Bachelor's degree in Computer Science and Engineering

Visión Artificial

2023-2024

Práctica Final

"Detección de Objetos con Faster-RCNN"

Leandro Andrada Guio – 100451162

Ignacio Arnaiz Tierraseca – 100428997

Luis Daniel Casais Mezquida – 100429021

Daniel Obreo Sanz – 100451058

Profesor:

Fernando Díaz de María

1. Introducción

En ésta práctica se busca diseñar y entrenar una red neuronal convolucional de proposición de regiones (RCNN), basada en el modelo Faster-RCNN[1], y usando la base de datos de PASCAL VOC 2012[2] para detección de objetos.

Los objetos a detectar son botellas, sillas, mesas, y sofás, lo que, añadiendo el fondo, nos da un total de 5 clases. El entrenamiento de la red se hizo con 12 *epochs*, un *step size* de 4, y un *learning rate* de 0.001, con *batch size* de 1.

Como medida de evaluación se usó el *F1 score*, el cual tiene en cuenta tanto la precisión como el *recall* del modelo.

2. Desarrollo

2.1. *Data augmentation*

Para la práctica se cuenta con un conjunto de datos basados en imágenes de distintos objetos en distintos espacios, a fin de que la red diseñada sea capaz de reconocer y localizar objetos de cuatro clases: 'botella', 'silla', 'mesa' y 'sofá'.

A fin de poder mejorar el entrenamiento de la red, se han llevado a cabo una serie de transformaciones sobre el conjunto de imágenes originales para así proveer al conjunto de entrenamiento de una mayor variación y un mayor número de datos de entrenamiento[3]. Todas las transformaciones son aleatorias, ya que al ejecutar cada *epoch* se generará una nueva versión.

Dado que el *dataset* trabaja con imágenes con la librería Pillow[4], decidimos usar las transformaciones de la librería torchvision[5], ya que permiten trabajar con este formato. La clase `torchvision.transforms.ColorJitter`[6] permite modificar los parámetros cromáticos de una imagen dada, como el brillo, contraste, saturación y matiz. En la implementación actual, tras experimentar con diferentes valores, se usan valores de 0.28 para el brillo, 0.3 para contraste y saturación, y 0.06 para el matiz, que aunque pueda resultar bajo, permite mantener los colores en términos generales entre tonos marrones o amarillos/rojos oscuros (colores que pueden ser alterados por la luz natural o bombillas), ya que a valores más elevados comienzan a aparecer colores que no serían naturales.

[1] L.-C. Chen, G. Papandreou, F. Schroff and H. Adam, *Rethinking Atrous Convolution for Semantic Image Segmentation*, 2017. arXiv: 1706.05587 [cs.CV]

[2] http://host.robots.ox.ac.uk/pascal/VOC/voc2012

[3] El código de estas transformaciones se encuentra en `src/data_augmentation.py`.

[4] https://python-pillow.org

[5] https://pytorch.org/vision

[6] https://pytorch.org/vision/stable/generated/torchvision.transforms.ColorJitter

Se decidió no incluir otras transformaciones de rotación o recorte de las imágenes, ya parte de los objetos podría quedar fuera del campo visible, lo cual provocaba peores resultados en el modelo.

En la Figura 1 se pueden observar ejemplos de posibles transformaciones de la imagen original (1a).

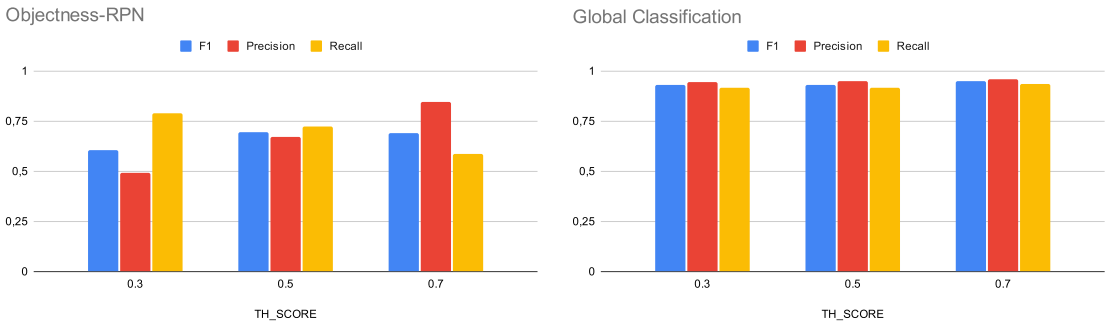
(a) *Imagen original* **(b)** *Transformación 1* **(c)** *Transformación 2*

