Detección de caras usando una red YOLO preentrenada

Visión por Computador

Miguel Lentisco Ballesteros Antonio Coín Castro

Curso 2019-20

Índice

1. Introducción

Dataset WIDERFACE (Yang, Luo, Loy, & Tang, 2016).

2. Cosas por hacer

Varios:

- Entender todo el código. Eliminar lo que no sea necesario.
- Adaptar código para poder evaluar el conjunto de test (a partir de filelist, sin anotaciones).
- Ver por qué no coincide la métrica de evaluación de `evaluate_coco` con la de `codalab`. Reimplementar para que coincidan. Posiblemente cambiar el cálculo de AP a la interpolación en 101 pasos (https://kharshit.github.io/blog/2019/09/20/evaluation-metrics-for-object-detection-and-segmentation). Si no funciona, probar con 11 pasos.

Entrenamiento:

- Hacer finetuning a partir de los pesos de COCO iniciales (congelar unas cuantas capas, ¿cuáles?)
- Cambiar optimizador a SGD, RMSProp, Adabound(https://github.com/Luolc/AdaBound/blob/maste <—
- Entrenar más épocas. <—
- Entrenar con un valor mayor de *xywh_scale* en el config. Por ejemplo 2? <—
- Entrenar con un mayor tamaño de entrada de las imágenes. Ahora mismo en Colab no es viable.
- Aumentar el umbral *ignore_thresh*, por ejemplo a 0.6 ó 0.7.

Evaluación:

- Aumentar tamaño de entrada (no sé si tiene sentido que supere al input_size de entrenamiento) <—
- Aumentar umbral supresión de no máximos, por ejemplo a 0.6

Opcionales:

- Reimplementar la funcíón de supresión de no máximos en su versión vectorizada, para que sea más rápida. Adaptar implementación de https://www.pyimagesearch.com/2015/02/16/faster-non-maximum-suppression-python/

3. Información a tener en cuenta

- The output of the model is, in fact, encoded candidate bounding boxes from three different grid sizes: 13x13, 26x26 y 52x52.
- Explicación de mAP: `https://medium.com/@jonathan_hui/map-mean-average-precision-for-ob`

PASOS SEGUIDOS:

1. Convertir las anotaciones de WIDERFACE a formato VOC. Para ello se ha usado el archivo `convert.py`, adaptado de https://github.com/akofman/wider-face-pascal-voc-annotations/blob/master/convert.py
2. Generar anchor boxes para nuestro conjunto usando k-means con gen-anchors.py, y ponerlos en el config.
3. Descargar los pesos backend.h5 preentrenados en COCO.
4. Comenzar el entrenamiento en nuestro conjunto.
5. Validar usando el servidor de Codalab (https://competitions.codalab.org/competitions/2014).

*** Entrenamiento tras 100 épocas: ***

Parámetros: min-input: 288, max-input: 512

loss: 19.5426 - yolo_layer_1_loss: 0.8661 - yolo_layer_2_loss: 5.1030 - yolo_layer_3_loss: 13.5735 AP (Pascal VOC 2007): 0.4739 mAP (COCO 2017): 0.3

Apéndice: Funcionamiento del código

Bibliografía

Yang, S., Luo, P., Loy, C. C., & Tang, X. (2016). WIDER FACE: A Face Detection Benchmark. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.