Master en Ciencia de Datos e Ingeniería de Computadores

Extracción de Características en Imágenes

Extracción de Rasgos

Pedro Jesús López Abenza
48454588-M

Universidad de Granada
Curso 2018/2019

Índice

1. Introducción

Este trabajo pretende ejercer como una introducción a la **extracción de rasgos en imágenes** y la aplicación de métodos típicamente aplicados para la obtención de dichos rasgos. Así pues, se han utilizado métodos basados en los descriptores presentados en la asignatura como es el caso de los descriptores **HOG**, **LBP**, **U-LBP** y la combinación **HOG+LBP**, los cuales se tratarán de implementar, evaluar y aplicar a lo largo del trabajo.

El objetivo es crear un **algoritmo capaz de detectar qué tenemos en la imagen mediante el aprendizaje de un conjunto de imágenes** en base al estudio de los descriptores anteriormente citados. Para ello, se ha suministrado una base de datos de imágenes dentro de las cuales contamos con un conjunto de entrenamiento de un total de 4306 imágenes, que permitirá realizar el aprendizaje a nuestro clasificador, y otro de test con 1100 imágenes, que servirá para comprobar la calidad de lo aprendido. Dentro de esta base de datos contamos con dos tipos de imágenes: En una de las carpetas, *pedestrians*, encontramos imágenes en las cuales aparecen personas en diversas situaciones; mientras que en la otra carpeta, *background*, aparecen imágenes con diversos tipos de fondos.

La idea es que **un clasificador SVM realice el aprendizaje de dichas imágenes mediante los distintos descriptores estudiados en la asignatura**, de manera que si le damos otra imagen distinta a las estudiadas sea capaz de predecir correctamente si lo que aparece en ella es una persona o un fondo en base a las características de la misma y si estas se asocian a las analizadas en el entrenamiento. Para ello, trataremos de implementar esto mediante los métodos HOG, LBP, U-LBP y la combinación HOG+LBP, de modo que analizaremos cuál de ellos es el que mejores resultados ofrece, realizando un barrido de **los distintos parámetros que nos permiten describir el clasificador SVM** (Núcleo, grado, parámetro γ, etc.) y se evaluará cuál es la combinación que mejores resultados ofrecen en la evaluación de la calidad del clasificador: En este caso, **se han elegido la precisión, el AUC y el *F-score***.

Además, yendo un paso más allá, es posible aplicar esta idea para realizar un **barrido de una imagen de tamaño cualquiera** a través de ventanas de tamaño igual a las imágenes de entrenamiento para así **detectar en ellas la presencia del objeto estudiado** (en este caso personas) en dichas ventanas. Así pues, se producirá un conjunto de ventanas etiquetadas como positivas, de modo que trataremos de realizar un agrupamiento eficiente para aquellas que representen la misma persona. Asimismo, se analizará cuál de los descriptores es el que mejores resultados ofrece a la hora de clasificar una cierta imagen.

Este trabajo ha sido realizado en **Python**, utilizando **PyCharm** como entorno IDE.

2. Evaluación del descriptor HOG

2.1. Descripción del método HOG

La primera de las tareas a realizar es la evaluación de la bondad del método HOG implantado mediante diversas medidas estudiadas a lo largo del Master, como precisión, *recall*, F1-*score*, AUC... Así como, en base a ellas, buscar una optimización de los parámetros del clasificador utilizado (que, como sabemos, en este caso es el SVM).

Como sabemos, el método HOG es un método basado en gradientes sobre el espacio de color tridimensional, según el cual se obtienen los rasgos de la imagen mediante la extracción de histogramas en base a la dirección del gradiente. Así pues, en este método se definirán unos intervalos que contengan las posibles direcciones del gradiente, que tendrán su espacio en el histograma.

De tal modo, cada gradiente se define según sus dos intervalos más cercanos y contribuirá en el histograma con un peso proporcional a su distancia al centro de cada intervalo. Por tanto, se estudiarán bloques de píxeles de un cierto tamaño predefinido y se obtendrán las características de dicho bloque en base a la concatenación de histogramas con valores entre 0 y el número de particiones que hayamos realizado en el espacio de direcciones.

2.2. Implementación del método HOG

De acuerdo a estas ideas, la estructura de nuestro programa (llamado *HOG_Evaluation.py*) es la siguiente:

1. Estudio de los datasets de *train* y *test* mediante el método HOG: Cada imagen, con clase conocida, es descrita por las características obtenidas en base al descriptor HOG.

2. Uso del clasificador basado en SVM. Aplicación de un tuneado para buscar los parámetros más óptimos.

3. Realización de la predicción del conjunto de *test* mediante el clasificador definido utilizando dichos parámetros óptimos.

En este *Script* se carga el descriptor HOG de la librería *OpenCV*, contenido en la función *cv2.HOGDescriptor()*. Tras ello, se define la función *hog_images()*, la cual recibe la dirección de una carpeta (para así elegir entre *train* ó *test*) y aplica el descriptor HOG sobre cada una de las imágenes de la carpeta, formando una matriz que contendrá dichas características y un vector que contendrá la correspondiente clase, los cuales serán la salida de la misma.

Se procede a aplicar dicha función sobre los conjuntos de *train* ó *test*, obteniéndose las características de cada imagen y su correspondiente clase. Tras ello, se aplica un *GridSearchCV* para buscar los mejores parámetros en base a la aplicación de la técnica de **validación cruzada con 5 *folds*** con las combinaciones de un conjunto de valores para dichos parámetros. Este algoritmo nos indica la mejor combinación de parámetros en base a la calidad de la bondad del clasificador(en este caso se ha elegido como medida de calidad *accuracy*). El algoritmo utiliza estos parámetros para realizar posteriormente la predicción de los datos de test.

2.3. Evaluación de la bondad del método HOG

Este *Script* nos dará una serie de resultados para las medidas de bondad seleccionadas en base de las distintas combinaciones de parámetros. Estos datos quedan recogidos en el primer apartado del ***Apéndice I***. En dicha tabla podemos ver como la mejor combinación de parámetros para el clasificador SVM utilizando el descriptor HOG es:

Descriptor HOG						
C	**Degree**	γ	**Kernel**	**Accuracy**	**AUC**	**F1-Score**
10	-	0.01	rbf	0.9811890385508593	0.9801560114685229	0.978680037042342

Como vemos, la mejor combinación es un **Kernel radial con parámetros** $C = 10$ **y** $\gamma = 0.01$, la cual nos da una **precisión** de 0.9811890385508593, lo cual es bastante óptimo. Por su parte, la calidad de bondad mediante las medidas de **AUC** y ***F1-Score*** son 0.9801560114685229 y 0.978680037042342, respectivamente, los cuales son también valores notablemente positivos.

Se aprecia que en algunos de los casos hay combinaciones con valores nulos de *F-Score*. Ello se debe a que en esos casos el producto resultante de la precisión y el *recall* da problemas y hace el *F-Score* incalculable (Lo cual nos lo indica el compilador durante la ejecución).

A continuación, veamos cómo funciona este descriptor a la hora de clasificar imágenes más allá de las de entrenamiento. Estudiaremos las características de las imágenes del conjunto de *test* y trataremos de predecir su clase (es decir, si es persona o fondo). Obtenemos los siguientes resultados, resumidos en la siguiente matriz de confusión, donde vemos un clasificador relativamente correcto:

		Predicted	
		Positive	Negative
Actual	Positive	597	3
	Negative	17	483

3. Implementación y evaluación del descriptor LBP

3.1. Descripción del método LBP

Otra de las tareas básicas a realizar en este proyecto es la implantación del método LBP. Como sabemos, en este método se realiza una caracterización de la imagen mediante el estudio de cada uno de los píxeles en base a comparaciones con sus ocho vecinos. De tal modo, si el valor del píxel central es mayor que el del vecino se toma un 1 y en caso contrario un 0. Esto crea una secuencia de ceros y unos con un orden predefinido de modo que nos permite caracterizar dicho píxel con un valor entre 0 y 255 como resultado del paso a decimal de la secuencia binaria anterior.

De tal modo, se definirán bloques de píxeles de un cierto tamaño y se estudiarán en base a dichas reglas, de modo que se obtendrán las características para dicho bloque en base a la concatenación de histogramas con valores entre 0 y 255, por lo que en este caso se tendrá un total de 256 etiquetas.

3.2. Implementación del método LBP

De acuerdo a estas ideas, la estructura de nuestro programa (llamado *LBP_Basico.py*) es la siguiente:

1. Definición de la funciones auxiliares para la correcta implementación para el estudio de una imagen de 128×64 mediante el descriptor LBP en Python.

2. Estudio de los datasets de *train* y *test* mediante el método LBP: Cada imagen, cuya clase (persona o fondo) nos es conocida, es descrita por unas características obtenidas en base al descriptor LBP.

3. Uso del clasificador basado en SVM. Aplicación de un tuneado para buscar los parámetros más óptimos.

4. Realización de la predicción del conjunto de *test* mediante el clasificador definido utilizando dichos parámetros óptimos.

En este *Script* se realiza la implementación del descriptor LBP. Para ello, se comienza definiendo una serie de funciones que será aplicadas recursivamente una en otra: La función *comp-pix()* es la encarga de la comparación entre dos pixels vecinos, lo cual es aplicado en la función *LBP_pixel()*, la cual realiza dicha comparación para un cierto pixel y cada uno

de sus 8 vecinos (ya que, por sencillez, se ha definido para todo el proyecto un radio unidad en la elección de vecinos), devolviendo una lista con los valores binarios correspondientes.

