# REDES NEURONALES CONVOLUCIONALES 2

## CLASIFICACIÓN Y LOCALIZACIÓN

La **localización de un objeto en una imagen** puede expresarse como una **tarea de regresión**: para predecir un cuadro delimitador alrededor del objeto, un enfoque habitual es predecir las coordinadas horizontal y vertical del centro del objeto, además de su altura y su anchura. Eso significa que tenemos que predecir cuatro números. No hace falta cambiar mucho el modelo; solo tenemos que añadir una segunda capa de salida densa con cuatro unidades (por lo general, encima de la capa global average pooling) y puede entrenarse utilizando la pérdida MSE:

# Cargar el modelo Xception preentrenado en ImageNet sin la capa densa superior (include\_top=False)

base\_model = tf.keras.applications.xception.Xception(

weights="imagenet", include\_top=False

)

# Capa de Global Average Pooling para reducir la dimensionalidad

avg = tf.keras.layers.GlobalAveragePooling2D()(base\_model.output)

# Capa de salida para la clasificación con activación softmax

class\_output = tf.keras.layers.Dense(n\_classes, activation="softmax")(avg)

# Capa de salida para la regresión de localización (4 valores para coordenadas de bounding box)

loc\_output = tf.keras.layers.Dense(4)(avg)

# Crear el modelo combinando las entradas y las salidas

model = tf.keras.Model(inputs=base\_model.input, outputs=[class\_output, loc\_output])

# Compilar el modelo con múltiples pérdidas y ponderaciones de pérdida

model.compile(

loss=["sparse\_categorical\_crossentropy", "mse"],

loss\_weights=[0.8, 0.2],

optimizer=optimizer, # Asegúrate de que 'optimizer' está definido previamente

metrics={"class\_output": "accuracy", "loc\_output": "mse"},

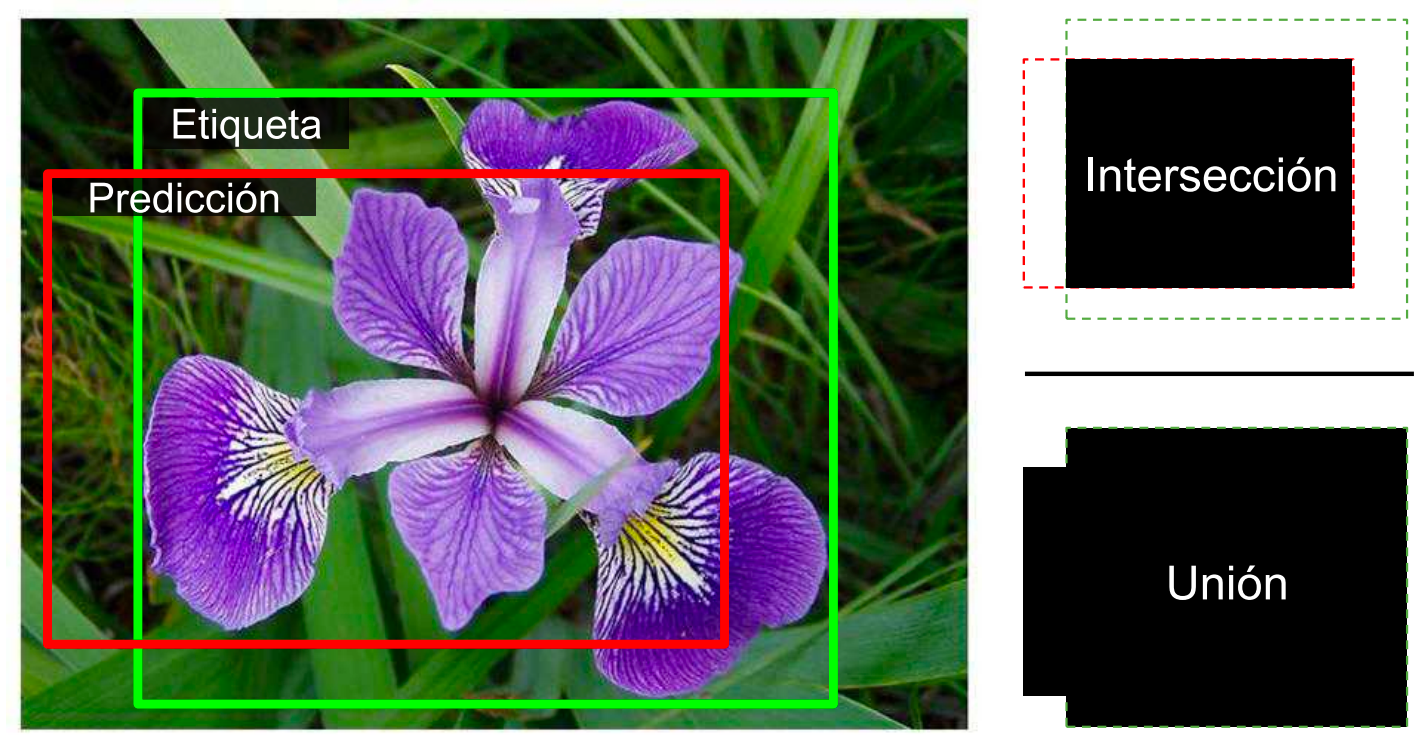
)