(d) *Transformación 3* **(e)** *Transformación 4* **(f)** *Transformación 5*

Fig. 1: *Ejemplo de transformaciones aplicadas a una imagen*

2.2. Influencia de los Umbrales en la Inferencia

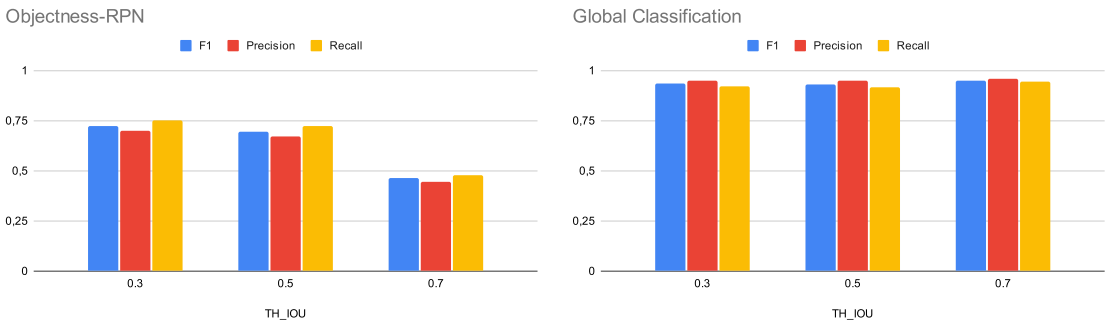
Viendo la Figura 2a del Objectness-RPN, a medida que se aumenta el TH_SCORE aumenta la precisión y disminuye el *recall*, esto se debe a que el modelo es más selectivo en la detección de regiones. Mientras que en la Figura 2b, podemos ver que se mantiene estable, por lo tanto el clasificador no se ve afectado.



(a) *Objectness-RPN* **(b)** *Global Classification*

Fig. 2: *TH_SCORE*

Como se puede observar en la Figura 3a, a medida que se aumenta el umbral para determinar si una región se considera correcta o no, se reducen todas las métricas, esto se debe a que es difícil para nuestro modelo detectar las regiones correctas. Mientras que para el clasificador en la Figura 3b podemos ver que se mantiene estable, por lo tanto el clasificador no se ve afectado.



(a) *Objectness-RPN* **(b)** *Global Classification*

Fig. 3: *TH_IOU*

2.3. *Anchors*

Analizando las salidas del modelo, y comparando los *anchors* empleados (su tamaño y relación de aspecto) con las *bounding boxes* de las imágenes proporcionadas, se puede observar que la distribución de las predicciones, con respecto a la relación de aspecto y el tamaño es bastante similar a la verdad objetiva (Figura 4).

Debido a los *anchors* empleados, los objetos que mejor se detectan suelen ser los que tienen *bounding boxes* con una relación de aspecto más cercana a uno (Figura 5). Por lo que las sillas, que tienen *bounding boxes* más 'cuadradas', se predicen mejor con este modelo.

En cuanto al tamaño de los *anchors*, se predicen mejor los objetos más grandes, seguramente a que la reducción del tamaño de la imagen hace que se pierda algo de información.

Como conclusión de esta sección, se podría mejorar el modelo añadiendo *anchors* con una relación de aspecto algo superior (3:1, o 4:1) para captar los objetos en esa sección que no se están percibiendo. De igual manera, se podría probar a añadir *anchors* de menor tamaño, para captar las botellas o de mayor tamaño, para captar mejor las mesas o los sofás.

2.4. *RoI pooling* y clasificación

La técnica de *RoI pooling* resulta de gran eficacia a la hora de clasificar y detectar objetos pues busca normalizar el tamaño de las regiones en las que pondrá atención el modelo para