La función *LBP_basic()* aplica este algoritmo en una cierta *sub*-imagen, realizando la transformación a decimal del código binario y calcula el histograma correspondiente a dicho bloque. Por su parte, la función *LBPbas_compute()* aplicará toda esta idea a lo largo de la imagen, de modo que obtendrá una concatenación de histogramas, que serán las características LBP de la imagen. Cabe destacar que para tratar el caso de los píxeles exteriores del bloque, se han utilizado bloques de 18×18, realizándose el cálculo LBP para los píxeles situados en la región central (formando así un bloque de 16×16). De este modo, dichos píxeles originalmente exteriores ahora sí tendrán todos sus vecinos disponibles. Además, para poder aplicar esta idea en el caso de los píxeles exteriores de la ventana, se ha definido un *padding*, de modo que se añade un contorno exterior de píxeles con valor 1, de modo que así ya es posible realizar la comparación adecuadamente.

Tras ello, se define la función *lbp_images()* que, de forma igual a la función homónima anterior, recibe parte de la dirección de una carpeta (para así elegir entre *train* ó *test*) y aplica el descriptor LBP sobre cada una de las imágenes de la misma, formando una matriz de características y un vector de clases, los cuales serán su salida. De nuevo, se aplica dicha función sobre los conjuntos de *train* ó *test* y, obtenidas las características y clases, se aplica un *GridSearchCV* que buscará los mejores parámetros mediante la aplicación de una **validación cruzada con 5 *folds*** con las combinaciones de un conjunto de valores para dichos parámetros, indicándonos la mejor combinación en base a la calidad de la bondad del clasificador (que también será medida con *accuracy*). El algoritmo utiliza estos parámetros para realizar la predicción de los datos de test.

3.3. Evaluación de la bondad del método LBP

Tendremos así una serie de resultados para las medidas de bondad seleccionadas en base de las distintas combinaciones de parámetros. Estos datos quedan recogidos en el segundo apartado del ***Apéndice I***. En dicha tabla podemos ver como la mejor combinación de parámetros para el clasificador SVM utilizando el descriptor LBP es:

Descriptor LBP						
C	**Degree**	γ	**Kernel**	**Accuracy**	**AUC**	**F1-Score**
10	-	0.01	rbf	0.9934974454249884	0.9937804890624847	0.9927206945058119

Como vemos, la mejor combinación de parámetros es un **Kernel radial con parámetros** $C = 10$ **y** $\gamma = 0.01$. Esta nos da una **precisión** de 0.9934974454249884, lo cual es muy bueno, incluso mejor que lo obtenido con el descriptor HOG. Por su parte, la calidad de bondad

mediante las medidas de **AUC** y ***F1-Score*** son 0.9937804890624847 y 0.9927206945058119, respectivamente, los cuales son también mejores a los del otro descriptor.

No obstante, si realizamos una comparación con los resultados obtenidos por las otras combinaciones de parámetros se aprecia como también hay otras que obtienen calidades en la bondad muy cercanas a esta, por lo que el descriptor LBP es bastante bueno en general. No obstante, de nuevo se aprecia que en algunos de los casos hay combinaciones con valores nulos de *F-Score*, pues vuelve a existir el citado problema con el producto de la precisión y el *recall* que hace el *F-Score* incalculable.

Para evaluar la capacidad del descriptor LBP creado a la hora de clasificar imágenes más allá de las utilizadas en el entrenamiento, usaremos de nuevo las imágenes del conjunto de *test* para predecir si se trata de una persona o un fondo. Los resultados obtenidos se resumen en la siguiente matriz de confusión, donde vemos un clasificador mejor que el anterior:

		Predicted	
		Positive	Negative
Actual	Positive	598	2
	Negative	6	494

Si comparamos esta matriz de confusión con la obtenida para el descriptor HOG, se observa que, aunque ambos son muy buenos a la hora de clasificar personas como tal, el descriptor LBP ofrece un mejor rendimiento a la hora de clasificar como no-persona a los fondos, de modo que el número de falsos positivos se reducen de forma importante.

4. Implementación y evaluación del método U-LBP

4.1. Descripción del método U-LBP

A continuación, se pretende implementar el método U-LBP. Como sabemos, este es una variación del método LBP en la cual, dada la secuencia obtenida al comparar un píxel con sus vecinos, se considera que hay dos tipos de combinaciones. Para ello, se ha de contar el número de transiciones 0-1 existentes en la secuencia, recordando que existe un carácter cíclico en nuestros códigos. Se distinguen, por tanto, como códigos uniformes a aquellos con 0 o 2 transiciones, mientras que aquellos que presentan más son no-uniformes. De tal modo, el algoritmo considera que los códigos uniformes tendrán una etiqueta propia y característica del mismo, mientras que aquellos códigos que son no-uniformes se consideran un todo y se

engloban en una única etiqueta (que será la última).

Así, para el caso de 8 bits (que es el presente en este caso al considerar un radio de primeros vecinos), habrán 58 códigos no uniformes que, unidos a la etiqueta de los no-uniformes, deja un total de 59 etiquetas posibles para los píxeles. Ello supone una gran simplificación en comparación con el método LBP visto anteriormente, el cual tenía 256 valores posibles. Por tanto, en el problema U-LBP, se ha de obtener el código correspondiente al píxel objeto de estudio y se le asignará la etiqueta asociada a dicho código. De nuevo, se definirán bloques de píxeles y se estudiarán en base a dichas reglas, obteniéndose las características para ese bloque en base a la concatenación de una serie de histogramas obtenidos, los cuales en este caso contarán con valores entre 0 y 58.

4.2. Implementación del método U-LBP

De acuerdo a estas ideas, la estructura de nuestro programa (llamado *LBP_Unif.py*) es la siguiente:

1. Definición de la funciones auxiliares para la correcta implementación para el estudio de una imagen de 128×64 mediante el descriptor U-LBP en Python.

2. Definición del diccionario de códigos para el número de bits pertinentes (que en este caso es 8, pues nuestro problema se ha fijado con un radio a primeros vecinos) y su correspondiente etiqueta como uniforme/no-uniforme, en función del número de transiciones en el código.

3. Estudio de los datasets de *train* y *test* mediante el método U-LBP: Cada imagen, cuya clase (persona o fondo) nos es conocida, es descrita por unas características obtenidas en base al descriptor U-LBP.

4. Uso del clasificador basado en SVM. Aplicación de un tuneado para buscar los parámetros más óptimos.

5. Realización de la predicción del conjunto de *test* mediante el clasificador definido utilizando dichos parámetros óptimos.

En dicho *Script* se realiza la correspondiente implementación del descriptor U-LBP también de forma manual. Se utilizan nuevamente las funciones anteriores de *comp-pix()* y *LBP_pixel()* para definir el código binario asociado a un pixel en base a comparaciones con sus vecinos. Sin embargo, en este caso se utilizan las funciones *jump()* y *cyclic()*, que combinadas nos permiten calcular las transiciones 0-1 en el código asociado a un pixel y etiquetarlo

en función de si es uniforme o no.

La función *LBP_unif()* aplica el algoritmo U-LBP en una cierta *sub*-imagen, valorando si dicho código binario es uniforme o no. Para ello, esta función recoge un diccionario con los distintos códigos posibles para 8-bits y su correspondiente etiqueta, el cual ha sido definido utilizando las funciones *bitsDicc()* para generar el diccionario y *convert()* para transformar las listas de bits obtenidas por la función anterior en *strings*. De tal modo, la función devuelve el histograma para el correspondiente bloque.

Por su parte, la función *LBPunif_compute()* aplicará todo esto en la imagen, de modo que obtendrá una concatenación de histogramas que contendrán las características U-LBP para una cierta imagen. Destacar que de nuevo en este caso se ha seguido la misma idea para afrontar el problema de los píxeles exteriores del bloque y ventana: Se han utilizado bloques de 18×18, realizándose el cálculo U-LBP para los píxeles situados en la región central (formando así un bloque de 16×16) y se ha realizado un *padding* que añade un contorno exterior de píxeles con valor 1.

Tras ello, se define la función *ulbp_images()* que, al igual que las funciones homónimas anteriores, recibe parte de la dirección de una carpeta (para así elegir entre *train* ó *test*) y aplica el descriptor U-LBP sobre cada una de las imágenes de la misma, formando una matriz de características y un vector de clases, los cuales serán su salida.

De nuevo, se aplica dicha función sobre los conjuntos de *train* ó *test* y, obtenidas las características y clases, se aplica un *GridSearchCV* que realiza una búsqueda de los mejores parámetros en base a la aplicación de una **validación cruzada con 5 *folds*** con las combinaciones de un conjunto de valores para dichos parámetros, buscando la mejor en base a la calidad de la bondad del clasificador (que también será medida con *accuracy*). El algoritmo utiliza estos parámetros para realizar la predicción de los datos de test.

4.3. Evaluación de la bondad del método U-LBP

Tendremos así una serie de resultados para las medidas de bondad seleccionadas en base de las distintas combinaciones de parámetros. Estos datos quedan recogidos en el tercer apartado del ***Apéndice I***. En dicha tabla podemos ver como la mejor combinación de parámetros para el clasificador SVM utilizando el descriptor U-LBP es:

Descriptor U-LBP						
C	**Degree**	γ	**Kernel**	**Accuracy**	**AUC**	**F1-Score**
0.1	3	0.1	poly	0.9928007431490943	0.9928424973364298	0.9919190030936181

Por tanto, de nuevo la mejor combinación de parámetros es un **Kernel polinómico de grado 3 con parámetros** $C = 0.1$ **y** $\gamma = 0.1$. Dicha combinación de parámetros nos da una **precisión** de 0.9928007431490943 y unos valores de 0.9928424973364298 y 0.9919190030936181 para las medidas de **AUC** y ***F1-Score***,respectivamente. Por tanto, en este caso baja la calidad de las distintas medidas de la bondad de la clasificación, pero también son muy positivas dado que el muy reducido coste computacional del cálculo se ven notablemente reducido con este método en comparación al LBP.