Pero, ahora, tenemos un problema: el conjunto de datos de las flores no tiene cuadros delimitadores alrededor de las flores, por lo que tenemos que añadirlos nosotros mismos. A menudo, esa es una de las partes más difíciles y costosas de un proyecto de machine learning: conseguir las etiquetas. Es recomendable dedicar tiempo a buscar las herramientas adecuadas. Para anotar imágenes con cuadros delimitadores, podemos utilizar una herramienta de etiquetado de imágenes de código abierto, como VGG lmage Annotator, Labellmg, Openlabeler o lmglab, o quizá una herramienta comercial como LabelBox o Supervisely.

**Supongamos que hemos obtenido los cuadros delimitadores para cada imagen del conjunto de datos de las flores** (por ahora, vamos a dar por hecho que hay un solo cuadro delimitador por imagen). Hay que crear un conjunto de datos cuyos elementos sean lotes de imágenes preprocesadas junto con sus etiquetas de clase y sus cuadros delimitadores. **Cada elemento debería ser una tupla con la forma (images, (class\_labels, bounding\_boxes)).**

**Los cuadros delimitadores deberían normalizarse de manera que las coordenadas horizontal y vertical, así como la altura y la anchura, vayan de 0 a 1.**

A menudo, el MSE funciona bastante bien como función de pérdida para entrenar el modelo, pero no es la mejor métrica **para evaluar lo bien que puede predecir el modelo los cuadros delimitadores. La métrica más común para esto es la "intersección sobre unión" (loU, intersection over union**): el área de solapamiento entre el cuadro delimitador predicho y el cuadro delimitador objetivo, dividida entre el área de su unión. **En Keras, la implementa la clase tf.keras.metrics.MeanIoU.**



*Métrica loU para cuadros delimitadores*.

