VISIÓN ARTIFICIAL
Práctica 4: Segmentación de imágenes

El principal objetivo de esta práctica es crear una aplicación de segmentación de imágenes utilizando un modelo neuronal de segmentación semántica ya entrenado. A continuación, se describen los detalles concretos de esta práctica.

Introducción

Las técnicas de segmentación semántica permiten asociar cada píxel de una imagen a una categoría concreta, suponiendo un conjunto limitado y conocido de categorías. Estas técnicas se basan en el uso de redes neuronales profundas para, a partir de grandes conjuntos de datos, aprender la relación entre una imagen de entrada sin procesar (datos crudos) y la imagen segmentada. Uno de los modelos de segmentación semántica más conocidos es FCN (Fully Convolutional Networks) (figura 1), formada por una secuencia de capas convolucionales (detección de características) cuya salida se encuentra conectada a capas deconvolucionales (extrapolación - *upsampling*) que proporcionan el resultado final. Este será el modelo neuronal que utilizaremos en esta práctica.

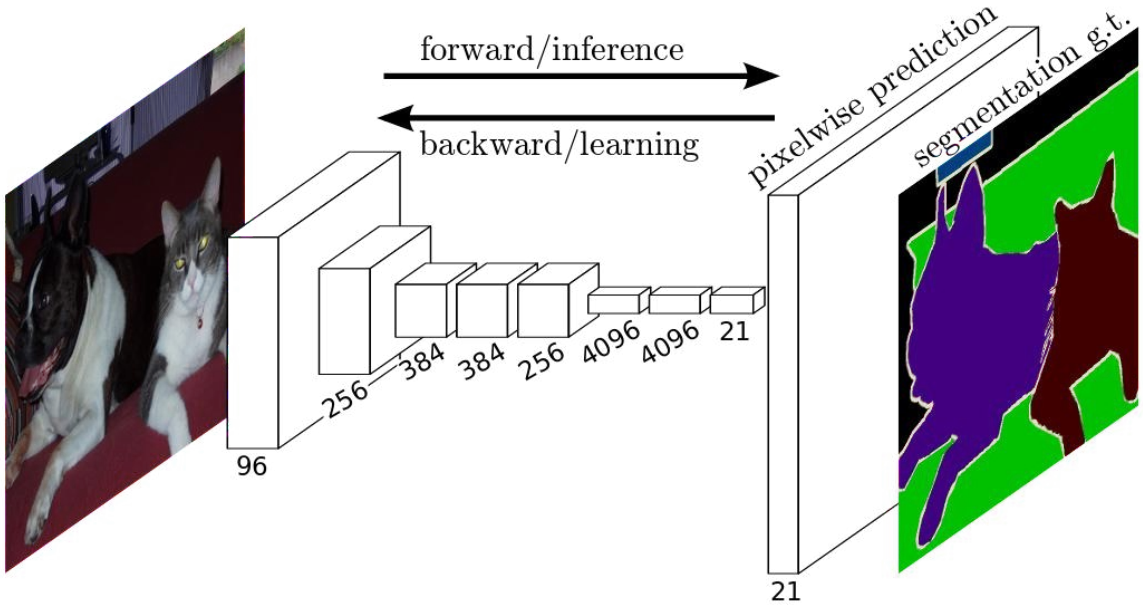

Figura 1: Arquitectura de FCN y esquema de aprendizaje.

FCN está entrenada para distinguir entre 21 categorías relacionadas con interiores y exteriores (bicicleta, coche, persona, perro, gato, …). Dentro de estas 21 categorías, se encuentra la categoría *background*, que es la asociada con aquellos píxels que no se pueden clasificar en ninguna de las otras categorías. El resultado proporcionado por la red es una imagen, con el mismo alto y ancho que la imagen de entrada, formada por 21 planos. En cada uno de estos planos, cada píxel tiene asociado un valor de tipo *float*. Estos valores indican la puntuación asignada al píxel para cada categoría. La categoría final de un píxel se obtiene como aquella para la que el valor asignado es máximo.