Nuevamente se puede apreciar comparando con los resultados obtenidos por otras combinaciones de parámetros que hay otras con bondades muy similares. Deducimos de nuevo que hay parámetros que no son muy decisivos en las características del clasificador frente a otros que tienen un gran peso en su calidad. Por último, de nuevo se clasificarán las imágenes del conjunto de *test* mediante el modelo creado con el descriptor U-LBP, con el objetivo de predecir si se trata de una persona o un fondo. Obtenemos los siguientes resultados, resumidos en la siguiente matriz de confusión, donde vemos un clasificador mejor que el anterior:

		Predicted	
		Positive	Negative
Actual	Positive	596	4
	Negative	9	491

5. Combinación de los métodos HOG y LBP

Por otro lado, otra de las técnicas posibles a aplicar es combinar las características extraídas por los descriptores HOG y LBP para una cierta imagen. De tal modo, nuestro clasificador contará con más información y de diversa índole, de modo que, a priori, deberían de mejorar sus predicciones. Esto queda realizado mediante el *Script* denominado *HOG_LBP.py*, de forma que en este se aplicará el cálculo de las características HOG y LBP y se concatenarán, de modo que se contará con un total de 30660 características. La función *hoglbp_images()* se encargará de ir tomando cada una de las fotos de nuestra base de datos y obtener las características de los descriptores HOG y LBP para cada una de ellas y posteriormente concatenar estas características.

Tras ello, nuevamente se ha aplicado un *GridSearchCV* que realiza una búsqueda de los mejores parámetros en base a la aplicación de una **validación cruzada con 5 *folds*** para las combinaciones de un conjunto de valores para dichos parámetros y buscando la mejor

combinación de parámetros en base a la calidad de la bondad del clasificador (que también será medida con *accuracy*). El algoritmo utiliza estos parámetros para realizar la predicción de los datos de test.

5.1. Evaluación de la bondad del método HOG+LBP

Así pues, el objetivo de esta sección es combinar ambos métodos y comprobar si efectivamente hay una mejora en el predictor, evaluando la bondad de sus predicciones en base a las medidas de bondad seleccionadas. Así pues, se obtienen los resultados para dichas medidas dados las distintas combinaciones de parámetros del clasificador, los cuales quedan recogidos en el cuarto apartado del ***Apéndice I***. En dicha tabla podemos ver como la mejor combinación de parámetros para el clasificador SVM utilizando la combinación de los descriptores HOG+LBP es:

Descriptor HOG+LBP						
C	**Degree**	γ	**Kernel**	**Accuracy**	**AUC**	**F1-Score**
0.1	3	0.1	poly	0.995355318160706	0.9955577425438915	0.9947984186329326

Como vemos, la mejor combinación de parámetros es un **Kernel polinómico de grado 3 con parámetros** $C = 0.1$ **y** $\gamma = 0.1$. Esta nos da una **precisión** de 0.995355318160706, lo cual es un valor mejor que lo obtenido para los descriptores por separado. Por su parte, la calidad de bondad mediante las medidas de **AUC** y ***F1-Score*** son 0.9955577425438915 y 0.9947984186329326, respectivamente, de modo que también son superiores a los de los otros descriptores.

No obstante, de nuevo se aprecian como otras de las combinaciones de parámetros de parámetros tienen calidades en la bondad muy cercanas a esta, tal y como ocurría en algunos casos de los descriptores HOG y LBP por separado. Además, de nuevo se aprecia que en algunos de los casos hay combinaciones con valores nulos de *F-Score*, pues vuelve a existir el citado problema con el producto de la precisión y el *recall* que hace el *F-Score* incalculable.

Para evaluar la capacidad de predicción del descriptor HOG+LBP creado a la hora de clasificar imágenes más allá de las utilizadas en el entrenamiento, de nuevo se utilizarán las imágenes del conjunto de *test* para predecir si se trata de una persona o un fondo. Los resultados obtenidos se resumen en la siguiente matriz de confusión, donde vemos un clasificador mejor que los anteriores:

		Predicted	
		Positive	Negative
Actual	Positive	600	0
	Negative	5	496

Vemos como la combinación de los descriptores HOG+LBP es mejor que da unos resultados en la predicción incluso mejores que los del LBP por separado (y notablemente mejores que los del HOG), fallando únicamente al clasificar 5 imágenes de las 1100 que formaban el conjunto de *test*. Cabe destacar que, dado el conjunto de *test* analizado, no hay fallo en la detección de personas mediante la utilización de la técnica HOG+LBP: **Todas las imágenes en las que aparecían personas fueron clasificadas correctamente**. Por su parte, la detección de falsos positivos (clasificación como persona de los fondos) mejora en una única clasificación a lo obtenido en el LBP.

6. Búsqueda de personas a diferentes escalas

Por último, como desafío se propone hacer uso de nuestro clasificador entrenado mediante los diferentes descriptores para buscar la presencia de personas en una imagen cualquiera, lo cual se realizará a través del archivo *Escalado.py*. De tal modo, se definirán ventanas obtendrán las características relativas al descriptor elegido y tras ello se le aplicará el clasificador entrenado y con parámetros óptimos tras la validación cruzada. Así, el objetivo será conocer en qué posiciones de la imagen se sitúan las personas.

Otro de los aspectos fundamentales de este ejercicio es tener que barrer la imagen en busca de personas a diferentes tamaños. Para ello, lo más lógico parece ser aplicar una transformación de escalado sobre la imagen a estudiar, de forma que se re-escalará en diferentes tamaños, teniendose así varias imágenes para analizar (aunque en realidad son una misma imagen a distintos tamaños). Así pues, se realizará un barrido de cada una de estas imágenes mediante la aplicación de ventanas de 128×64, con un desplazamiento de 16×16. Una vez localizada la presencia de personas tras la aplicación de nuestro clasificador, se deshará dicha transformación de escalado, obteniéndose la región en la imagen real en la cual el clasificador ha predicho la presencia de una persona.

Tras el barrido de las distintas imágenes se realizará un estudio de los resultados (que serán rectángulos cuyos vértices serán los de las ventanas etiquetadas como personas por el clasificador), de modo que se buscará realizar un agrupamiento de aquellos rectángulos que

hagan una descripción de la misma persona. Aunque *OpenCV* cuenta con una función capaz de realizar *clusters* de los rectángulos encontrados y que se invoca mediante el comando *cv2.grouprectangle*, no parece ser funcionar del todo bien. Por ello, el autor ha realizado la **función** *iou_rectangle_merge* que, dados dos rectángulos con un cierto solape, se formará un rectángulo capaz de contener a ambos rectángulos originales siempre y cuando el cociente entre el área de solape de los rectángulos y el área total de los rectángulos sea superior a un cierto umbral a elección del usuario.

En este punto hemos de citar 3 aspectos que serán clave a la hora de la calidad de nuestra búsqueda:

- **El factor de escalado**: Determinará cual es el factor con el que la imagen total se escala.
- El **intervalo de valores de escalado**: Determinarán las imágenes escaladas (con el factor anterior) que se consideran para el estudio.
- **El área umbral**: Dentro de la función de combinación de rectángulos habrá que definir un área a partir de la cual se decide si dos rectángulos se unen o no.

Por tanto, la calidad de nuestros resultados se verá fuertemente influido por estos parámetros. De tal modo, se elegirán algunas de las imágenes encontradas en la web, procediendo a la búsqueda de personas mediante los descriptores empleados en todo el trabajo. Las imágenes que han sido seleccionadas, una en escala de grises y otra en color (aunque para obtener las características esta será transformada en escala de grises), son las siguientes:

En cuanto a las dimensiones de estas imágenes, la imagen en escala de grises tiene un tamaño de 1000×800, mientras que en el caso de la imagen en color es de 600×400. Por tanto, el objetivo será utilizar los clasificadores entrenados con las características de los descriptores estudiados en este trabajo y comprobar su capacidad de detección en dichas imágenes:

6.1. Detección a través del clasificador basado en el descriptor HOG

En primer lugar, se ha analizado la imagen utilizando el clasificador SVM con los parámetros correspondientes a los que mejores resultados ofrecían en la validación cruzada realizada con las características del **descriptor HOG**. Así, se ha barrido, mediante ventanas de 128×64 las dos imágenes de estudio, así como distintas versiones de éstas a otras escalas:

Imagen en escala de grises

Para el caso de la imagen en escala de grises se ha definido una **escala** de 0.04, por lo que la imagen se ha hecho muy pequeñita, y unos **valores de escalado** entre 4 y 13. Por su parte, **valor umbral** para la fusión de rectángulos se ha definido como 0.2. El resultado obtenido es el siguiente:

