Sistema de detección automática de socavones en el asfalto a partir de imágenes

Diego Castro Viadero

Septiembre 2019

Abstract

El estado del asfalto en carreteras tanto de ámbito nacional como de ámbito urbano es de alta importancia en relación a la seguridad vial. En la actualidad, no existe un sistema de detección automática de socavones en el asfalto. Tan sólo se tiene conocimineto de los mismos cuando han sido los causantes de un accidente vial o de una queja ciudadana (detección pasiva).

Este proyecto pretende desarrollar un sistema de detección automática y activa de socavones a partir de imágenes, que permita a las autoridades pertinentes conocer el número y ubicación de los mismos. Los principales objetivos son:

- Detección temprana y activa de socavones a partir de imágenes
- Creación de una base de datos con la relación de socavones detectados (número y ubicación)

Los principales beneficios son:

- Optimización de recursos necesarios para la reparación de socavones
- Aumentar la seguridad vial de las carreteras y evitar accidentes
- Aumentar la satisfacción de la ciudadanía en relación al estado de las carreteras de su municipio

!!! TODO

Abstract

!!! TODO

Contenido

1 Introducción

1.1 Motivación y Objetivos

!!! TODO

1.2 Estructura del trabajo

!!! TODO

2 Estado del arte

El problema que se pretende resolver podría ser afrontado de dos posibles maneras:

- Como un problema de clasificación de imágenes
- Como un problema de detección de objetos

El primero de los enfoques es más sencillo y está más estudiado. Dada una imagen, se determina una clase a la que pertenece la imagen. En los problemas de clasificación cada una de las imágenes se centran en un único objeto. Este tipo de problemas de clasificación se resuelven comúnmente con redes neuronales convolucionales. Existen numerosas arquitecturas de redes neuronales convolucionales ya definidas y estudiadas para resolver este tipo de problemas, como por ejemplo: VGG-16, LeNet, ResNet, GoogLeNet/Inception, etc.

El segundo de los enfoques es más complicado, y presenta varios retos. El primero de ellos es que las imágenes no se centran en un único objeto, sino que puede haber múltiples objetos a detectar y además tratarse de objetos de distintos tipos. El segundo de los restos es el tamaño de los objetos a identificar, que puede ser variable. Y el tercero de los retos es que se están resolviendo dos problemas al mismo tiempo: localizar objetos en una imagen y clasificar los objetos localizados.

Para resolver los problemas de detección de objetos existen dos aproximaciones. La primera de las aproximaciones es una aproximación clásica, basada en técnicas de machine learning. Un ejemplo representativo de esta aproximación clásica es Viola-Jones, que se basa en clasificadores binarios y que se ha usado en las cámaras de fotos para la detección de caras.

El uso del deep learning para la detección de objetos ha supuesto una revolución y ha cambiado las reglas del juego. Esta aproximación para la resolución de este tipo de problemas es relativamente reciente y ha estado en constante evolución.

!!! TODO

- R-CNN
- Fast R-CNN
- Faster R-CNN
- SDD
- YOLO (YOLO, YOLOv2, YOLOv3)
- Mask R-CNN

3 Definición de requisitos y análisis

3.1 Definición de requisitos

!!! TODO

3.2 Arquitectura

!!! TODO

3.3 Tecnologías

!!! TODO

4 Datos

4.1 Descripción de las fuentes de datos a utilizar

El juego de datos ha sido obtenido de kaggle [2] y se compone de un total de 1900 imágenes, tomadas desde el interior de un coche, con un tamaño igual a 3680x2760 píxeles (formato 4:3), y de un conjunto de ficheros de texto con el etiquetado de las mismas. Las imágenes se dividen en dos subconjuntos: uno de 1297 imágenes para el entrenamiento y otro de 603 imágenes para la evaluación del modelo. Por cada uno de los subconjuntos de imágenes existe un fichero de texto con el etiquetado de las mismas. Cada una de las líneas del los ficheros de texto contiene las etiquetas de una imagen. La estructura de cada línea es la siguiente:

```
<RUTA_IMG> <NUMERO_DE_ETIQUETAS>( <X0> <Y0> <ANCHO> <ALTO>)+
```

Para facilitar el posterior tratamiento, se ha realizado una transformación del formato de los ficheros de etiquetas al siguiente formato:

```
<RUTA_IMG>( <X0>,<Y0>,<ANCHO>,<ALTO>,<CLASE>)+
```

4.2 Estudio de los datos

En una fase inicial se ha realizado un análisis del tamaño de los socavones con respecto al tamaño de la imagen. Esto es un aspecto importante a tener en cuenta de cara a determinar el algoritmo a utilizar para la detección de objetos. Los algoritmos de detección de objetos, en general se comportan peor cuanto más pequeños son los objetos a detectar.

