UNIVERSIDAD DE GRANADA

MÁSTER DE CIENCIA DE DATOS E INGENIERÍA DE COMPUTADORES

CURSO ACADÉMICO 2019-2020

EXTRACCIÓN DE CARACTERÍSTICAS EN IMÁGENES

Sistemas de clasificación de imágenes basados en características

Desarrollo y análisis de modelos de clasificación de imágenes basados en SVM haciendo uso de diferentes tipos de descriptores.

Nicolás Cubero

6 de Marzo de 2020

Índice

Índice de figuras

Índice de tablas

1. Introducción

En el presente proyecto se pretende llevar a cabo el desarrollo y análisis de diferentes clasificadores de imágenes basados en SVM (*Support Vector Machine*) haciendo uso de diferentes técnicas de extracción de características en imágenes. En concreto, se pretende abordar el problema de detección de peatones en imágenes, lo que supone un problema de clasificación con dos tipos de imágenes: imágenes con peatón e imágenes que muestran un fondo sin peatones.

El desarrollo de estos clasificadores se hará haciendo uso de diferentes técnicas de extracción de características en imágenes: descriptores basados en *Histogram of Gradient* (HOG), descriptores basados en *Local Binnary Pattern* (LBP) en su versión básica y también basados en LBP Uniforme.

Se llevarán a cabo diversas experimentaciones, considerando estas técnicas de extracción de rasgos, diferentes tipos de *kernel* SVM, valores de constante de regularización C y otros parámetros relativos a los modelos de SVM, realizando un estudio de su rendimiento atendiendo a diferentes métricas y mecanismos de particionamiento de conjunto de entrenamiento y test.

2. Descripción de *scripts* y del entorno de ejecución

Primeramente, se describirá de forma breve todos los *scripts* elaborados a lo largo de todo el proyecto, tanto los *scripts* ejecutables como los módulos implementados y usados por los anteriores, indicándose para cada uno, su funcionamiento y su uso.

La implementación de los *scripts*, así como la ejecución de todos los experimentos se llevaron a cabo bajo el entorno de desarrollo **Spyder** [1], uno de los entornos de ejecución para el lenguaje *Python* más destacados en el ámbito de la ciencia de datos.

Spyder resulta de utilidad, en este proyecto, tanto por ofrecer funciones de verificación de la sintáxis del código, como por permitir la ejecución de los *scripts* en una terminal interactiva que permite controlar de forma completa la ejecución del *script*.

[1] https://www.spyder-ide.org/

No obstante, la versión final de los *scripts* elaborados pueden ser ejecutados directamente admitiendo la parametrización completa de su ejecución por medio de argumentos en la línea de comandos y ficheros *JSON* de configuración.

2.1. Scripts ejecutables

Se trata de los scripts empleados para la elaboración y evaluación de los modelos y la clasificación de imágenes de entrada a partir de los mismos:

2.1.1. train_HOG_SVM_kfold.py

Permite entrenar clasificadores SVM haciendo uso de descriptores HOG computados a partir de las imágenes que constituyen el conjunto de datos y evalúa su *accuracy* medio considerando un particionamiento *kfold* estratificado. Este script no guarda ningún modelo generado pero escribe los resultados de la experimentación en otro fichero *JSON* cuyo nombre de fichero es generado a partir del nombre del fichero *JSON* pasado como argumento.

Debe de ser usado de la siguiente forma:

```
train_HOG_SVM_kfold.py <Documento JSON de experimentación>
```

Por su parte, el documento *JSON* especificado al *script* deberá de presentar alguno de los siguientes campos:

- **kfold_splits**: Número de particiones a considerar en kfold. Debe de ser un valor entero mayor que 0. Por defecto se toman 5 particiones.
- **kernel_type** Tipo de kernel de SVM. Debe de ser un *string* con alguno de los siguientes valores: SVM_LINEAR, SVM_POLY, SVM_RBF y SVM_SIGMOID. Por defecto se toma el kernel por defecto en la implementación de SVM de *OpenCV*.
- **Cvalue**: Valor de la constante de regularización. Debe ser un valor real o lista de valores reales (si se quieren probar varios valores) mayor que 0. Si no se especifica, se toma el valor por defecto en la implementación de SVM de *OpenCV*.
- **degree**: Grado del polinomio para el kernel polinómico. Valor entero mayor que 0 o lista de valores enteros mayores que 0 (para el caso en el que se quieran probar varios valores).

- **gamma**: Valor de gamma para los kernels rbf, sigmoide y polinómico. Debe de ser un valor real o lista de valores reales si se desean probar varios valores. Si no se especifica, se tomará el valor por defecto en la implementación de SVM de *OpenCV*.

- **coef0**: Término independiente para los kernel sigmoidal y polinómico: Debe de ser un valor real o lista de valores reales para el caso en los que se deseen probar varios valores. Si no se especifica, se tomará el valor 0 por defecto.

Para el caso de los parámetros que admiten lista de valores, si se especifican listas de valores para estos parámetros, se realizará un experimento por cada combinación de valores de estos parámetros.