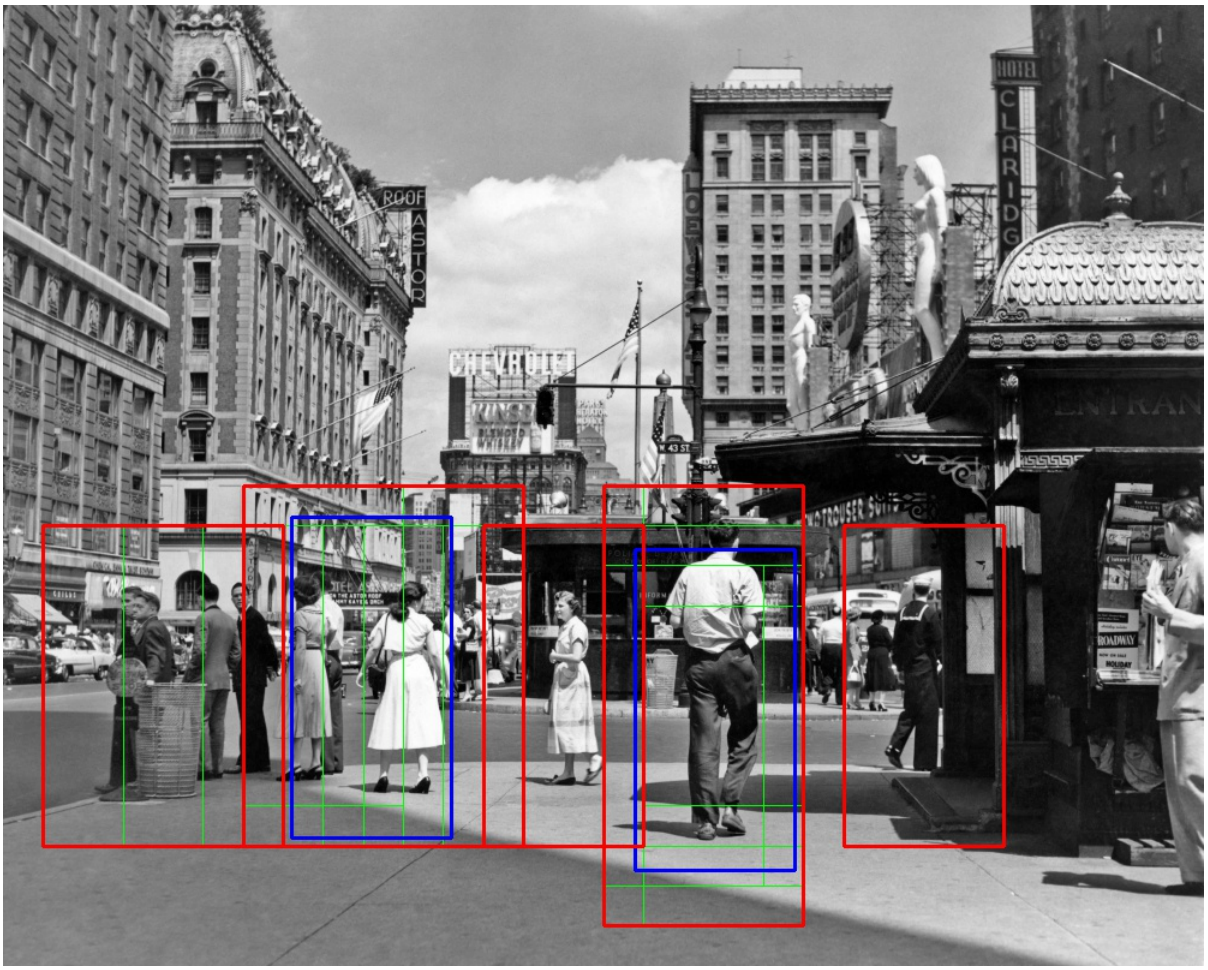
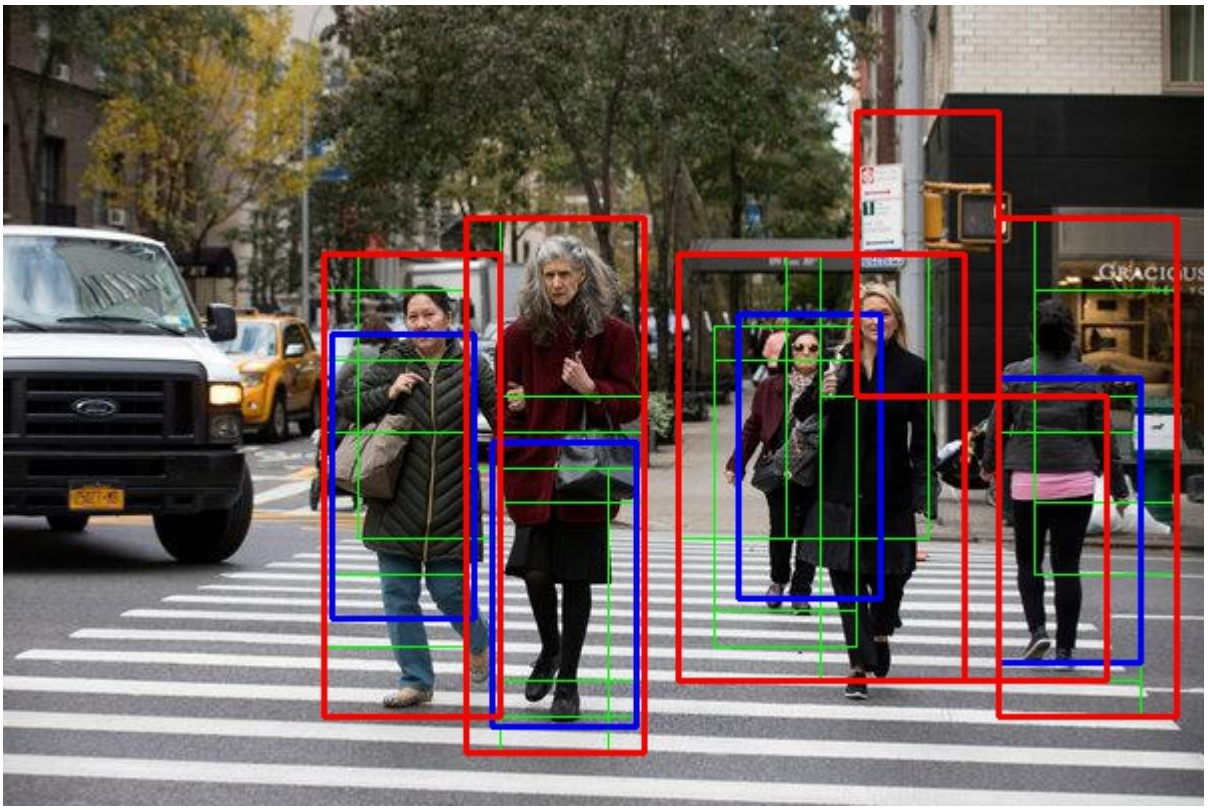


Imagen en color

Por su parte, para el caso de la imagen en color se ha definido una **escala** de 0.1 y unos **valores de escalado** entre 5 y 9. Por su parte, **valor umbral** para la fusión de rectángulos se ha definido como 0.2. El resultado obtenido es el siguiente:



Los rectángulos verdes son las ventanas definidas como personas por el clasificador. Se aprecia que no hay una coincidencia exacta entre rectángulos, pues ello depende de las posiciones tomadas en el barrido por las ventanas para las distintas versiones (tamaños) de la imagen. Por su parte, los cuadrados azules hacen referencia a los rectángulos definidos por *clustering* mediante la función *groupRectangle* de *OpenCV*, mientras que los cuadrados rojos son los rectángulos definidos por el algoritmo de *iou_rectangle_merge* creado por el autor para la unión de rectángulos.

Vemos como **la detección de personas es notablemente correcta con este algoritmo** en ambos casos, aunque algo mejor en el caso de la imagen en grises. En ambos casos realiza una detección de personas muy positiva, de modo que todas las personas en primera plana son correctamente detectadas. En el caso de la imagen en color vemos alguna ventana definida erróneamente como persona.

6.2. Detección a través del clasificador basado en el descriptor LBP

Asímismo, se ha analizado la imagen utilizando el clasificador SVM con los parámetros correspondientes a los que mejores resultados ofrecían en la validación cruzada realizada con las características del descriptor LBP:

Imagen en escala de grises

Para el caso de este descriptor se han mantenido la **escala** y los **valores de escalado**, con valores de 0.04 y entre 4 y 13, respectivamente. Sin embargo, en este caso se ha definido **valor umbral** para la fusión de rectángulos como 0.5. El resultado obtenido es el siguiente:

