Lab2: Video Object Detection

Javier Vinuesa

I. INTRODUCTION

Esta práctica consiste en la detección de objetos en vídeo. La práctica se va a dividir en dos sesiones. En la primera de estas, descargaremos un código de GitHub para la detección de objetos e implementaremos las instrucciones de como ejecutarlo. Además creearemos un repositorio en GitHub donde subiremos estas instrucciones. En la segunda sesión de la práctica implementaremos otro método para la detección de objetos, basado en el método de la primera práctica. Una vez implementado, utilizaremos los dos métodos para procesar distintos vídeos pertenecientes a la base de datos UCF 101 [2] y compararemos los resultados obtenidos. Finalmente elegiremos el mejor método de los dos.

II. METHOD

Para la primera sesión de la práctica partimos de un repositorio de GitHub [1]. En este repositorio se encuentra el código para realizar detecciones en vídeo a partir del método YoloV2 + SeqNms. En este repositorio venia paso por paso como ejecutar el código. El problema es, que estos pasos no estaban completos, por lo que hemos tenido que ir completándolos por nuestra cuenta, para finalmente poder ejecutar el código y obtener nuestras detecciones. La segunda parte de esta primera sesión consistía en crear un repositorio propio de GitHub, en el cual explicásemos de manera correcta (no como en el repositorio del que partíamos) como ejecutar el código para la detección de objetos en vídeo. Por lo tanto los objetivos principales para esta primera parte de la práctica son: familiarizarse con la utilización de código de repositorios como GitHub, con la utilización de código de terceros y por último con la compartición de codigo.

La segunda sesión de la práctica consistía en la implementación del método de detección de objetos en vídeo YoloV2. Para ello se parte del código desarrollado en la primera sesión de la práctica. La segunda tarea de esta sesión es la comparación de resultados entre el método utilizado en la primera sesión y el método utilizado en esta. Por lo tanto el objetivo para esta segunda sesión de la práctica es la familiarización con el uso de los métodos Seq-Nms y YoloV2

A. YoloV2 + SeqNms

Este es el método para la detección de objetos en vídeo que hemos utilizado en la primera parte de la práctica. Consiste en la combinación de dos técnicas. En primer lugar vamos a ver como funciona Yolo. Yolo es un método Unifed o One Stage de detcción de objetos en imagen. Esto quiere decir que realiza en el mismo paso tanto la detección como la clasificación del objeto. Para ello Yolo divide el frame en un grid y asigna a cada celda del grid una clasificación. Este tipo de métodos son menos precisos que los two stages, pero tienen como ventaja un menor coste computacional. Una vez realizada la detección con Yolo, se ha aplicado un método de post procesado conocido como SeqNms. Este método soluciona problemas como la detección del mismo objeto varias veces en un frame. Para ello se queda con las detecciones con mayor confianza de cada objeto en cada una de las frames (siempre que superen cierto umbral) y realiza la unión de estas en el tiempo. El resto de detecciones las elimina. De esta manera se consiguen tener unas detecciones únicas y robustas por cada objeto.

B. YoloV2

Este es el método para la detección de objeto en vídeo que hemos implementado en la segunda sesión de la práctica. Es el miso método explicado en la sección II.A pero simplificado. La simplificación viene dada porque en esta segunda sesión no hemos realizado el método de post-procesado Seq-Nms y simplemente nos hemos quedado con el resultado que nos aportaba Yolo.

