos.

Además, es conveniente que los pares positivos sean "diferentes" y que los pares negativos sean "parecidos". Es decir que visualmente sea difícil decidir si dos fotos son diferentes o similares, aunque conociendo el *ground truth* sabemos que así es. De esta manera logramos que sea más difícil cumplir con el margen. Así continuaremos actualizando los pesos y mejorando la red.

4. Aplicaciones de las redes siamesas

4.1. Seguimiento visual

Cuando se trata de seguir a múltiples objetivos, la técnica más utilizada es el Seguimiento por Detección (*Tracking-by-Detection*, TbD).

TbD consiste en una primera fase en la que se detectan todos los objetivos, blancos o *targets* a seguir y después se asocian con los *trackers*, estructuras de datos que almacenan y procesan la trayectoria de cada uno de ellos.

La asociación es una tarea característica y fundamental en TbD ya que:

- mantiene la identidad del objetivo,
- sirve para decidir si debemos crear o eliminar uno o varios trackers, según se produzcan entradas, salidas, oclusiones o apariciones.

Precisamente, con las redes siamesas podemos aprender a decidir si dos imágenes, o recortes de imágenes, son similares o diferentes. En la Figura 4 se muestra un esquema de como funcionaría la asociación, y en la Figura 5 la arquitectura completa.

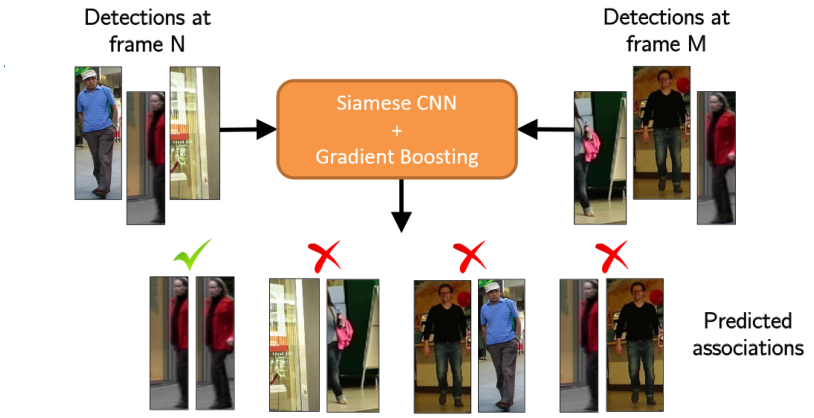

Figura 4: Proceso de asociación mediante redes siamesas para seguimiento visual con TbD. (Fuente: L. Leal-Taixé, et al: *Learning by tracking: Siamese CNN for robust target association*, CVPR, 2016)

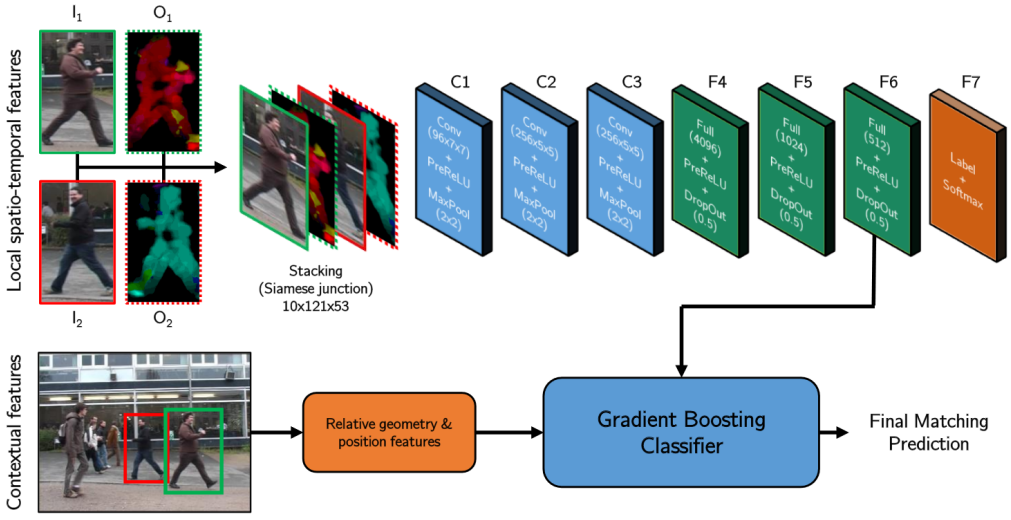

Figura 5: Arquitectura propuesta para TbD con redes siamesas. (Fuente: L. Leal-Taixé, et al: *Learning by tracking: Siamese CNN for robust target association*, CVPR, 2016)

4.2. Co-segmentación

La co-segmentación consiste en segmentar objetos de la misma o de varias clases que son comunes o están presentes en un par de imágenes. Esto significa que el método aprender a ignorar tanto fondos similares como diferentes. En la Figura 6 se muestra un ejemplo de los resultados que se desean obtener.