Especificación de objetivos y desarrollo

Módulo *dnn* de OpenCV

Para la implementación de esta práctica necesitamos utilizar el módulo *dnn* de OpenCV. Debemos por lo tanto incluir la librería en nuestro proyecto, añadiendo la opción ***-lopencv_dnn*** a la línea ***LIBS*** del fichero ***proyVA.pro***. Asimismo, es necesario incluir el fichero ***opencv2/dnn/dnn.hpp*** en ***mainwindow.h*** (*#include opencv2/dnn/dnn.hpp*). El espacio de nombres definido en esta librería es *dnn*. Para usar cualquier clase o función proporcionada por ella es necesario precederla de ***dnn::*** o bien incluir en *mainwindow.h* la sentencia "*using namespace dnn;*"

Creación y uso del modelo neuronal

Como se ha indicado anteriormente, el objetivo principal de la práctica es crear una aplicación que proporcione un resultado de segmentación semántica sobre imágenes. Para ello, se utilizará el modelo FCN, cuyos ficheros se proporcionan como parte del material de la práctica. Estos ficheros son los siguientes:

- *fcn.prototxt* y *fcn.caffemodel*: ficheros de parámetros y de pesos, respectivamente. El modelo se encuentra entrenado con Caffe. Estos son los que ficheros que deben indicarse para cargar el modelo utilizando la función *readNetFromCaffe* de OpenCV.
- *fcn-classes.txt*: fichero de texto con los nombres de las categorías.
- *fcn-colors.txt*: fichero de texto con combinaciones RGB para representar visualmente cada categoría.

Una vez creada la red (objeto de la clase *Net* del módulo *dnn* de OpenCV) con los ficheros del modelo, ésta puede ser usada para inferencia utilizando los métodos *setInput* y *forward*. La imagen de entrada que debe ser indicada mediante el método *setInput* tiene que estar representada en formato BGR. A los atributos BGR de cada píxel se les debe restar el valor medio BGR de la imagen. Este procesamiento puede llevarse a cabo mediante la función *blobFromImage* de OpenCV. Concretamente, los parámetros que debemos indicar en esta función para construir la entrada de la red son:

- *size*: tamaño de imagen que se utilizará para su segmentación (coincidirá con el tamaño de la salida de la red).
- *mean*: valor medio de cada canal (R, G y B) que se restará a cada píxel (la media de los 3 atributos puede obtenerse mediante la función *mean* de OpenCV).
- *swapRB*: debe ponerse a *true* para intercambiar los canales *R* y *B*, obteniendo así una representación en formato BGR.

Tras la llamada a *setInput*, el método *forward* proporciona el resultado de aplicar la imagen de entrada al modelo neuronal. Concretamente, dicho resultado se devuelve en un objeto de la clase *Mat* con 4 dimensiones (1ª dimensión:1, 2ª dimensión: número de categorías (21), 3ª dimensión: alto de imagen, 4ª dimensión: ancho de imagen). El ancho y alto de la imagen resultante es el mismo que el de la imagen de entrada.

Obtención de la imagen segmentada

La salida de la red, como ya se ha comentado, no proporciona directamente la categoría de cada píxel, sino que devuelve, por cada uno de ellos, la puntuación obtenida en cada una de las categorías. Para obtener la imagen segmentada, es necesario asociar a cada píxel el identificador de la categoría a la que pertenece. Dicha categoría será la que maximice la puntuación. Así, una vez obtenida la salida de la red, será necesario crear la imagen segmentada (*segmentedImg*) siguiendo un proceso en el que por cada píxel se localice la categoría de máxima puntuación y se asigne a dicho píxel el identificador (índice) de dicha categoría.