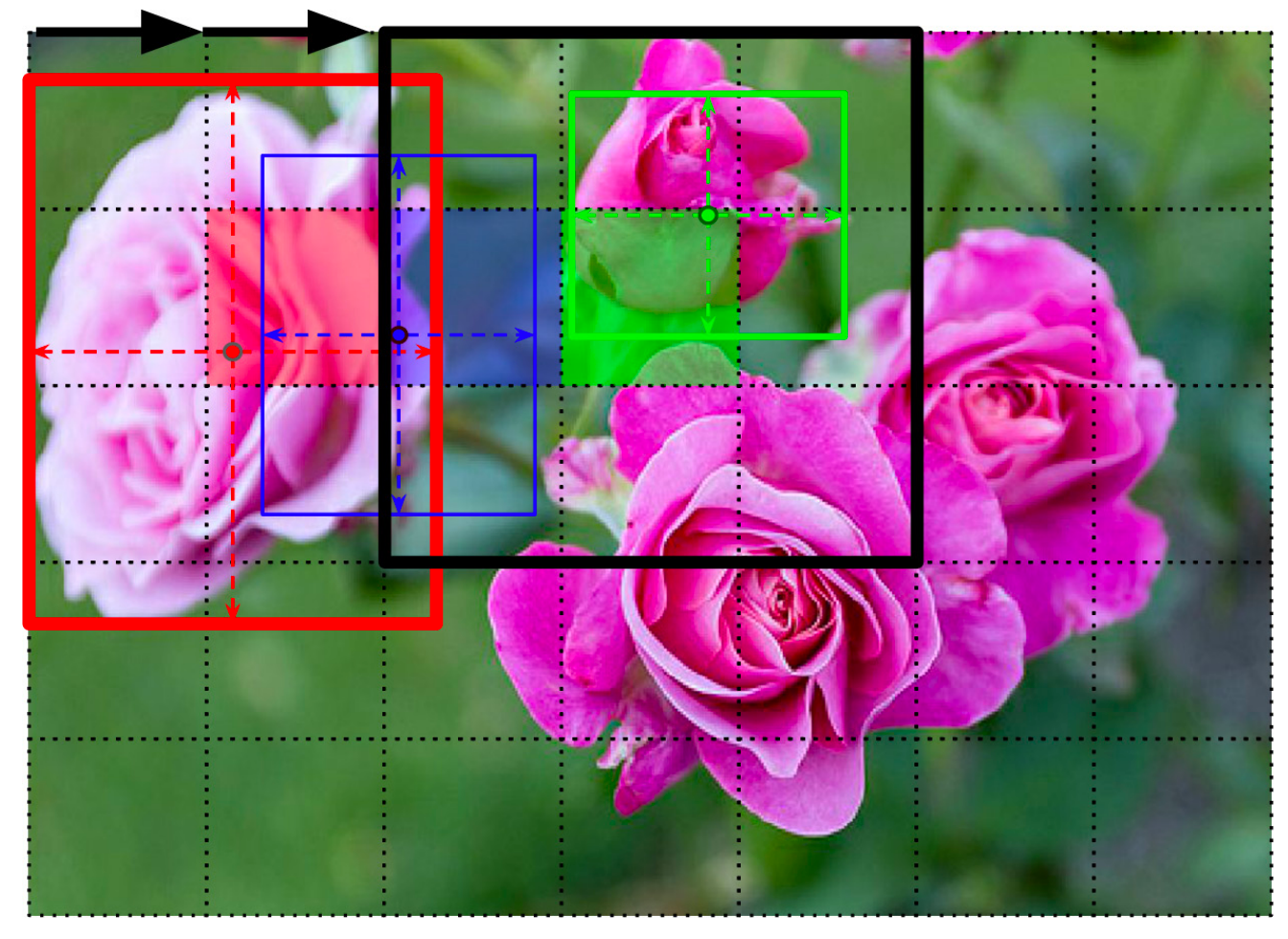
Clasificar y localizar un solo objeto está bien, pero ¿qué pasa si las imágenes contienen varios objetos (como suele ocurrir en el conjunto de datos de las flores)?

## DETECCIÓN DE OBJETOS

**La tarea de clasificar y localizar múltiples objetos en una imagen se denomina "detección de objetos".** Hasta hace unos años, un enfoque común era coger una CNN entrenada para clasificar y localizar un solo objeto más o menos centrado en la imagen y, después, deslizarla través de la imagen y hacer predicciones en cada paso.

Por lo general, **la CNN se entrenaba para predecir no solo las probabilidades de clases y un cuadro delimitador, sino también una "puntuación del carácter de objeto":** se trata de la probabilidad estimada de que la imagen contenga realmente un objeto centrado cerca del medio. Es una **salida de clasificación binaria**; puede producirla una capa de salida densa con una sola unidad, utilizando la función de activación sigmoide y entrenarse usando la pérdida de entropía cruzada binaria.