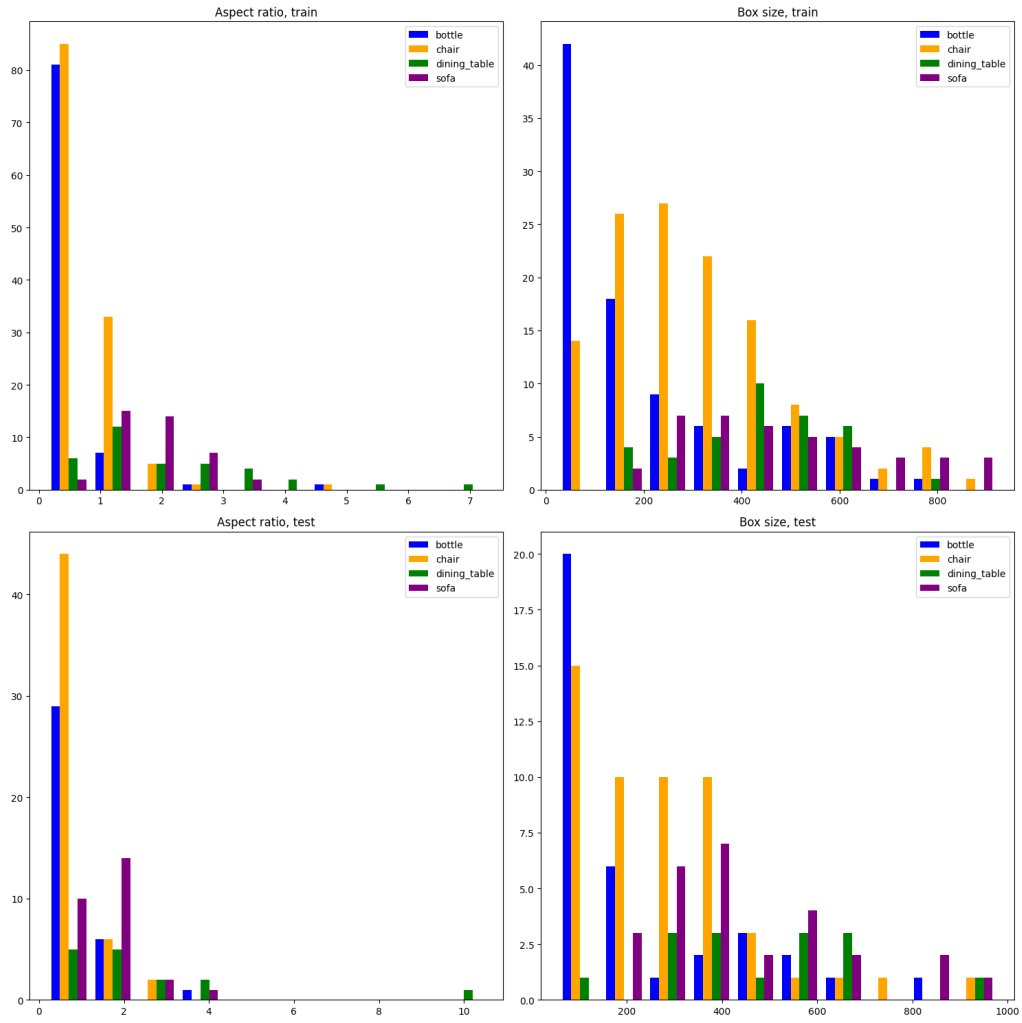


Fig. 4: *Relación entre la relación de aspecto y las bounding boxes*

que sean uniformes y evitar que la diferencia de tamaño pueda distorsionar los resultados. Es posible extraer la clase de un objeto a partir de la RoI si se trata de objetos que no sean muy similares en cuanto a forma, ya que será posible detectar variaciones en la activación del mapa, pero si se trata de objetos similares en forma aumenta el riesgo de confusión.

Por ejemplo, en el caso de una objeto ‘botella’ y otro ‘silla’, será factible clasificarlos teniendo únicamente en cuenta la RoI, ya que en el caso de la botella la actividad se concentrará en forma vertical en el centro de la región, mientras que la silla mostrará mayor activación en las zonas cercanas a los bordes donde generalmente se encontrarán las patas y respaldo.

La matriz de confusión generada (Figura 6) resulta de gran utilidad para comprobar si realmente el modelo está siendo de capaz de clasificar correctamente los objetos de cada clase y poder comprobar cuáles son aquellos que se confunden entre sí con mayor frecuencia.

Para interpretar la matriz de confusión se tiene en cuenta que la diagonal principal representa los aciertos, es decir, los objetos que se han clasificado correctamente como