Generación del resultado visual de la segmentación

La imagen segmentada obtenida del procesamiento anterior no se puede visualizar directamente, ya que el valor asociado con cada píxel no es más que un identificador. No obstante, a partir de esta imagen y de los colores asociados con las diferentes categorías (contenido del fichero *fcn-colors.txt*), es posible generar una imagen RGB en la que cada objeto detectado aparezca representado por un color identificativo. Para ello, el identificador de cada píxel de la imagen segmentada se utilizará para acceder a la tabla de colores de categorías para obtener los valores de RGB de la categoría asociada. Dichos valores de RGB se asignaran al píxel correspondiente de la nueva imagen.

La imagen obtenida contiene ya un resultado visual. Este resultado visual da una idea de la precisión de la segmentación, pero que no es del todo completa. Para proporcionar una idea más precisa del resultado, permitiremos al usuario combinar el resultado visual de la segmentación con la imagen original. Esto se realizará aplicando una suma ponderada de las 2 imágenes de manera que los pesos asociados a cada imagen sumen 1 (ver figura 2).

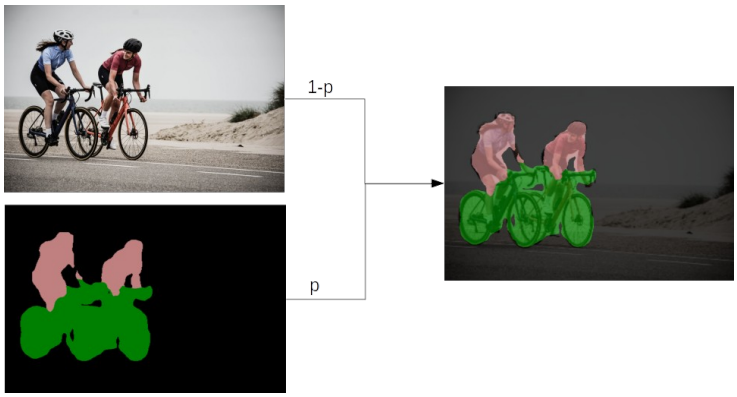

Figura 2: Generación del resultado visual final. La imagen de color de categorías y la imagen original son combinadas mediante una suma ponderada. El peso asociado a la imagen de categorías es "*p*", con valor entre 0 y 1. El peso que se aplica a la imagen original se obtiene como "*1-p*".

Interfaz gráfica de usuario

La figura 3 muestra un ejemplo de interfaz gráfica de usuario para la práctica. La segmentación se llevará a cabo sobre la imagen RGB obtenida de la captura actual de cámara o cargada desde fichero. El resultado visual se mostrará en el visor de la derecha tras combinar la imagen de categorías y la imagen original tal y como se comentó anteriormente. La segmentación sólo se aplicará cuando el usuario pulse sobre el botón "Segment Image", ya que el proceso no proporciona resultados en tiempo real a menos que se ejecute sobre GPU.