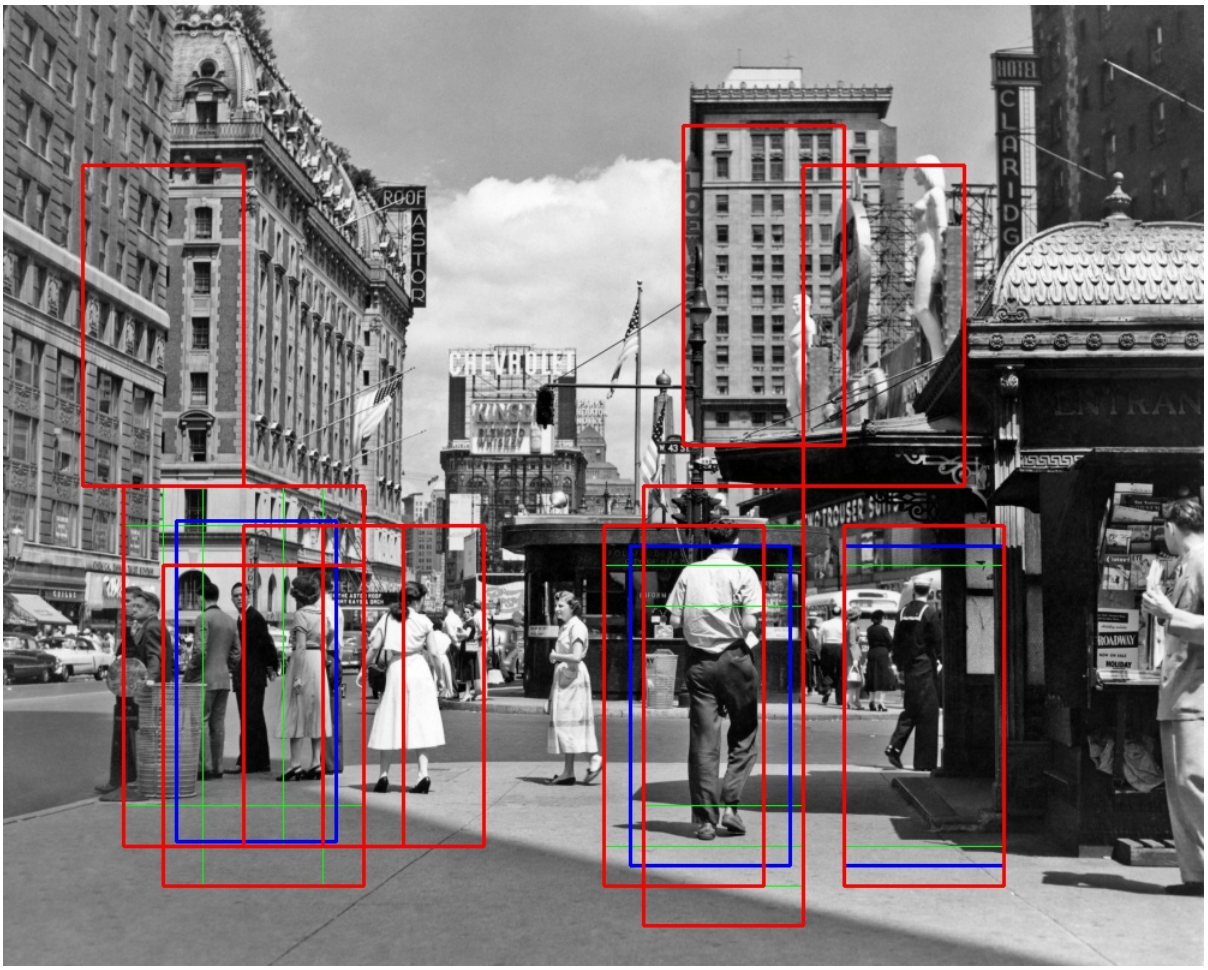
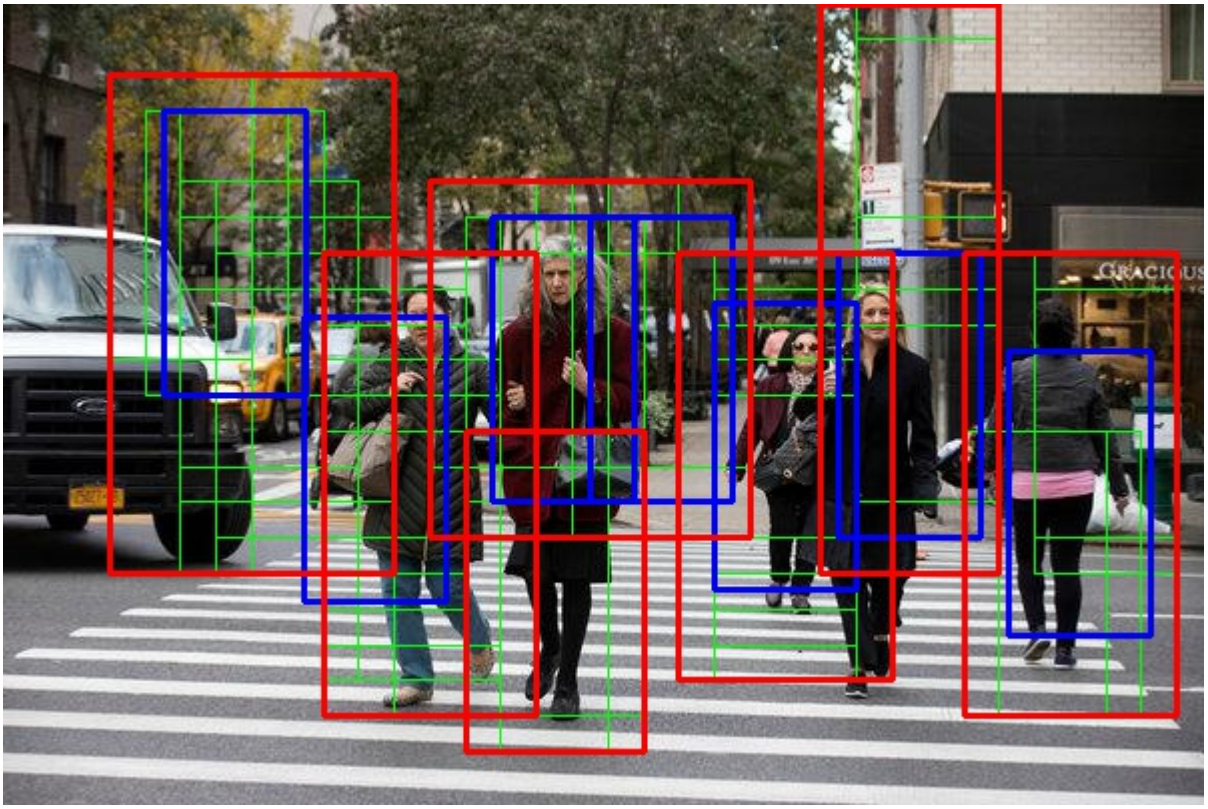


Imagen en color

En el caso del descriptor LBP se han mantenido todos los parámetros, con una **escala** de 0.1, unos **valores de escalado** entre 5 y 10 y un **valor umbral** para la fusión de rectángulos de 0.2. El resultado obtenido es el siguiente:



Se aprecia como la clasificación realizada por el clasificador basado en el descriptor LBP tiene un menor rendimiento en ambas imágenes. En el caso de la imagen en grises vemos como alguna persona de primera plana se queda sin detectar y, además, toma como personas a un par de elementos del paisaje. No obstante, se puede apreciar como uno es una especie de estatua con forma humana, por lo que podemos comprender esta confusión en el clasificador. En el caso de la imagen en color se aprecia una calidad similar en la detección de personas, aunque de nuevo clasifica como persona a una región donde en realidad no la hay.

6.3. Detección a través del clasificador basado en el descriptor U-LBP

Por otro lado, se ha analizado la imagen utilizando el clasificador SVM basado en los mejores parámetros dadas las características del descriptor U-LBP:

Imagen en escala de grises

Para el caso de este descriptor se han mantenido los parámetros de **escala** y los **valores de escalado**, con valores de 0.04 y entre 4 y 13, respectivamente. No obstante, en este caso se ha definido un **valor umbral** para la fusión de rectángulos de 0.4. El resultado obtenido es el siguiente:

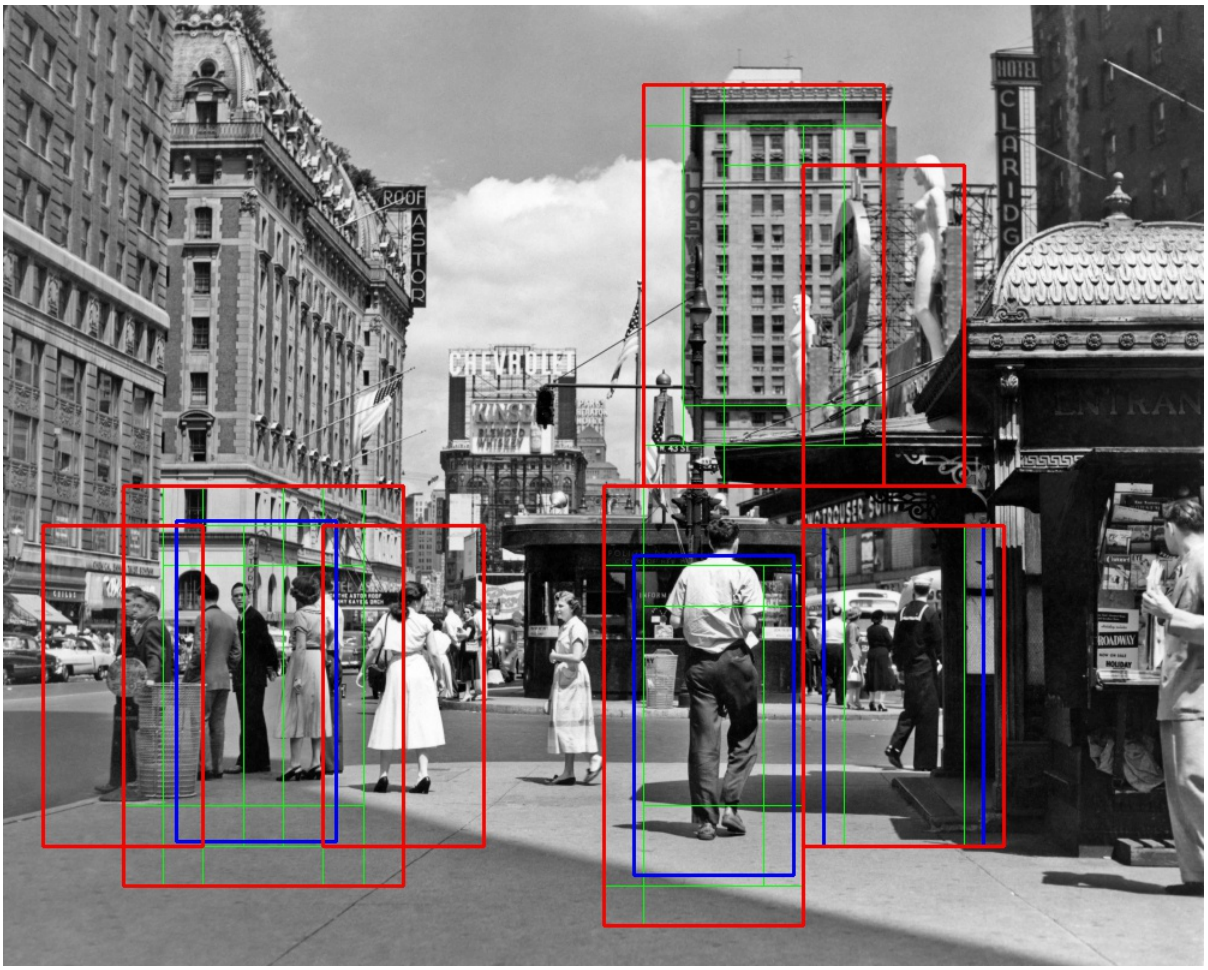
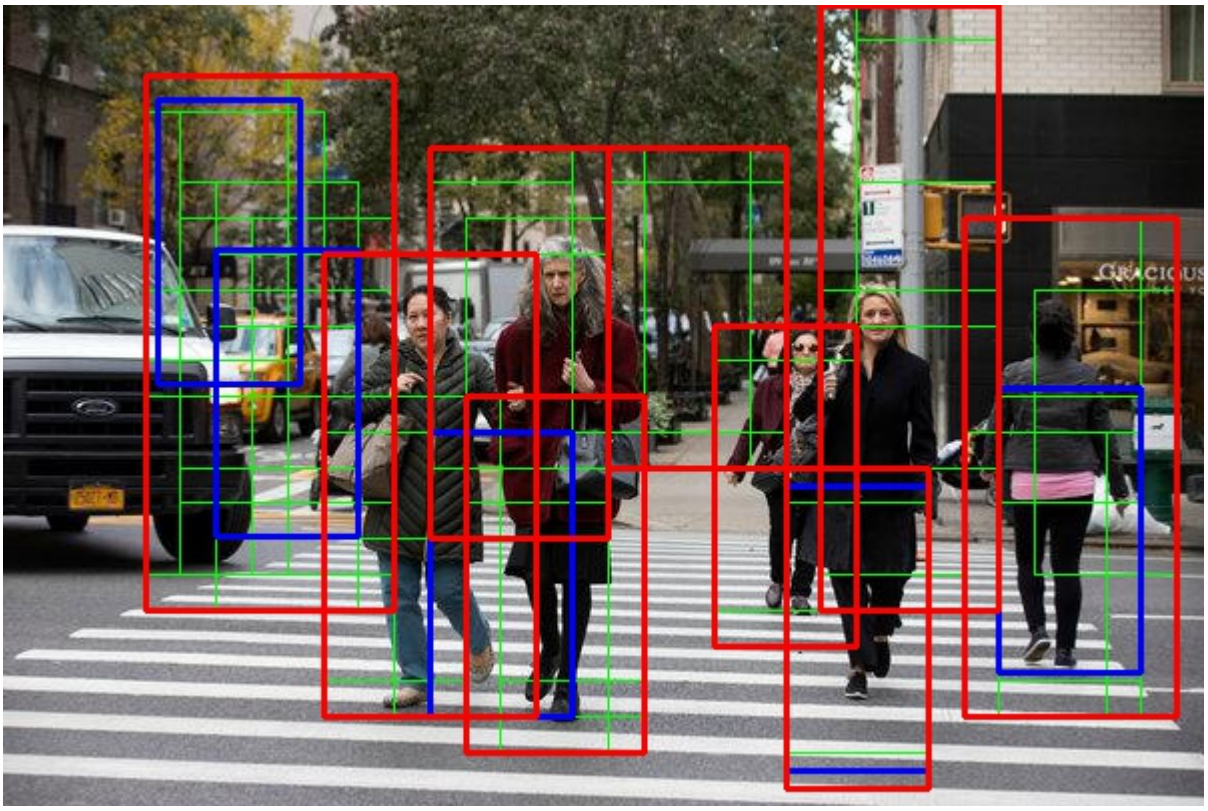


Imagen en color

Para el caso de este descriptor se han mantenido los parámetros de los casos HOG y LBP, tomando valores de 0.1 para la **escala**, entre 5 y 10 para los **valores de escalado** y 0.2 para el **valor umbral** para la fusión de rectángulos, respectivamente. El resultado obtenido es el siguiente:



Se aprecia como la clasificación realizada en base al descriptor U-LBP es muy similar a la realizada por el clasificador basado en el descriptor LBP. Así, de nuevo en ambas imágenes se aprecia la clasificación como persona de regiones donde en realidad no la hay. Además, vemos como en la imagen en color el método de agrupación de rectángulos de *OpenCV* empeora su calidad, tomando como centroides del *cluster* a regiones donde no hay personas.

6.4. Detección a través del clasificador basado en el descriptor HOG+LBP

Por último, se ha analizado la imagen utilizando el clasificador SVM con los parámetros correspondientes a los que mejores resultados ofrecían en la validación cruzada realizada con la combinación de las características de los descriptor HOG y LBP.

Imagen en escala de grises

De nuevo en este caso sese ha definido los parámetros de una **escala** y **valores de escalado** con valores de 0.04 y entre 4 y 13, respectivamente. Por su parte, el **valor umbral** para la fusión de rectángulos se ha definido como 0.5, al igual que en el caso del descriptor LBP. El resultado obtenido es el siguiente:

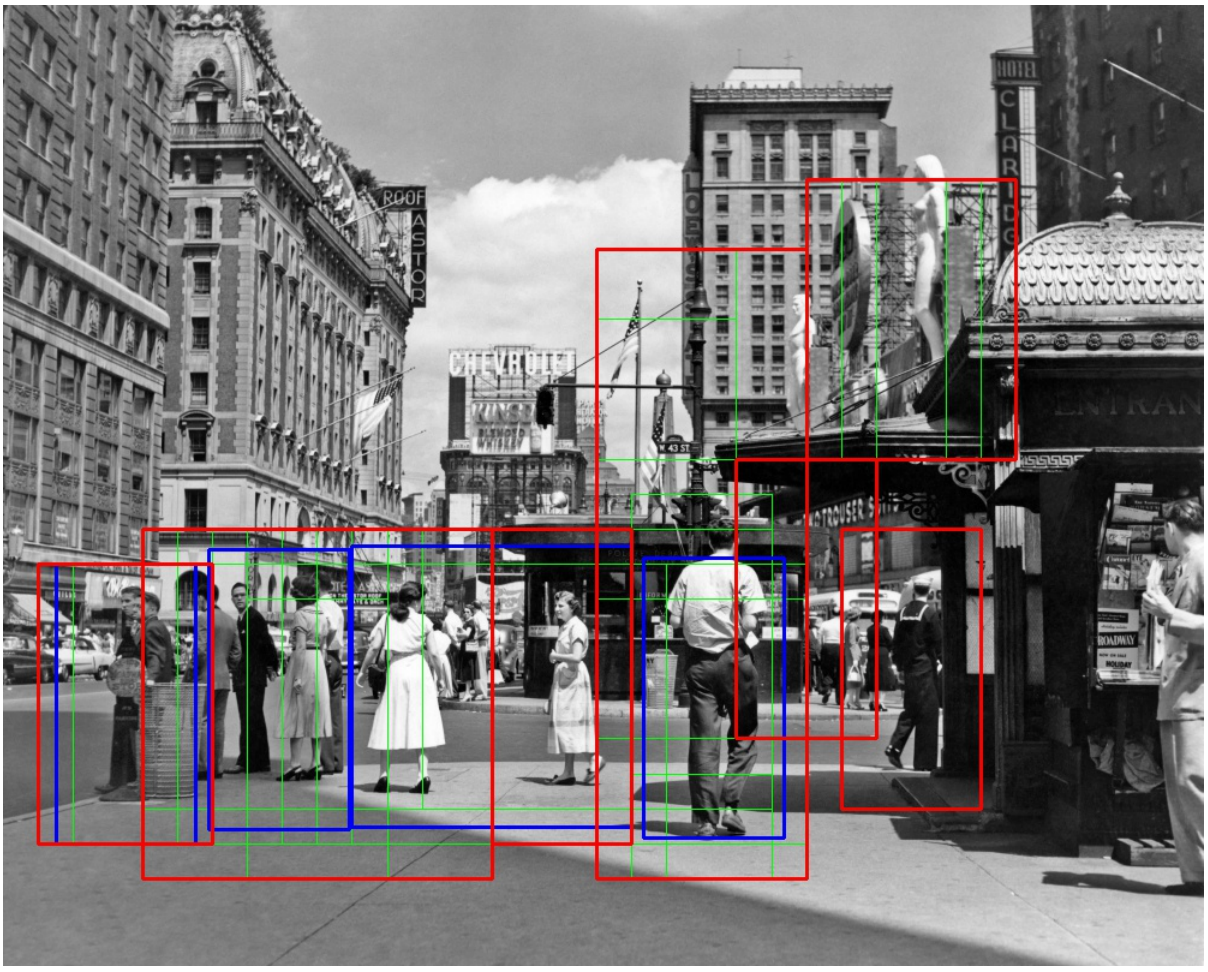
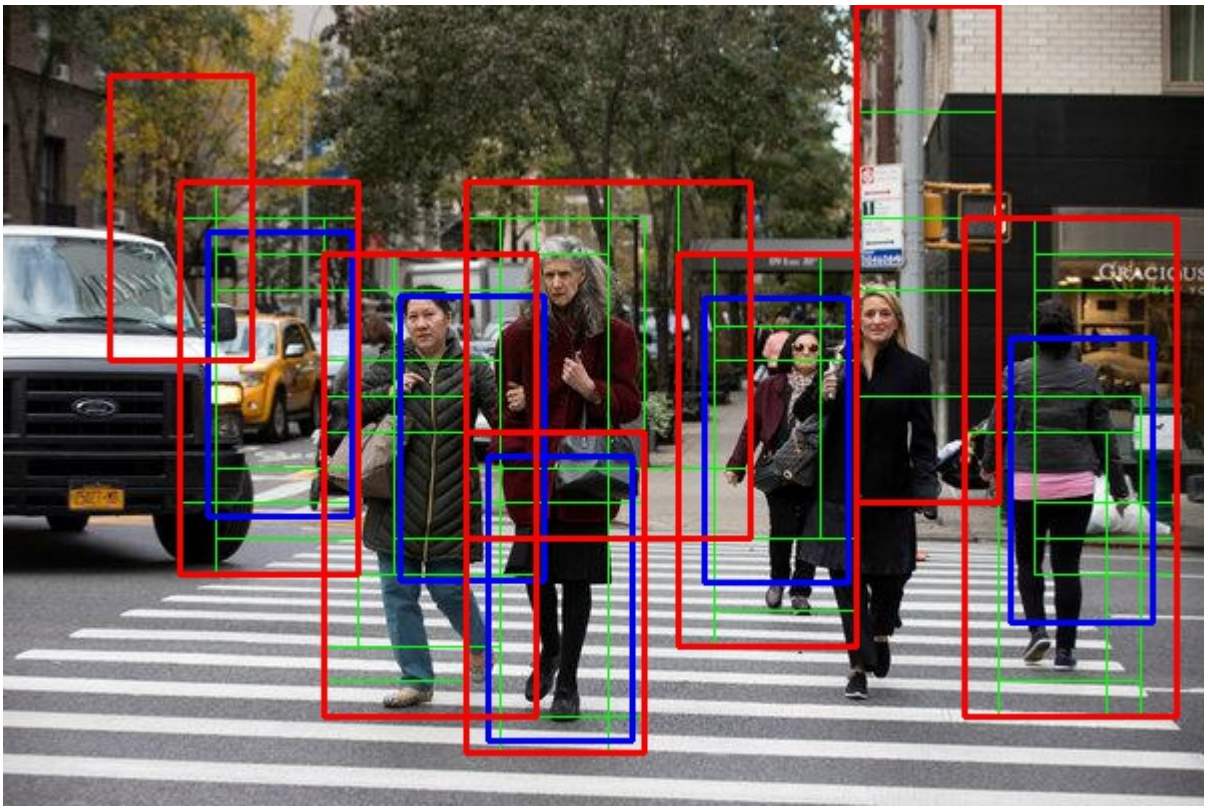


