# Detección automática de personas en imágenes

El objetivo de este trabajo es implementar (cada grupo decidirá hasta dónde llega) un detector automático de personas en imágenes. Para llevarlo a cabo, vamos a separar el proyecto en dos fases:

- Fase de extracción de características y aprendizaje
- Fas de detección

En la primera fase, vamos a seguir el mismo procedimiento que emplearíamos en cualquier otro problema de clasificación supervisada. Partiremos de un conjunto de imágenes de entrenamiento que contienen imágenes de la clase positiva (personas) con una resolución de 134x70 píxeles.

Tendréis también a vuestra disposición un conjunto de imágenes (200) de imágenes de mayor resolución que no contienen personas. El objetivo es que extraigáis de dichas imágenes subimágenes de dimensión 134x70 con la cual poder extraer imágenes de la clase negativa.

El objetivo de esta fase es entrenar un modelo de clasificación supervisada (regresión logística, red neuronal, SVM, etc.) que aprenda las características propias de las personas y permita distinguir una imagen que contenga una persona de una que no la contenga. El problema viene en determinar con qué características vamos a alimentar el aprendizaje. Nuestra primera idea podría ser la utilización directa de los píxeles. Sin embargo, enseguida nos damos cuenta de que no tiene mucho sentido: el número de píxeles es demasiado grande, la variabilidad de intensidades es muy grande, la localización de una persona dentro de la imagen no tiene por qué ser siempre la misma, y un largo etcétera de problemas. Para solucionarlos, debemos transformar las imágenes en un vector de características que nos permitan discriminar de manera mucho más precisa y con una mayor capacidad de generalización la presencia o no de una persona en la imagen.

Objetivo 1: investiga diferentes métodos de extracción de características en imágenes. En concreto, fíjate en HOG, que convierte una imagen en un conjunto de mini-histograma de gradientes que sirven para “definir” la forma de los objetos que aparecen en la imagen. La librería scikit-image tiene un método que permite transformar una imagen en un HOG. Investiga sus parámetros, etc...

Una vez elegido el método de extracción de características, debemos transformar nuestro imageset en un dataset que podamos utilizar en scikit-learn.

Objetivo 2: ¿cuántas imágenes necesitamos para entrenar el modelo? ¿Debemos hacer algún preprocesamiento (ecualización del histograma, etc.) antes de extraer las características? ¿Podemos implementar algún modelo sencillo para aumentar el número de imágenes de train sin tener que recurrir a buscar nuevas imágenes? ¿Qué división hago en train/test? ¿Cómo valido el modelo? ¿Qué modelo es mejor? ¿Hay diferencias significativas entre modelos? ¿Y entre características? Aquí podéis poner en práctica todo lo aprendido en Aprendizaje Formal...

Con el modelo ya entrenado, es hora de pasar a la segunda fase, la fase de detección. Como habrás podido comprobar, las imágenes de entrenamiento corresponden a imágenes relativamente pequeñas de personas. Evidentemente, cuando queremos aplicar esto sobre imágenes reales, la imagen no solo contiene una persona, sino que tiene muchas más cosas (paisaje, casas, coches, etc.). Por tanto, debemos hacer una búsqueda en forma de algoritmo de ventana deslizante. Un algoritmo de ventana deslizante es muy parecido en funcionamiento a una convolución. Básicamente, vamos a coger una ventana del mismo tamaño que las ventanas con las que hemos entrenado el modelo y vamos a colocarla en la primera posición de la imagen en la que quepa. Extraeremos la correspondiente subventana de la imagen, la mandaremos a evaluar al modelo de clasificación y me dirá si hay o no hay una persona en la imagen. Una vez testeada, moveré la ventana de detección a la derecha y volveré a extraer la subimagen y a preguntar al modelo si hay persona o no. Una vez que consiga pasar por todas las ventanas de la imagen tendré posibles localizaciones de personas en aquellos puntos en los que el modelo me haya predicho que existe persona.

Objetivo 4: implementa un algoritmo de ventana deslizante. Evalúa los siguientes problemas, proponiendo (si lo ves necesario) posibles soluciones:

- ¿Cómo son los tiempos de ejecución de dicho algoritmo? ¿Podemos hacer algo para reducirlo? Por ejemplo, no tener un desplazamiento de un píxel, sino mover la ventana de detección de más en más píxeles...
- ¿Existen múltiples positivos para una misma persona? Podría ocurrir que en una localización específica el modelo prediga persona y que, al mover la ventana de detección a la derecha, también prediga persona aun cuando estamos en la misma persona. Y puede que esto haya ocurrido también arriba/abajo. ¿Podemos hacer una única localización para esa persona? Quizá se podría coger la probabilidad del modelo y quedarnos con la máxima probabilidad en una misma zona...
- El entrenamiento del modelo se ha realizado con imágenes de 134x70. Sin embargo, no todas las personas en una imagen tienen por qué tener dicho tamaño. ¿Podríamos hacer algo en el algoritmo para detectar personas de menor/mayor tamaño? Por ejemplo, podría pensarse en modificar el tamaño de la ventana de detección o bien en fijarlo, pero entonces ampliar/reducir las imágenes para buscar personas que ocupen más/menos.

Entrega: debéis entregar el código realizado y un documento (informe) donde expliquéis qué habéis hecho, qué decisiones habéis tomado, qué resultados habéis tenido, qué mejoras habéis propuesto y todo aquello que consideréis importante reseñar. El informe debe ser explicativo pero conciso.