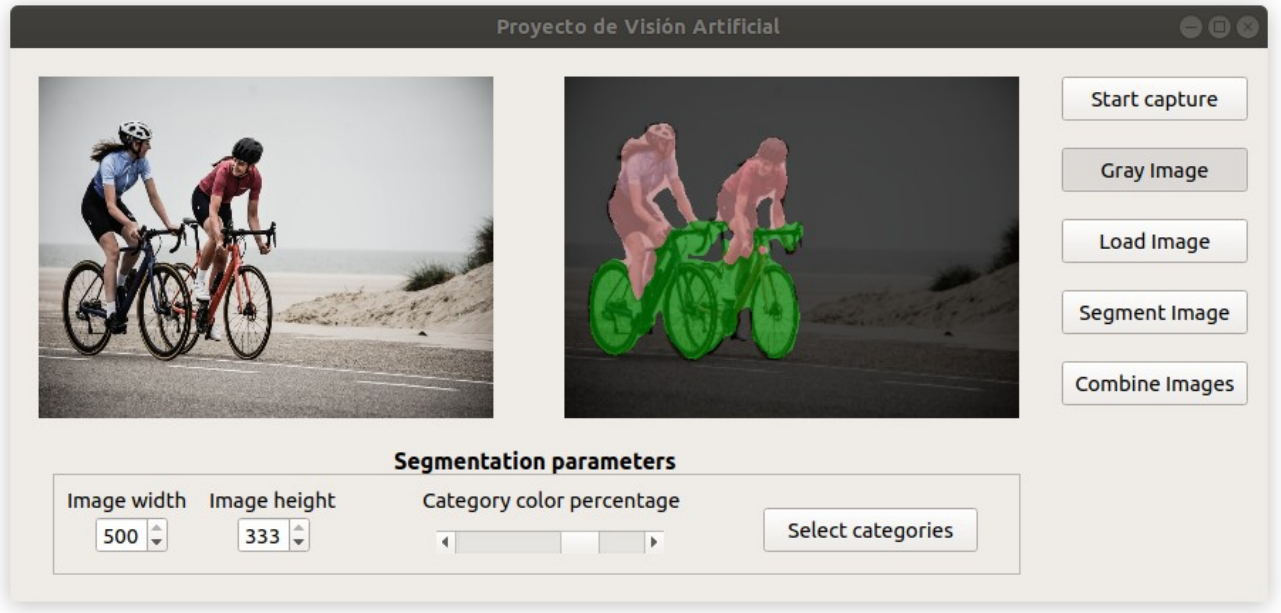

Figura 3: Ejemplo de interfaz gráfica de usuario para la práctica.

Las opciones incluidas en “Segmentation parameters” permitirán al usuario configurar cómo se llevará a cabo el proceso de segmentación y la visualización de resultados. Los valores de “Image width” e “Image height” indican las dimensiones que debe tener la imagen cuando actúa de entrada a la red. Los valores máximos de las dimensiones deben fijarse a 320x240 cuando la imagen es la recibida a través de la cámara. En caso de tratarse de una imagen de fichero, estos valores se fijarán teniendo en cuenta las dimensiones de la imagen cargada y considerando que, como máximo, ancho y alto no pueden ser superiores a 500 (esto evitará que el proceso de inferencia de la red sea excesivamente lento). Así, en la opción “Load Image”, además de cargar una imagen de fichero en *colorImage*, deberán modificarse los 2 elementos que indican las dimensiones utilizadas como entrada de la red teniendo en cuenta el tamaño de la imagen cargada. Si una de las dimensiones es mayor que 500, se fijara dicha dimensión a 500 y se calculará la otra de manera que se mantenga la relación de aspecto.

La barra de desplazamiento etiquetada como “Category color percentage” indica el peso que debe asignarse a la imagen de categorías (valor de p en el esquema de la figura 2) para componer la imagen final que se visualizará en el visor de la derecha. Esta barra proporciona un rango de valores entre 0 y 100. Por lo tanto, para calcular el peso final, será necesario dividir el valor actual de este elemento de la interfaz entre 100.

Por último, el botón “Select categories” da acceso a un diálogo que permite al usuario seleccionar las categorías que deberán ser detectadas (ver figura 4). Las categorías que no estén seleccionadas en este diálogo deberán tratarse de la misma forma que el fondo (categoría *background*, con identificador 0). Es decir, si para un píxel la categoría para la que la red asigna el máximo valor es una categoría no activa en el diálogo, deberá asignarse el identificador 0 para ese píxel en la imagen segmentada (*segmentedImg*).