Imagen en color

Para el caso de este descriptor HOG+LBP se han mantenido los parámetros de otros descriptores, tomando valores de 0.1 para la **escala**, entre 5 y 10 para los **valores de escalado** y 0.2 para el **valor umbral** para la fusión de rectángulos, respectivamente. El resultado obtenido es el siguiente:



Podemos ver como la calidad de la clasificación con estas características tiene resultados diferentes según la imagen. En el caso de la imagen en grises se aprecian resultados muy similares a los del clasificador basado en HOG e incluso detecta alguna región donde hay una forma humana como es el caso de la estatua mencionada, por lo que la detección es bastante positiva. Sin embargo, en el caso de la imagen de color se ve una clasificación más similar a la del método LBP, con varios falsos positivos. Además, incluso podriamos decir que no termina de detectar a alguna de las personas. Por tanto, los resultados son dispares en este caso y no son todo lo positivos que cabría esperar según la evaluación de la bondad realizada previamente.

7. Conclusiones

En conclusión, en este trabajo se ha podido realizar una introducción para la aplicación de la inteligencia computacional para la detección de objetos. Para ello, hemos hecho uso de los algoritmos presentados en clase, como son los descriptores HOG, LBP, U-LBP y la combinación de los dos primeros.

En primer lugar, **se ha analizado la calidad de los distintos métodos mediante la evaluación de la bondad del clasificador optimizado**. Se ha aplicado técnicas de validación cruzada con 5 *folds* y se puede apreciar como, en general, los métodos funcionan bastante bien, con errores muy bajos entorno al 1 %. Además, cabe destacar que todos los métodos suelen presentar mayor cantidad de **errores del tipo falso positivo**, clasificando como ventanas con persona a aquellas donde en realidad no lo hay. Por tanto, queda claro que, aunque muy buenos, no son infalibles. Se ha comprobado como **el mejor método obtenido es el HOG+LBP**. Por tanto, la combinación de los rasgos del descriptor HOG con los de LBP es muy positiva y optimiza la clasificación en nuestro clasificador.

En segundo lugar, se ha aplicado esto al **estudio de una imagen cualquiera con el objetivo de detectar personas**. Aunque se aprecian ciertos fallos en la detección de personas, por lo general realiza clasificaciones de personas correctas. Como hemos dicho antes, presenta mayor cantidad de falsos positivos, pues suele fallar clasificando ventanas en las que no hay ninguna persona como si sí la hubiese. Entre los problemas encontrados en este ejemplo encontramos la adecuada elección de los parámetros definidos en el marco teórico. Además, también se ha encontrado bastante dificultad a la hora de realizar una adecuada agrupación de los conjunto de ventanas solapadas en ventanas únicas, debido a que pueden haber varias que hagan referencia a una misma persona. De tal modo, se ha implementado manualmente un algoritmo que realice esto. Asímismo, también se ha comprobado como la propia imagen (calidad, etc.) afectan en la capacidad del clasificador para detectar a las personas correctamente.

Este tipo de desarrollos puede dar lugar a **sistemas inteligentes muy útiles**, enfocados al diagnóstico médico y a la monitorización de la seguridad y vigilancia de lugares y personas mediante la adecuada implementación en el algoritmo de las imposiciones pertinentes según el caso. Un ejemplo claro es el presentado en clase para la detección de caídas de personas en un centro de cuidado de ancianos.

8. Apéndice: Tablas con los resultados de la validación cruzada para los distintos parámetros considerados

8.1. Descriptor HOG

Descriptor HOG						
C	**Degree**	**γ**	**Kernel**	**Accuracy**	**AUC**	**F1-Score**
0.1	-	0.01	rbf	0.9479795633999071	0.9479117323061261	0.9418800280415919
0.1	-	0.01	linear	0.9751509521597771	0.9740447768322458	0.9718500441630079
0.1	2	0.01	poly	0.9521597770552717	0.9516773853814298	0.9463083999504509
0.1	3	0.01	poly	0.9577333952624245	0.9565430932596061	0.9521921482662121
0.1	-	0.1	rbf	0.555039479795634	0.5	0.0
0.1	-	0.1	linear	0.9751509521597771	0.9740447768322458	0.9718500441630079
0.1	2	0.1	poly	0.9781699953553181	0.9769191801660201	0.9752226059231116
0.1	3	0.1	poly	0.979098931723177	0.9777560488342403	0.9762550798578707
0.1	-	1	rbf	0.555039479795634	0.5	0.0
0.1	-	1	linear	0.9751509521597771	0.9740447768322458	0.9718500441630079
0.1	2	1	poly	0.9774732930794241	0.976084813037281	0.9744037549518474
0.1	3	1	poly	0.979098931723177	0.9777560488342403	0.9762550798578707
1	-	0.01	rbf	0.9770088248954947	0.9755626326445102	0.9738491733049992
1	-	0.01	linear	0.9714352066883418	0.9702317614566945	0.9676184039646016
1	2	0.01	poly	0.9758476544356711	0.9741543140415203	0.9724664141223776
1	3	0.01	poly	0.9779377612633534	0.9761929808861699	0.9748217266065184
1	-	0.1	rbf	0.9579656293543892	0.9559224200774221	0.9520010031960618
1	-	0.1	linear	0.9714352066883418	0.9702317614566945	0.9676184039646016
1	2	0.1	poly	0.9774732930794241	0.976084813037281	0.9744037549518474
1	3	0.1	poly	0.979098931723177	0.9777560488342403	0.9762550798578707
1	-	1	rbf	0.555039479795634	0.5	0.0
1	-	1	linear	0.9714352066883418	0.9702317614566945	0.9676184039646016
1	2	1	poly	0.9774732930794241	0.976084813037281	0.9744037549518474
1	3	1	poly	0.979098931723177	0.9777560488342403	0.9762550798578707
10	-	0.01	rbf	0.9811890385508593	0.9801560114685229	0.978680037042342
10	-	0.01	linear	0.9714352066883418	0.9702317614566945	0.9676184039646016
10	2	0.01	poly	0.9781699953553181	0.9769191801660201	0.9752226059231116
10	3	0.01	poly	0.9797956339990711	0.9783323933956226	0.9770162988914273
10	-	0.1	rbf	0.9591267998142127	0.9571238408608199	0.9533374840352627
10	-	0.1	linear	0.9714352066883418	0.9702317614566945	0.9676184039646016
10	2	0.1	poly	0.9774732930794241	0.976084813037281	0.9744037549518474
10	3	0.1	poly	0.979098931723177	0.9777560488342403	0.9762550798578707
10	-	1	rbf	0.555039479795634	0.5	0.0
10	-	1	linear	0.9714352066883418	0.9702317614566945	0.9676184039646016
10	2	1	poly	0.9774732930794241	0.976084813037281	0.9744037549518474
10	3	1	poly	0.979098931723177	0.9777560488342403	0.9762550798578707