Como se observa en la figura 1, la mayoría de los socavones tienen una anchura inferior a 200 píxeles y una altura inferior a 50 píxeles. Este factor será tenido en cuenta en el preprocesamiento de las imágenes.

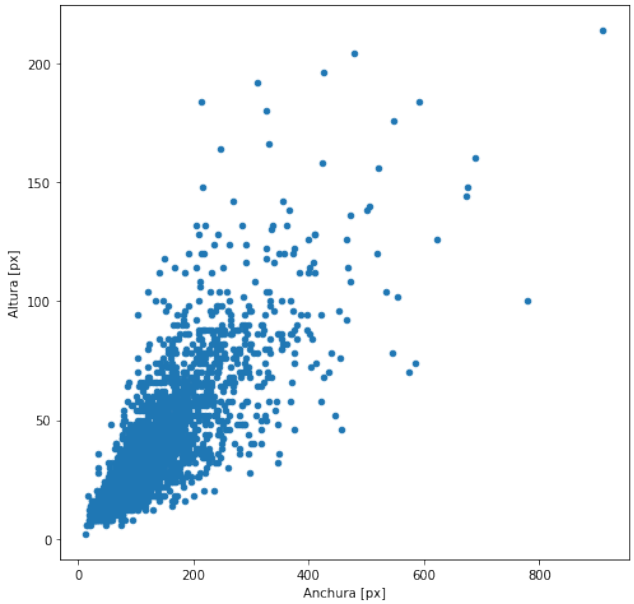


Figura 1: Tamaños de los socavones en píxeles

También se ha realizado un estudio de la localización de los socavones en las imágenes. Tal y como se ve en la figura 2, los baches están localizados principalmente en el centro de la imagen. La parte inferior se corresponde con el salpicadero del coche y la parte superior se corresponde con paisaje.

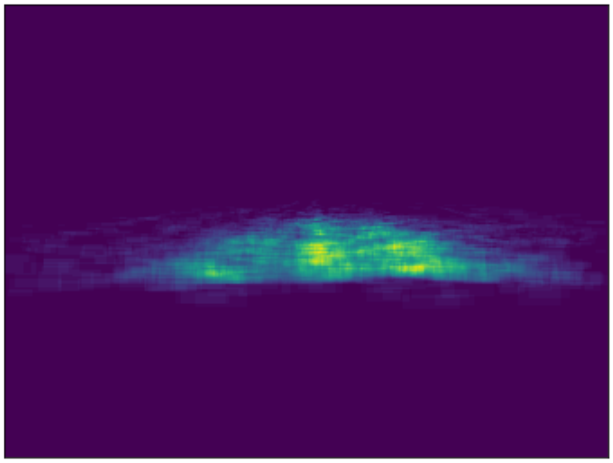


Figura 2: Localizaciones de los socavones en las imágenes

4.3 Preprocesamiento de las imágenes

Tanto en la fase de entrenamiento, como para hacer una predicción, las imágenes van a ser redimensionadas al tamaño de la red neuronal, la cual tiene una relación de aspecto 1:1. Para redimensionar una imagen con una relación de aspecto 4:3, y al mismo tiempo, transformarla en una imagen con relación de aspecto 1:1, lo que se hace es redimensionar el lado más grande de la imagen manteniendo la relación de aspecto, es decir, aplicando el mismo factor de redimensionamiento al lado más pequeño. Una vez redimensionada, se rellena con gris la zona superior y la zona inferior de la imagen para cuadrarla. En la figura 3 se muestra un ejemplo gráfico.

Figura 3: A la izquierda la imagen original redimensionada a tamaño 920x690 px (manteniendo la relación de aspecto 4:3). A la derecha la imagen redimensionada con el relleno para que tenga una relación de aspecto 1:1 (920x920 px)

El redimensionamiento se hace en base al lado más grande de la imagen, que en el ejemplo anterior es la anchura. Para determinar el factor de redimensionamiento, se divide la anchura la imagen final entre la anchura de la imagen original, en este caso: $920/3680 = 0.25$. A continuación, se aplica este factor de redimensionamiento a ambos lados de la imagen, resultando en un tamaño de 920x690 píxeles. Por último se calcula el relleno que haría falta a cada lado de la imagen: $(920 - 690)/2 = 115$.