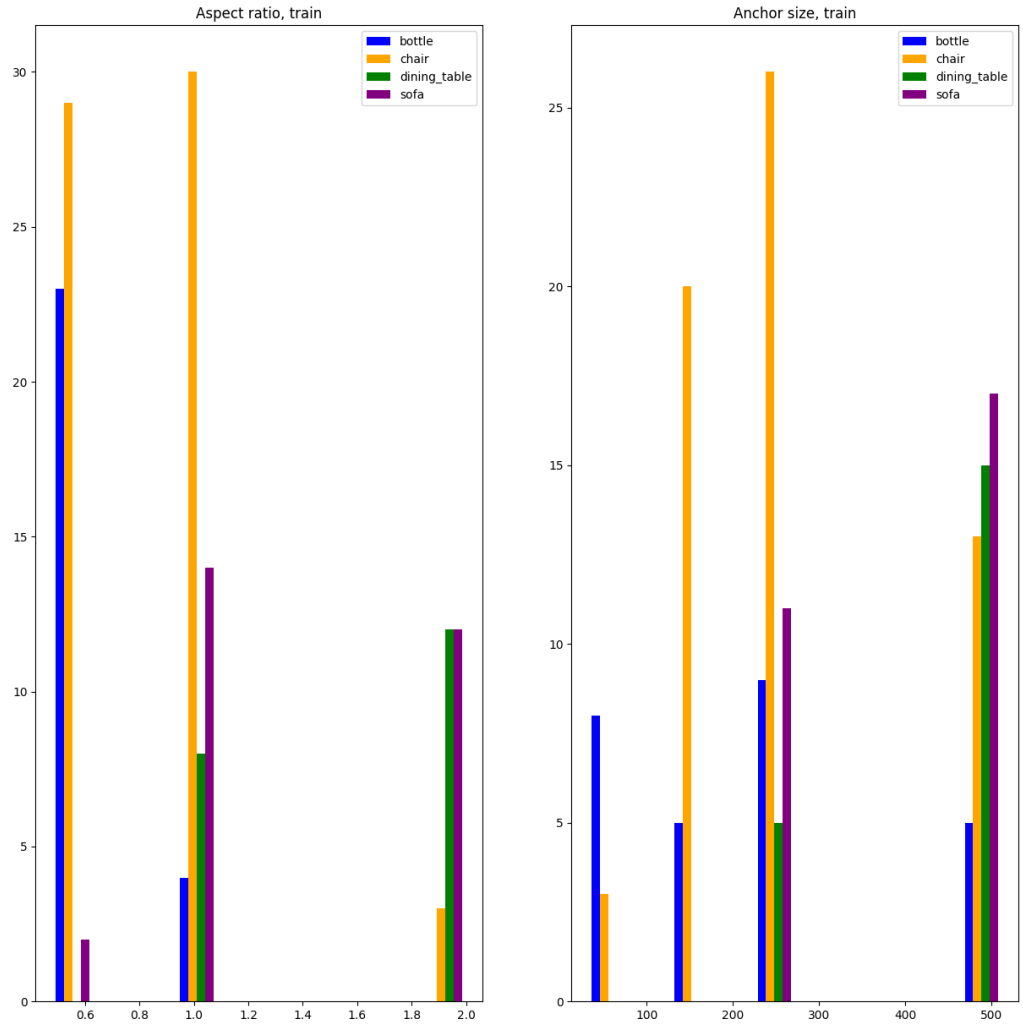


Fig. 5: *Relación entre la relación de aspecto y los anchors*

su clase. En ella primeramente se observa que la clase botella no se ha confundido con las otras clases, mientras que entre las otras tres clases sí que ha habido cierto grado de confusión entre ellas, especialmente entre 'sofá' y 'silla'.

Esto se debe a que tanto 'sofá' como 'mesa' y 'silla' comparten múltiples características al ser todos muebles, tener varias patas sobre las que apoyarse, superficie plana, tamaños similares y poder encontrarse en entornos similares como puede ser un comedor. Por otro lado, la clase 'botella' apenas comparte características con ellas al tratarse de objetos mayormente de otro material, plástico, tamaño por lo general bastante menor y carecer de elementos característicos comunes a las otras clases como las patas.

A fin de mejorar la clasificación, podría ser interesante, ya que se conocen que clases se confunden más, incluir más imágenes de entrenamiento que incluyan objetos de clase 'sofá' y 'silla' desde más ángulos o en mayor variedad de entornos.

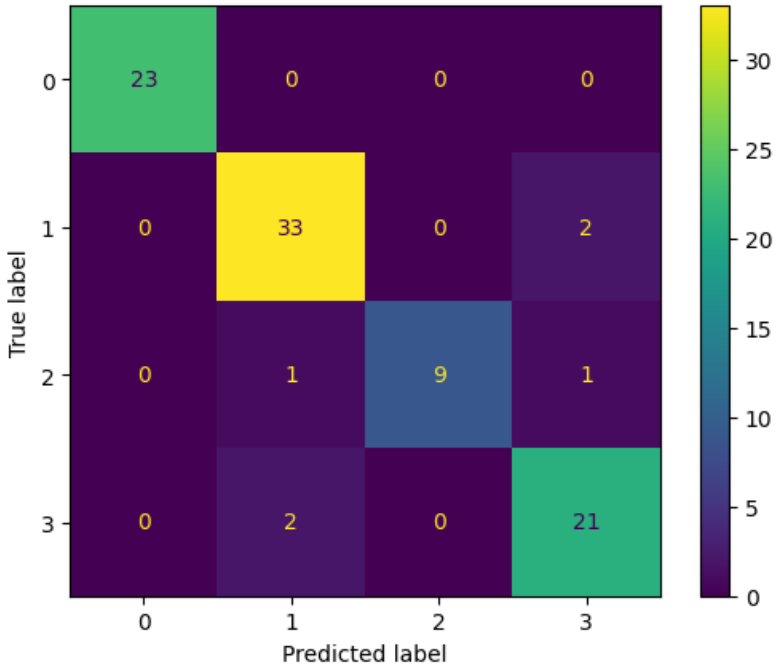


Fig. 6: *Matriz de confusión*