8.2. Descriptor LBP

Descriptor HOG						
C	**Degree**	**γ**	**Kernel**	**Accuracy**	**AUC**	**F1-Score**
0.1	-	0.01	rbf	0.9737575476079888	0.9742908917413289	0.9707756860265879
0.1	-	0.01	linear	0.9914073385973061	0.9915875587181352	0.9903751556901438
0.1	2	0.01	poly	0.9772410589874594	0.9775844711862807	0.9745873912349845
0.1	3	0.01	poly	0.9793311658151417	0.9798290381417682	0.9769522549370728
0.1	-	0.1	rbf	0.555039479795634	0.5	0.0
0.1	-	0.1	linear	0.9914073385973061	0.9915875587181352	0.9903751556901438
0.1	2	0.1	poly	0.9932652113330237	0.9935198304853057	0.9924596787203169
0.1	3	0.1	poly	0.993032977241059	0.9933619332670295	0.9922044074581382
0.1	-	1	rbf	0.555039479795634	0.5	0.0
0.1	-	1	linear	0.9914073385973061	0.9915875587181352	0.9903751556901438
0.1	2	1	poly	0.9932652113330237	0.9935198304853057	0.9924596787203169
0.1	3	1	poly	0.993032977241059	0.9933619332670295	0.9922044074581382
1	-	0.01	rbf	0.9900139340455179	0.9903836363952567	0.988837058947691
1	-	0.01	linear	0.9914073385973061	0.9915875587181352	0.9903751556901438
1	2	0.01	poly	0.9916395726892708	0.9919514905175907	0.9906470736201825
1	3	0.01	poly	0.9921040408732001	0.9923700463130459	0.9911606636574429
1	-	0.1	rbf	0.9245239201114723	0.9310274590955377	0.9212090084306236
1	-	0.1	linear	0.9914073385973061	0.9915875587181352	0.9903751556901438
1	2	0.1	poly	0.9932652113330237	0.9935198304853057	0.9924596787203169
1	3	0.1	poly	0.993032977241059	0.9933619332670295	0.9922044074581382
1	-	1	rbf	0.555039479795634	0.5	0.0
1	-	1	linear	0.9914073385973061	0.9915875587181352	0.9903751556901438
1	2	1	poly	0.9932652113330237	0.9935198304853057	0.9924596787203169
1	3	1	poly	0.993032977241059	0.9933619332670295	0.9922044074581382
10	-	0.01	rbf	0.9934974454249884	0.9937804890624847	0.9927206945058119
10	-	0.01	linear	0.9914073385973061	0.9915875587181352	0.9903751556901438
10	2	0.01	poly	0.9932652113330237	0.9935198304853057	0.9924596787203169
10	3	0.01	poly	0.993032977241059	0.9933619332670295	0.9922044074581382
10	-	0.1	rbf	0.9284718996748723	0.9345845760501811	0.9250172729768171
10	-	0.1	linear	0.9914073385973061	0.9915875587181352	0.9903751556901438
10	2	0.1	poly	0.9932652113330237	0.9935198304853057	0.9924596787203169
10	3	0.1	poly	0.993032977241059	0.9933619332670295	0.9922044074581382
10	-	1	rbf	0.555039479795634	0.5	0.0
10	-	1	linear	0.9914073385973061	0.9915875587181352	0.9903751556901438
10	2	1	poly	0.9932652113330237	0.9935198304853057	0.9924596787203169
10	3	1	poly	0.993032977241059	0.9933619332670295	0.9922044074581382

8.3. Descriptor HOG+LBP

Descriptor HOG						
C	**Degree**	**γ**	**Kernel**	**Accuracy**	**AUC**	**F1-Score**
0.1	-	0.01	rbf	0.9807245703669298	0.9817047786711876	0.9786092035392252
0.1	-	0.01	linear	0.9932652113330237	0.9932602983458212	0.9924378822131009
0.1	2	0.01	poly	0.9911751045053414	0.9914810551883069	0.990129427714501
0.1	3	0.01	poly	0.9951230840687413	0.995347965791787	0.9945397504700542
0.1	-	0.1	rbf	0.555039479795634	0.5	0.0
0.1	-	0.1	linear	0.9932652113330237	0.9932602983458212	0.9924378822131009
0.1	2	0.1	poly	0.9951230840687413	0.9952448270398521	0.99453163228223
0.1	3	0.1	poly	0.995355318160706	0.9955577425438915	0.9947984186329326
0.1	-	1	rbf	0.555039479795634	0.5	0.0
0.1	-	1	linear	0.9932652113330237	0.9932602983458212	0.9924378822131009
0.1	2	1	poly	0.9951230840687413	0.9952448270398521	0.99453163228223
0.1	3	1	poly	0.995355318160706	0.9955577425438915	0.9947984186329326
1	-	0.01	rbf	0.9944263817928471	0.9947204964826396	0.9937624114175894
1	-	0.01	linear	0.9932652113330237	0.9932602983458212	0.9924378822131009
1	2	0.01	poly	0.9951230840687413	0.9952448270398521	0.99453163228223
1	3	0.01	poly	0.995355318160706	0.9955577425438915	0.9947984186329326
1	-	0.1	rbf	0.555039479795634	0.5	0.0
1	-	0.1	linear	0.9932652113330237	0.9932602983458212	0.9924378822131009
1	2	0.1	poly	0.9951230840687413	0.9952448270398521	0.99453163228223
1	3	0.1	poly	0.995355318160706	0.9955577425438915	0.9947984186329326
1	-	1	rbf	0.555039479795634	0.5	0.0
1	-	1	linear	0.9932652113330237	0.9932602983458212	0.9924378822131009
1	2	1	poly	0.9951230840687413	0.9952448270398521	0.99453163228223
1	3	1	poly	0.995355318160706	0.9955577425438915	0.9947984186329326
10	-	0.01	rbf	0.9951230840687413	0.9953483431848187	0.9945394548449419
10	-	0.01	linear	0.9932652113330237	0.9932602983458212	0.9924378822131009
10	2	0.01	poly	0.9951230840687413	0.9952448270398521	0.99453163228223
10	3	0.01	poly	0.995355318160706	0.9955577425438915	0.9947984186329326
10	-	0.1	rbf	0.5587552252670692	0.5041750659512483	0.016533255151795365
10	-	0.1	linear	0.9932652113330237	0.9932602983458212	0.9924378822131009
10	2	0.1	poly	0.9951230840687413	0.9952448270398521	0.99453163228223
10	3	0.1	poly	0.995355318160706	0.9955577425438915	0.9947984186329326
10	-	1	rbf	0.555039479795634	0.5	0.0
10	-	1	linear	0.9932652113330237	0.9932602983458212	0.9924378822131009
10	2	1	poly	0.9951230840687413	0.9952448270398521	0.99453163228223
10	3	1	poly	0.995355318160706	0.9955577425438915	0.9947984186329326

8.4. Descriptor U-LBP

Descriptor HOG						
C	**Degree**	**γ**	**Kernel**	**Accuracy**	**AUC**	**F1-Score**
0.1	-	0.01	rbf	0.9753831862517418	0.976014249994807	0.9726097353054719
0.1	-	0.01	linear	0.990710636321412	0.990648946676358	0.9895680332764343
0.1	2	0.01	poly	0.9763121226196005	0.976851496056059	0.9736148228135112
0.1	3	0.01	poly	0.9786344635392475	0.9790473053329211	0.9761556197991148
0.1	-	0.1	rbf	0.8392940083604273	0.8543005988228887	0.8461913692612651
0.1	-	0.1	linear	0.990710636321412	0.990648946676358	0.9895680332764343
0.1	2	0.1	poly	0.9921040408732001	0.9921112689596255	0.9911362522974732
0.1	3	0.1	poly	0.9928007431490943	0.9928424973364298	0.9919190030936181
0.1	-	1	rbf	0.555039479795634	0.5	0.0
0.1	-	1	linear	0.990710636321412	0.990648946676358	0.9895680332764343
0.1	2	1	poly	0.9921040408732001	0.9921112689596255	0.9911362522974732
0.1	3	1	poly	0.9928007431490943	0.9928424973364298	0.9919190030936181
1	-	0.01	rbf	0.9904784022294473	0.9906987020637377	0.9893385727740532
1	-	0.01	linear	0.9916395726892708	0.9915895744122353	0.9906077271618529
1	2	0.01	poly	0.9900139340455179	0.9901770075163475	0.9888100866868869
1	3	0.01	poly	0.9909428704133767	0.9910136332618774	0.989839394919602
1	-	0.1	rbf	0.9672549930329772	0.9696746578087283	0.9642541714294798
1	-	0.1	linear	0.9916395726892708	0.9915895744122353	0.9906077271618529
1	2	0.1	poly	0.9921040408732001	0.9921112689596255	0.9911362522974732
1	3	0.1	poly	0.9928007431490943	0.9928424973364298	0.9919190030936181
1	-	1	rbf	0.555039479795634	0.5	0.0
1	-	1	linear	0.9916395726892708	0.9915895744122353	0.9906077271618529
1	2	1	poly	0.9921040408732001	0.9921112689596255	0.9911362522974732
1	3	1	poly	0.9928007431490943	0.9928424973364298	0.9919190030936181
10	-	0.01	rbf	0.9925685090571296	0.9925814613662189	0.9916569593578357
10	-	0.01	linear	0.9916395726892708	0.9915895744122353	0.9906077271618529
10	2	0.01	poly	0.9921040408732001	0.9921112689596255	0.9911362522974732
10	3	0.01	poly	0.9928007431490943	0.9928424973364298	0.9919190030936181
10	-	0.1	rbf	0.9693450998606595	0.9715579289745826	0.9664693737690672
10	-	0.1	linear	0.9916395726892708	0.9915895744122353	0.9906077271618529
10	2	0.1	poly	0.9921040408732001	0.9921112689596255	0.9911362522974732
10	3	0.1	poly	0.9928007431490943	0.9928424973364298	0.9919190030936181
10	-	1	rbf	0.555039479795634	0.5	0.0
10	-	1	linear	0.9916395726892708	0.9915895744122353	0.9906077271618529
10	2	1	poly	0.9921040408732001	0.9921112689596255	0.9911362522974732
10	3	1	poly	0.9928007431490943	0.9928424973364298	0.9919190030936181