III. IMPLEMENTATION

A. Ejecución del Código del Repositorio

En este apartado vamos a explicar como ejecutar el código del repositorio GitHub [1] de manera correcta.
En primer lugar vamos a descargar el código del repositorio. A continuación vamos a crear y activar un entorno virtual con Python 3.6. Para ello nos movemos al directorio /opt/anaconda3.7 y ejecutamos este código por terminal:
conda create –name envname python=3.6
conda activate envname
En segundo lugar vamos a ir al archivo MakeFile, dentro de la carpeta seq_yolo_nms decargada y vamos a descativar los flags de OPENCV y CUDDN poniendo a '0' estas variables. A continuación, en el mismo MakeFile, debemos cambiar las rutas COMMON+= -DGPU -I/usr/local/cuda-8.0/include/ y LDFLAGS+= -L/usr/local/cuda-8.0/lib64 -lcuda -lcudart -lcublas –lcurand por COMMON+= -DGPU -I/usr/local/cuda-10.1/include/ y LDFLAGS+= -L/usr/local/cuda-10.1/lib64 -lcuda -lcudart -lcublas –lcurand. Seguido de esto debemos exportar un path intoduciendo por terminal: export PATH=/usr/local/cuda-10.1/bin$PATH:+:$PATH. Una vez realizados estos pasos, ejecutamos el make.
En tercer lugar, una vez realizado el make, vamos a descargar los pesos de nuestros modelos. Para ello introduciremos por terminal los siguientes comandos:

wget https://pjreddie.com/media/files/yolo.weights
wget https://pjreddie.com/media/files/yolov2-tiny-voc.weights
Como cuarto paso nos descargaremos un archivo de video .mp4 y lo copiaremos en la carpeta 'video'. A continuación vamos al archivo video2img.py y ponemos paréntesis a los print de las líneas 6 y 39. Antes de ejecutar este archivo debemos descargarnos la librería OpenCv con el siguiente comando: conda install -c conda-forge opencv. Una vez arreglados los print e instalada la librería, nos movemos a la carpeta de video y ejecutamos el archivo con el comando: python video2img.py -i nombre_video.mp4. Seguido de esto ejecutamos: python get_pkllist.py.
En el quinto paso volvemos a la carpeta seq_nms_yolo . En el archivo yolo_seqnms.py debemos poner los paréntesis a los print de las líneas: 46, 90, 104, 264, 270, 276, 290 y 292. Además deberemos cambiar el 'import cPickle as pickle' por 'import pickle'. También deberemos instalar la librería matplotlib con el siguiente comando: pip install matplotlib. En el archivo yolo_detection.py deberemos poner paréntesis a los print de las líneas: 134, 135, 146, 160 y 161. En este mismo archivo deberemos introducir un 'import os' y cambiar la línea 37 (lib = CDLL("libdarknet.so", RTLD_GLOBAL)) por lib = CDLL(os.path.join(os.getcwd(), "libdarknet.so"), RTLD_GLOBAL).
Una vez realizado estos pasos intoduciremos por terminal los export: 'export PATH=/usr/local/cuda-10.1/bin$PATH:+:$PATH', 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-10.1/lib64', 'export LIBRARY_PATH=$LIBRARY_PATH:/usr/local/cuda-10.1/lib64'
Una vez exportadas las librerías instalaremos el modulo de object detection con el siguiente comando: pip install tensorflow-object-detection-api. A continuación deberemos modificar el archivo yolo_detection.py cambiando las líneas net = load_net(cfg, weights, 0), meta = load_meta(data) e im = load_image(image, 0, 0) por net = load_net(bytes(cfg, 'utf-8'), bytes(weights, 'utf-8'), 0), meta = load_meta(bytes(data, 'utf-8')) y im = load_image(bytes(image, 'utf-8'), 0, 0) para que estas tengan las variables que utlizan en formato byte.
Después de esto debemos ir a la línea 116 del archivo label_map_util.py y sustituir tf.gfile.GFile(path, 'r') por tf.io.gfile.GFile(path, 'r').
A continuación deberemos instalar el módulo 'imageio' con el siguiente comando: conda install -c menpo imageio. Desupués nos iremos a la línea 291 del archivo yolo_seqnms.py y sustituímos scipy.misc.imsave() por imageio.imwrite().
Una vez realizados todos estos cambios ya podremos ejecutar el archivo yolo_seqnms.py con el comando: python yolo_seqnms.py. Esto nos va a generar las frames con detección en la capeta video/output.
Si queremos reconstruir el vídeo debemos movernos a la carpeta de vídeo y ejecutar el comando: python img2video.py -i output. Antes de ejecutarlo deberemos poner los paréntesis de los print en las líneas 6, 13 y 30. Además para que se visualice la detección en el vídeo de salida hay que modificar la línea 43 del archivo yolo_seqnms.py poniendo 'if box[0]==cls.encode('utf8'):' en vez de 'if box[0]==cls'
Una vez realizadas estas modificaciones podremos ejecutar el comando y el vídeo con detecciones nos aparecerá en la carpeta de vídeo con el nombre 'output.mp4'.