Siguiendo con este ejemplo, si en la imagen original hubiese un socavón de tamaño 160x24 píxeles, y se aplicase este factor de redimensionamiento, el socavón redimensionado tendría unas dimensiones de 40x6 píxeles, lo cual sería un tamaño bastante pequeño ya que únicamente tiene 6 píxeles de alto (de 920 que tiene la imagen).

Sin embargo, si previo al redimensionamiento de la imagen, se recortan los extremos izquierdo y derecho de la imagen, de tal forma que tenga una relación de aspecto 1:1, se consigue que el factor de redimensionamiento sea mayor y

que por tanto los socavones redimensionados sean también más grandes. Esta técnica tiene un inconveniente, y es que la imagen original se está recortando, por lo que está habiendo una pérdida de información. Este inconveniente no es un impedimento, ya que en el apartado 4.3 se ha comprobado que la mayor parte de los socavones están en el centro de las imágenes, y que recortando los extremos de las mismas la pérdida de información es mínima.

Para aplicar esta técnica, en primer lugar habría que calcular los recortes que hay que hacer a cada lado de la imagen original. Para ello se calcula la diferencia entre la anchura y la altura de la imagen y se divide por dos: $(3680 - 2760)/2 = 460$. Una vez recortada la imagen se calcula el factor de redimensionamiento: $920/2760 = 0.333$. Por último se aplicaría este factor de redimensionamiento a la altura y la anchura de la imagen.

Si aplicamos este nuevo factor de redimensionamiento al tamaño del socavón del ejemplo anterior (160x24 píxeles), el tamaño del socavón redimensionado sería 53x8 (un 75% más grande).

5 Técnicas de Deep Learning y métodos de evaluación

5.1 Explicar las técnicas de DL que se van a utilizar en el proyecto

!!! TODO

- mencionar transfer learning
- redes convolucionales

Para el entrenamiento se ha utilizado la técnica *hold-out*, que consiste en dividir el conjunto de datos en dos subconjuntos: uno que será utilizado para la fase de entrenamiento y el otro que será utilizado para evaluar el modelo entrenado. Dada la escasez de imágenes no se ha utilizado un conjunto de validación durante el entrenamiento.

5.2 Explicar los métodos de evaluación que se van a utilizar en el proyecto

En los problemas de detección de objetos en imágenes se tienen por un lado una serie de regiones definidas en cada una de las imágenes que se corresponden con los objetos que se tienen que detectar. Y por otro lado se tiene un listado de regiones con los las predicciones de objetos que se han realizado.

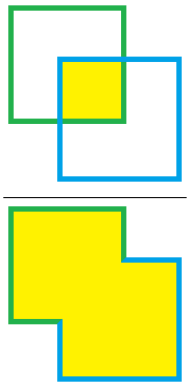
La métrica que se ha utilizado para evaluar el modelo obtenido es la AP (Average Precision), que es la métrica que se utiliza para evaluar modelos de detección de objetos.

Antes de explicar en qué consiste la métrica AP hay que explicar una serie de conceptos en los cuales está basada: IoU (intersección sobre la unión), precisión (precision) y sensibilidad (recall).

El concepto de IoU mide cuánto se solapan dos regiones: la predicha y la que debería ser detectada. Se calcula dividiendo la región obtenida mediante la intersección de la región predicha y la región a detectar entre la región obtenida

mediante la unión de ambas regiones.

$$AP = \frac{intersección}{unión} = \frac{\quad}{\quad}$$



La *precisión* (precision) mide la capacidad del modelo para detectar únicamente los objetos relevantes. Se calcula como el porcentaje de predicciones positivas acertadas frente a todas las predicciones positivas predichas:

$$precisión = \frac{TP}{TP + FP} = \frac{TP}{todas\ las\ predicciones\ positivas}$$

La *sensibilidad* (recall) mide la capacidad del modelo para detectar todos los objetos relevantes. Se calcula como el porcentaje de predicciones positivas acertadas frente a todas las existentes:

$$sensibilidad = \frac{TP}{TP + FN} = \frac{TP}{todas\ las\ regiones\ a\ detectar}$$

Tanto en el cálculo de la *precisión* como en el cálculo de la *sensibilidad*, para determinar si una predicción es positiva, se utiliza el *IoU*. Se define un umbral para el *IoU* (normalmente suele ser 0.5) y si se supera dicho umbral, la predicción es considerada una predicción positiva.