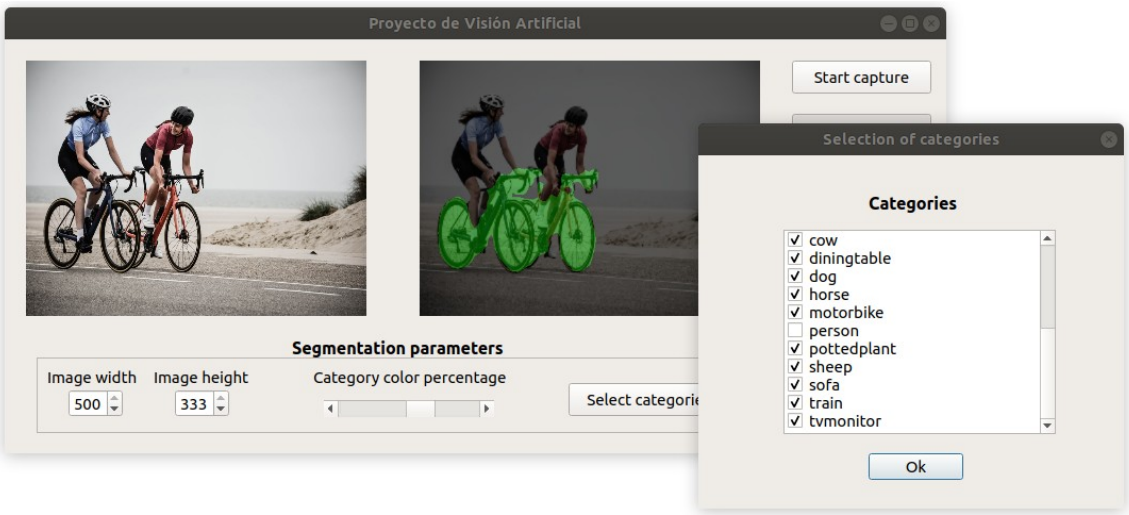

Figura 4: Diálogo de selección de categorías.

Ampliaciones

Se propone como ampliación aplicar el resultado de la segmentación a la copia de objetos de una imagen a otra imagen en las posiciones indicadas por el usuario. En la interfaz de la figura 3, esta opción está disponible a través del botón “Combine Images”. Al pulsar sobre este botón, se abre un nuevo diálogo como el que se muestra en las figuras 5, 6 y 7. Este diálogo contiene un visor para una nueva imagen (imagen combinada) de tamaño 640x480 y 3 botones. El primer botón (“Load background”) permite cargar una imagen de fichero que actuará como fondo en la imagen combinada (figura 5). La imagen cargada se representará en RGB y se redimensionará a 640x480. Con el botón “Paste” el usuario podrá copiar un objeto seleccionado de la imagen original (*colorImage*) a partir de una determinada posición de la imagen combinada. Para ello, el usuario deberá seleccionar una ventana en el visor de *colorImage* y, a continuación, indicar una posición del visor de la imagen combinada. Una vez indicada esa posición, el programa combinará la imagen de fondo actual con los píxels de la ventana de *colorImage* que pertenezcan a una categoría con identificador mayor que 0 (categoría distinta de *background*). El efecto de este procesamiento puede verse en la figura 6. Si el usuario selecciona otra posición, se descartará la copia previa y se copiará de nuevo la ventana a la nueva posición. Una vez que el usuario pulse sobre el botón “Paste” la copia se tomará como definitiva.

El diálogo incluye un tener botón “Save to file” mediante el cual el usuario podrá guardar en fichero la imagen combinada que aparezca actualmente en el visor. Deberá guardarse en BGR con resolución de 640x480.