B. Implementación del código YoloV2

En esta sección vamos a explicar como realizar el método utilizado en la segunda sesión de prácticas. La realización de este código es muy sencilla. Partimos del código Yolo+SeqNms utilizado en la anterior sesión. Para realizar el método simplificado Yolo, sin utilizar el post-procesado, basta con comentar las líneas 207 y 208 (son la segunda y tercera línea de la función 'dsnms(res)') del archivo yolo_seqnms.py. De esta forma las detecciones se realizarán sin el método de post-procesado. Para ejecutar el código y obtener los resultados hay que hacer lo mismo que en la anterior sesión de la práctica:
1. Descargar un vídeo, meterlo en la carpeta de vídeo y desde esa carpeta ejecutar 'video2img.py -i input.mp4' y a continuación 'python get_pkllist.py'.
2. Volvemos a la carpeta yolo_seq_nms y ejecutamos 'python yolo_seqnms.py'.
3. Nos movemos a la carpeta de vídeo y ejecutamos 'python img2video.py -i output'
. Una vez realizados estos tres pasos el vídeo con las detecciones aparecera en la carpeta de vídeo con nombre 'output.mp4'.

IV. DATA

El único dato que he utilizado en la primera sesión de la práctica ha sido el video 'v_BalanceBeam_g18_c01.avi' correspondiente a la base de datos UCF101 [2]. A este vídeo es al que le hemos realizado las detecciones de objeto.
En la segunda sesión de la práctica se han utilizado más vídeos con el objetivo de comparar los resultados de ambos modelos. Los vídeos que se han utilizado son también de la base de datos UCF101 [2] y son los siguientes: v_ApplyEyeMakeup_g19_c03.avi, v_ApplyLipstick_g21_c01.avi, v_Archery_g07_c01.avi, v_BabyCrawling_g11_c01.avi, v_BalanceBeam_g18_c01.avi".

V. RESULTS AND ANALYSIS

Los resultados de la primera sesión de la práctica son bastante buenos. Con este método detectamos todas las personas que aparecen en el vídeo 'v_BalanceBeam_g18_c01.avi' de manera correcta. La Figura 1 es un ejemplo de un frame del vídeo obtenido.

En la segunda sesión de la práctica hemos realizado la comparación entre ambos métodos. Si utilizas la velocidad normal del vídeo, ambaos métodos parece que realizan detecciones bastante similares. Esto es lógico ya que ambos estan realizando las detecciónes con el mismo método, solo que en la segunda sesión no estamos refinando el resultado. Sin embargo, si empiezas a pasar los frames a una velocidad más lenta, se puede observar como el método que no utiliza

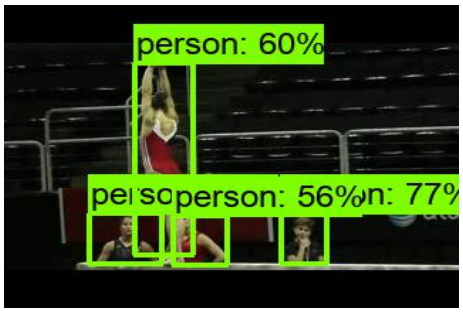
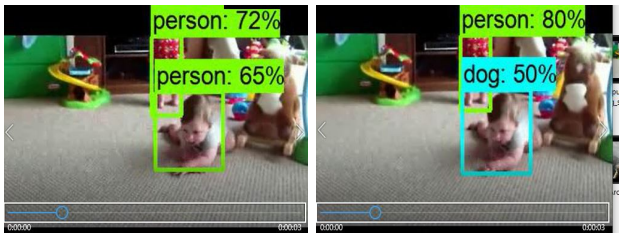