La métrica *AP* se calcula como el área debajo de la curva *precisión-sensibilidad* (precision-recall). En el eje de las abscisas se representa la *sensibilidad* (recall) y en el eje de las ordenadas se representa la *precisión* (precision).

A continuación se va a mostrar un ejemplo práctico de cómo se calcula la *AP*. Para este ejemplo se dispone de una serie de imágenes con un total de 4 socavones a detectar. En la tabla 1 se puede ver el cálculo de la *precisión* y de la *sensibilidad* para las predicciones obtenidas. La columna *Positivo* indica si la predicción es positiva, es decir, si el valor de *IoU* supera el umbral definido, que en este caso es 0.5. Las columnas *TP* y *FP* muestran el acumulado de sus respectivos valores.

IoU	Positivo	TP	FP	Precisión	Sensibilidad
0.912933	1	1	0	1.000000	0.25
0.711111	1	2	0	1.000000	0.50
0.387983	0	2	1	0.666667	0.50
0.387983	0	2	2	0.500000	0.50
0.387983	0	2	3	0.400000	0.50
1.000000	1	3	3	0.500000	0.75
0.225986	0	3	4	0.428571	0.75
0.225986	0	3	5	0.375000	0.75
1.000000	1	4	5	0.444444	1.00

Tabla 1: Cálculo de la precisión y sensibilidad para las predicciones

Una vez se tienen calculados los valores de *precisión* y de *sensibilidad* se calcula la curva *precisión-sensibilidad* como se puede ver en la figura 4. Para realizar el cálculo del área debajo de la curva se realiza un suavizado de la misma. Este suavizado consiste en establecer como valor de *precisión* para un determinado valor de *sensibilidad*, el valor de *precisión* más alto que se encuentre a su derecha. Por ejemplo, para la *sensibilidad* 0.6 se establece como valor de *precisión* el valor más alto a su derecha, que en este caso es 0.5. En color naranja se puede ver la curva suavizada.

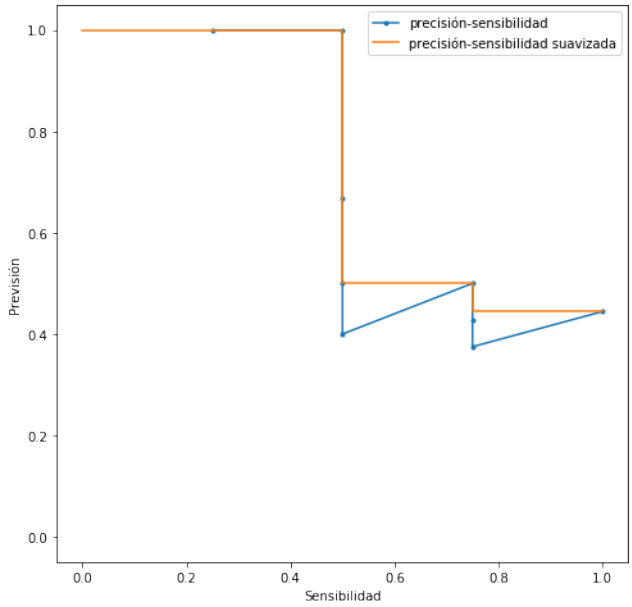


Figura 4: Curva precisión-sensibilidad

Por lo que finalmente, para este ejemplo, el cálculo del AP sería:

$$AP = (0.5-0)\cdot 1+(0.75-0.5)\cdot 0.5+(1-0.75)\cdot 0.44 = 0.5+0.125+0.11 = \mathbf{0.735}$$

6 Implementación y evaluación de las técnicas

6.1 Detalles de la implementación de las técnicas de DL aplicadas

!!! TODO

6.2 Evaluación de las técnicas

Se han entrenado dos versiones de YOLO: la versión 3 y la versión 3 tiny. Inicialmente se entrenó únicamente la versión 3, pero al ejecutarla en un dispositivo móvil se observó que el rendimiento lo hacía inutilizable, es este el motivo por el cual que se ha entrenado la versión tiny.

Para cada una de estas versiones se han entrenado varios modelos con distintos tamaños de red, por dos motivos principalmente: por una cuestión de rendimiento a la hora de ejecutar el modelo en un dispositivo móvil y por analizar cómo varía la precisión del modelo cambiando el tamaño de la red.