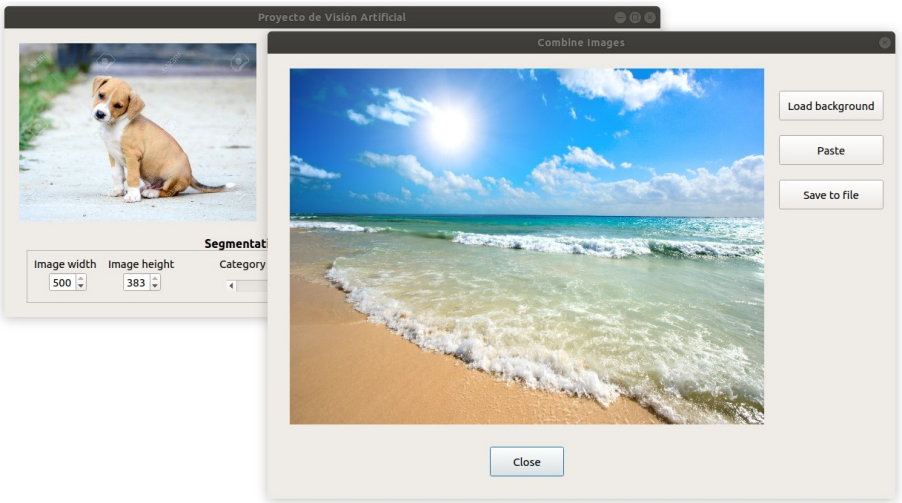

Figura 5: Nuevo diálogo para combinar imágenes. Resultado de aplicar la opción “Load background”.

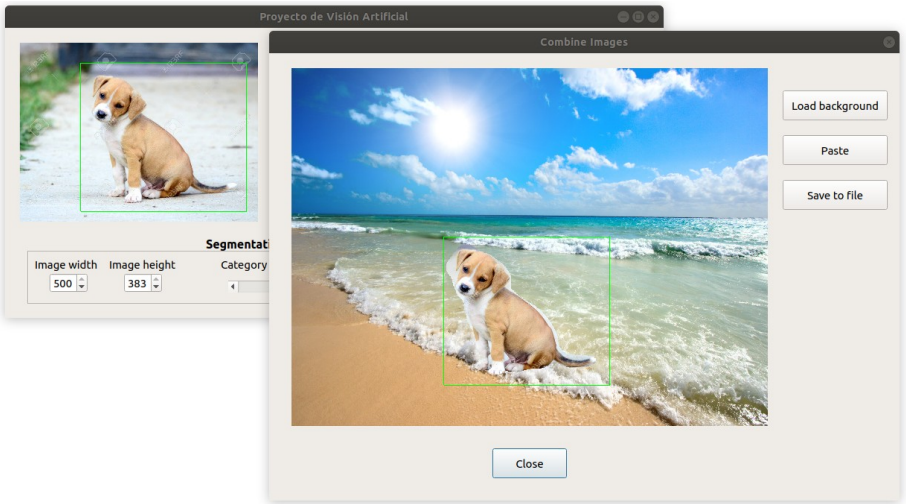

Figura 6: Resultado de seleccionar una posición de la imagen combinada tras la selección de ventana en el visor de *colorImage*.

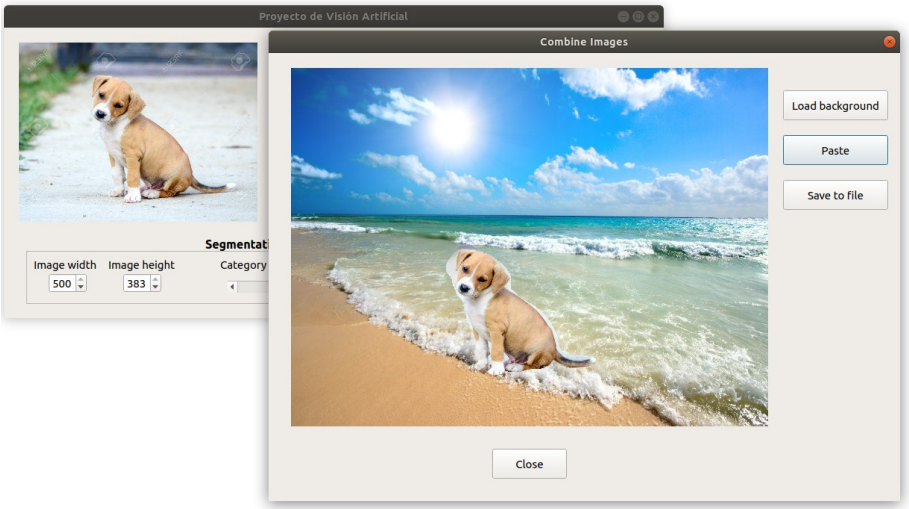

Figura 7: Resultado de ejecutar la opción “Paste”.