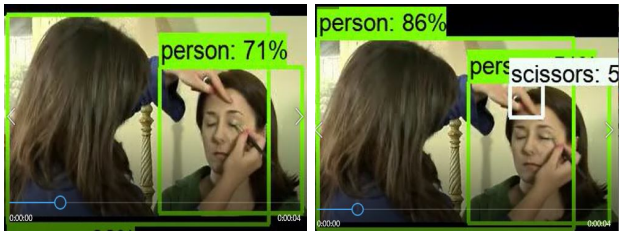
Fig. 1: Ejemplo de un Frame del vídeo 'v_BalanceBeam_g18_c01.avi' cuando se le ha aplicado el método YoloV2 + seqnms

el método de post procesado, realiza peores detecciones. En el método de la primera sesión esto no pasaba, ya que con el seq_nms eliminábamos las detecciónes que estaban por debajo de cierto umbral, cosa que ahora no estábamos haciendo. Por este motivo, hay detecciones con el método Yolo (segunda sesión) que no tienen sentido. Esto se puede ver en los ejemplos de las figuras 2, 3 y 4.



(a) Yolo+SeqNms (b) Yolo

Fig. 2: Comparación del método Yolo+SeqNms con el método Yolo en el vídeo BabyCrawling.avi



(a) Yolo+SeqNms (b) Yolo

Fig. 3: Comparación del método Yolo+SeqNms con el método Yolo en el vídeo ApplyMakeUP.avi

Como se puede observar en las figuras 2, 3 y 4, el método Yolo+SeqNms consigue mejores resultados. El Yolo al no refinar en el post-procesado las detecciónes, obtiene errores en detección muy burdos, como confundir a una persona con un snowboard (Figura 4), unas manos con unas tijeras (Figura 3) y un bebé con un perro (Figura 2).

(a) Yolo+SeqNms (b) Yolo

Fig. 4: Comparación del método Yolo+SeqNms con el método Yolo en el vídeo Archery.avi

VI. CONCLUSIONS

Lo primero que he reflexionado tras la realización de la primera sesión es la dificultad que tiene descargarse un código de GitHub y ponerlo en funcionamiento. No es tarea sencilla seguir las instrucciones que el creador del código surgiere, ya que en la mayoría de casos estas no son suficientes. Muchas veces las librerías que el tiene instaladas no son las mismas que las tuyas, puede que ciertas funciones ya no funcionen y además de muchas veces asumen conocimientos previos que puede que no tengas.

Tras la segunda sesión, queda claro que el método que mejor funciona de los dos comparados es el que implementamos en la sesión 1. Gracias al post-porcesdao Seq-Nms, las detecciones del método de la primera sesión son mucho más robustas y fiables. Por lo tanto, si tuviese que realizar una detección de objetos en vídeo, utilizaría el método Yolo + SeqNms, que aunque tiene un coste computacional un poco mayor, consigue mucho mejor resultado. El coste computacional adicional que tiene es debido al método de post-procesado. El tiempo adicional que tiene es prácticamente insignificante.

VII. TIME LOG

	Tiempo Tasks (h)	Tiempo Memoría (h)
Sesión 1	3	3
Sesión 2	1	3

TABLE I: Tiempo empleado en cada sesión

REFERENCES

[1] ZHANGHeng19931123. melodiepupu/seq_nms_yolo . GitHub. 2020. Available from: https://github.com/melodiepupu/seq_nms_yolo

[2] 2. Soomro K, Zamir A, Shah M. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild [Internet]. arXiv.org. 2012. Available from: https://arxiv.org/abs/1212.0402