Además se han utilizado distintos conjuntos de entrenamiento para entrenar todas las variantes del modelo. El primero de los conjuntos de entrenamiento se corresponde con el conjunto íntegro original (denominado *completo*). Los resultados obtenidos con este conjunto de entrenamiento obtuvieron unos valores bajos para la métrica AP, y tras analizar los motivos, se observó que había una gran cantidad de socavones demasiado pequeños que podían ser los causantes malos resultados. Por este motivo, se han utilizado dos conjuntos de entrenamiento adicionales aplicando filtros sobre los socavones. En el primero de estos conjuntos de entrenamiento adicionales se han filtrado los socavones con tamaño superior a 75x30 píxeles (denominado *filtro 75x30*) y en el segundo se han filtrado los socavones con tamaño superior a 100x40 píxeles (denominado *filtro 100x40*). Para cada uno de estos conjuntos de entrenamiento adicionales se ha creado también su correspondiente conjunto de evaluación aplicando el mismo filtro.

Con todos los modelo resultantes obtenidos se ha realizado una doble evaluación. Por un lado se han evaluado con los conjuntos de test correspondientes para cada uno de los conjuntos de entrenamiento (resultados en la tabla 2). Y por otro lado se han evaluado con un conjunto de imágenes generado para este proyecto (resultados en la tabla 3). Este conjunto de evaluación (denominado *propio*) se compone de unas 30 imágenes de 4032x3024 píxeles, con unos 60 socavones en total, obtenido desde la acera (a diferencia del original que fue obtenido desde el coche) y compuesto por fotos realizadas en España (a diferencia del original que fueron realizadas en Sudáfrica).

Versión YOLO	Tamaño	Juego datos	Épocas	Mejor AP
V3	256	completo	43	0.0747
V3	256	filtro 100x40	93	0.3077
V3	256	filtro 75x30	88	0.2513
V3	416	completo	18	0.1467
V3	416	filtro 100x40	93	0.4161
V3	416	filtro 75x30	93	0.3611
V3	640	completo	13	0.0186
V3	640	filtro 100x40	63	0.5475
V3	640	filtro 75x30	53	0.4106
V3 Tiny	256	completo	144	0.0046
V3 Tiny	256	filtro 100x40	136	0.0510
V3 Tiny	256	filtro 75x30	153	0.0392
V3 Tiny	416	completo	153	0.0145
V3 Tiny	416	filtro 100x40	153	0.1307
V3 Tiny	416	filtro 75x30	146	0.0869

Tabla 2: Resultados obtenidos con los conjuntos de evaluación originales

Versión YOLO	Tamaño	Juego datos	Épocas	Mejor AP
V3	256	completo	43	0.0289
V3	256	filtro 100x40	93	0.1018
V3	256	filtro 75x30	88	0.0179
V3	416	completo	18	0.0354
V3	416	filtro 100x40	93	0.0089
V3	416	filtro 75x30	93	0.0294
V3	640	completo	13	0.0017
V3	640	filtro 100x40	63	0.0342
V3	640	filtro 75x30	53	0.0961
V3 Tiny	256	completo	144	0.0086
V3 Tiny	256	filtro 100x40	136	0.0232
V3 Tiny	256	filtro 75x30	153	0.0371
V3 Tiny	416	completo	153	0.0000
V3 Tiny	416	filtro 100x40	153	0.0000
V3 Tiny	416	filtro 75x30	146	0.0006

Tabla 3: Resultados obtenidos con el conjunto de evaluación propio

7 Resultados

7.1 Resultados del proyecto

!!! TODO

8 Conclusiones

8.1 Evaluación del proyecto

!!! TODO

8.2 Alternativas y posibles mejoras que podrían haberse aplicado al proyecto (trabajos futuros)

!!! TODO

8.3 Conclusiones personales

!!! TODO

Referencias

[1] Jonathan Hui. *mAP (mean Average Precision) for Object Detection*. URL: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173. (accedido: 29/08/2019).

[2] Felipe Muller. *Nienaber Potholes 2 Complex*. URL: https://www.kaggle.com/felipemuller5/nienaber-potholes-2-complex. (accedido: 26/08/2019).

[3] Javier Rey. *Object Detection with Deep Learning: The Definitive Guide*. URL: https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning. (accedido: 29/08/2019).