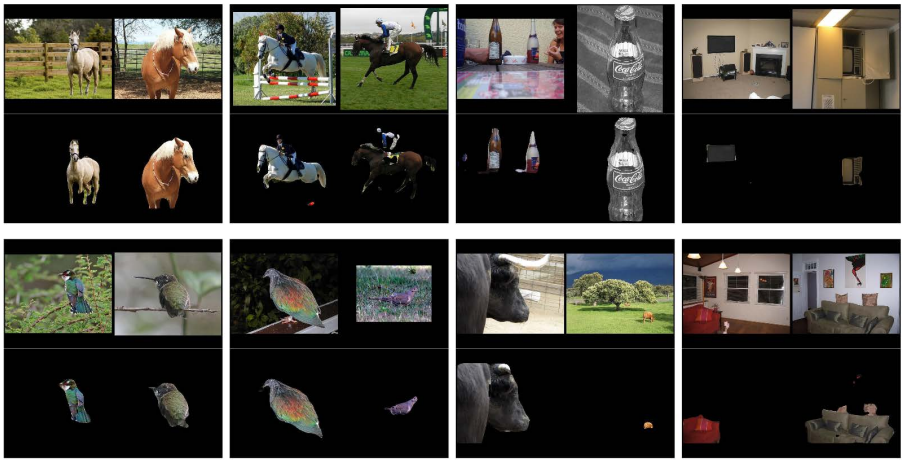

Figura 6: Co-segmentación (Fuente: W. Li, et al. *Deep Object Co-Segmentation*, ArXiv, 2018)

Li et al. han propuesto un método de co-segmentación basado en redes siamesas, cuya arquitectura se muestra en la Figura 7. Al par de CNN siamesas, A y B, con pesos compartidos, que vamos a denominar "Encoders siameses", le sigue una capa en la que se mide la correlación mutua entre los mapas de características que cada una de ellas devuelve, f_A y f_B.

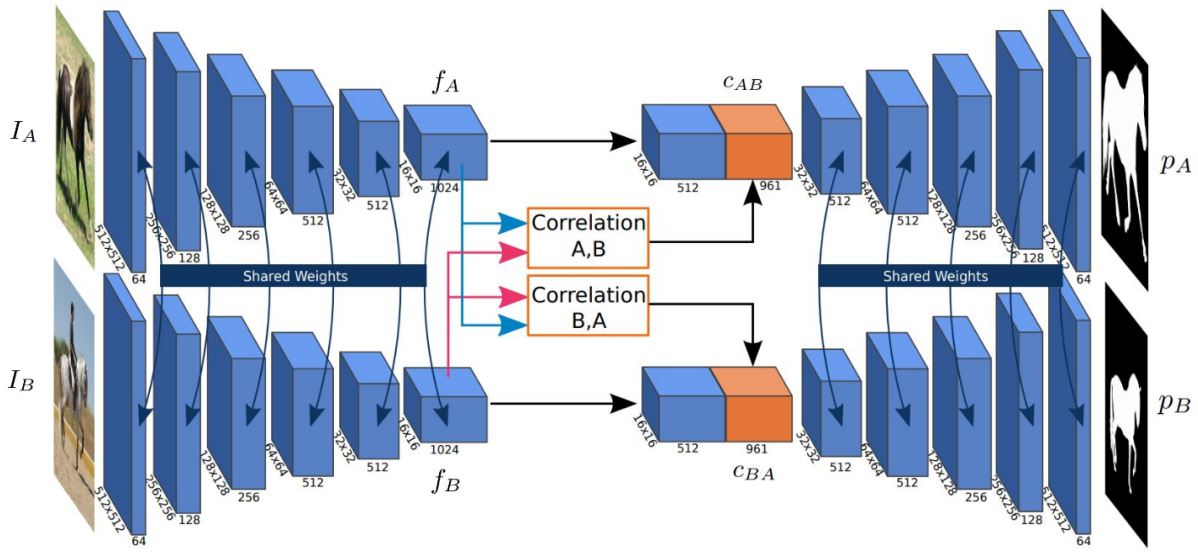

Figura 7: xxxx (Fuente: W. Li, et al. *Deep Object Co-Segmentation*, ArXiv, 2018)

La correlación mutua se mide entre un píxel de una imagen y una vecindad de resolución $D \times D$ de la otra. La expresión para la correlación entre A y B es $c_{A,B}(i,j,k) = \langle f_A(i,j), f_B(m,n) \rangle$; donde (i,j) son las coordenadas del pixel en A, (m,n) son las coordenadas de un píxel en B, dentro de la vecindad, y $k = (n-j)D + (m-i)$. Además las dimensiones de la vecindad están restringidas a $D = 2 \cdot max(w,h) + 1$, para mapas de características de resolución $w \times h$. La correlación $c_{B,A}$, entre B y A, tiene una expresión simétrica.

Es decir que el resultado de cada correlación es un tensor 3D, concretamente de tamaño $w \times h \times D^2$. Sendos tensores resultantes se concatenan a los mapas de características, y pasan a través de sendas redes convolucionales siamesas con la estructura de un decoder, por lo que las denominaremos "Decoders siameses". El objetivo es que cada uno de ellos produzca una imagen binaria, clasificando el fondo como "0" y el primer plano como "1".

Los resultados se comparan con el *ground truth*, que obviamente está disponible junto con el conjunto de imágenes. Como cada pixel es binario se puede utilizar como función de pérdida la entropía cruzada binaria. Y puesto que cada imagen producirá una pérdida, podemos combinarlas simplemente sumándolas. A partir de ese resultado ya podemos aplicar retropropagación para ajustar los pesos.

5. Ejercicios Prácticos

- Construir una red siamesa con pérdida triple a partir de la red siamesa proporcionada en clase.
- Crear una red siamesa para co-segmentación.