En el ejemplo, la imagen se ha troceado en una cuadrícula de 5 x 7, y vemos una CNN (el cuadro negro grueso) que se desliza por todas las regiones de 3 x 3 y realiza predicciones en cada paso.



*Detección de múltiples objetos al deslizar una RNC por la imagen.*

Este acercamiento sencillo a la detección de objetos funciona bastante bien, pero requiere ejecutar la CNN muchas veces (en este ejemplo, 15 veces), así que es bastante lento.

Hay muchos modelos de detección de objetos disponibles en TensorFlow Hub, a menudo con pesos preentrenados, como YOLOv11, SSD, Faster R-CNN y EfficientDet

<https://www.youtube.com/watch?v=3h1_-AaVjWY>

Hasta ahora, solo hemos considerado la detección de objetos en imágenes individuales, pero ¿qué pasa con los vídeos? Los objetos no solo deben detectarse en cada fotograma, sino que deben rastrearse a lo largo del tiempo. Vamos a echar un vistazo rápido al seguimiento de objetos.

## SEGUIMIENTO DE OBJETOS

El seguimiento de objetos es una tarea difícil: los objetos se mueven, pueden crecer o reducirse a medida que se acercan o se alejan de la cámara, su aspecto puede cambiar si giran o se mueven en unas condiciones de iluminación o con fondos diferentes, pueden quedar tapados por otros objetos, etc.

<https://www.youtube.com/watch?v=oig4o9RW_aM&t=615s>

Hasta ahora, hemos ubicado objetos utilizando cuadros delimitadores. A menudo, eso es suficiente, pero, en ocasiones, necesitamos ubicar objetos con una precisión mucho mayor, por ejemplo, para eliminar el fondo detrás de una persona durante una videoconferencia. Vamos a ver cómo llegar al nivel de los píxeles.

## SEGMENTACIÓN SEMÁNTICA

En la segmentación semántica, cada píxel se clasifica según la clase del objeto al que pertenezca (por ejemplo, carretera, coche, peatón, edificio, etc.). Sin embargo, los objetos diferentes de la misma clase no se distinguen.



*Segmentación semántica.*

La principal dificultad en esta tarea es que, cuando las imágenes pasan por una CNN corriente, van perdiendo de manera gradual su resolución espacial (debido a las capas con pasos de avance mayores que 1); por tanto, una CNN corriente puede acabar sabiendo que hay una persona en alguna parte en la parte inferior izquierda de la imagen, pero no será mucho más precisa que eso.

Hay muchos enfoques diferentes para abordar este problema: por ejemplo, la arquitectura Mask R-CNN.

<https://datasmarts.net/es/como-segmentar-imagenes-con-tensorflow-hub/>

La segmentación de instancias es similar a la segmentación semántica, pero, en vez de combinar todos los objetos de la misma clase en un bulto grande, cada objeto se distingue de los demás (por ejemplo, identifica cada bicicleta individual).

El campo de la visión por ordenador profunda es amplio y avanza deprisa, con todo tipo de arquitecturas nuevas cada año. Casi todas ellas se basan en redes neuronales convolucionales, pero, desde 2020, otra arquitectura de red neuronal ha entrado en el espacio de la visión por ordenador: los transformers. Los progresos que se han hecho en la última década han sido impresionantes y los investigadores están centrándose ahora en problemas cada vez más difíciles, como el aprendizaje antagónico (que intenta hacer que la red sea más resistente a imágenes diseñadas para engañarla), la explicabilidad (entender por qué la red hace una clasificación específica), la generación de imágenes realistas, el aprendizaje one shot (un sistema que puede reconocer un objeto tras haberlo visto solo una vez), la predicción de los siguientes fotogramas de un vídeo, la combinación de tareas de texto y de imágenes, y mucho más.