Un ejemplo de documento se expone a continuación:

```
{
  "kernel_type": "SVM_LINEAR",
  "Cvalue": [0.01, 0.05]
}
```

Script 1: Contenido del documento JSON "experimentos7_HOG_SVM_kfold_kernel_linear.json" donde se muestra un ejemplo de documento de experimentación admitido por el script.

2.1.2. train_LBP_SVM_kfold.py

Permite entrenar clasificadores SVM haciendo uso de descriptores LBP computados a partir de las imágenes que constituyen el conjunto de datos y evalúa su *accuracy* medio considerando un particionamiento *kfold* estratificado.
Debe de ser usado de la siguiente forma:

```
train_LBP_SVM_kfold.py <Documento JSON de experimentación>
```

El documento de experimentación admite el mismo formato que el *script*

`train_HOG_SVM_kfold.py`.

2.1.3. train_ULBP_SVM_kfold.py

Permite entrenar clasificadores SVM haciendo uso de descriptores LBP uniforme computados a partir de las imágenes que constituyen el conjunto de datos y evalúa su *accuracy* medio considerando un particionamiento *kfold*

estratificado.
Debe de ser usado de la siguiente forma:

```
train_ULBP_SVM_kfold.py <Documento JSON de experimentación>
```

El documento de experimentación admite el mismo formato que el *script* `train_HOG_SVM_kfold.py`.

2.1.4. train_HOG_SVM.py

Permite entrenar un clasificador SVM que identifica imágenes con peatones e imágenes de fondo haciendo uso de descriptores HOG. El entrenamiento se lleva a cabo haciendo uso de una partición fija de entrenamiento y test. Este *script* permite almacenar los modelos SVM generados.

Este *script* es usado de la siguiente forma:

```
train_HOG_SVM.py -d <Documento JSON de experimentación> [-s]
```

Con el parámetro -d se especifica la ruta al documento *JSON* de experimentación que debe de presentar alguno de los siguientes campos:

- **kernel_type** Tipo de kernel de SVM. Debe de ser un *string* con alguno de los siguientes valores: SVM_LINEAR, SVM_POLY, SVM_RBF y SVM_SIGMOID. Por defecto se toma el kernel por defecto en la implementación de SVM de *OpenCV*.
- **Cvalue**: Valor de la constante de regularización. Debe ser un valor real o lista de valores reales (si se quieren probar varios valores) mayor que 0. Si no se especifica, se toma el valor por defecto en la implementación de SVM de *OpenCV*.
- **degree**: Grado del polinomio para el kernel polinómico. Valor entero mayor que 0 o lista de valores enteros mayores que 0 (para el caso en el que se quieran probar varios valores).
- **gamma**: Valor de gamma para los kernels rbf, sigmoide y polinómico. Debe de ser un valor real o lista de valores realtes si se desean probar varios valores. Si no se especifica, se tomará el valor por defecto en la implementación de SVM de *OpenCV*.
- **coef0**: Término independiente para los kernel sigmoidal y polinómico: Debe de ser un valor real o lista de valores reales para el caso en los que se deseen probar varios valores. Si no se especifica, se tomará el valor 0 por defecto.

De igual modo que con los anteriores *scripts*, si se especifican listas de valores para los parámetros que admiten múltiples valores, se realizará un experimento por cada combinación de valores de estos parámetros.

Se aporta a continuación, un ejemplo de fichero de experimentación:

```
{
  "kernel_type": "SVM_RBF",
  "Cvalue": [0.001, 0.01, 0.05, 0.1, 0.5, 1, 2, 5],
  "gamma": [1e-3, 1e-2, 0.1, 0.5, 1, 2, 5, 10]
}
```

Script 2: Contenido del documento JSON "experimentos8_HOG_SVM_kfold_kernel_rbf.json" con un ejemplo de documento de experimentación.

Por su parte, si se especifica el parámetro *-s*, el *script* almacenará cada uno de los modelos generados en un fichero.

2.1.5. train_LBP_SVM.py

Permite entrenar un clasificador SVM que identifica imágenes con peatones e imágenes de fondo haciendo uso de descriptores LBP básicos. El entrenamiento se lleva a cabo haciendo uso de una partición fija de entrenamiento y test. Este *script* también permite almacenar los modelos SVM generados.

Este *script* es usado de la siguiente forma:

```
train_LBP_SVM.py -d <Documento JSON de experimentación> [-s]
```

Por su parte, el *script* admite la misma lista de argumentos y la misma parametrización en el documento *JSON* que el *script*

`train_HOG_SVM.py`.

2.1.6. train_ULBP_SVM.py

Permite entrenar un clasificador SVM que identifica imágenes con peatones e imágenes de fondo haciendo uso de descriptores LBP uniforme. El entrenamiento se lleva a cabo haciendo uso de una partición fija de entrenamiento y test. Este *script* también permite almacenar los modelos SVM generados.

Este *script* es usado de la siguiente forma:

```
train_ULBP_SVM.py -d <Documento JSON de experimentación> [-s]
```

Por su parte, el *script* admite la misma lista de argumentos y la misma parametrización en el documento *JSON* que el *script*

`train_HOG_SVM.py`.

2.1.7. test_HOG_SVM.py

Permite clasificar imágenes por medio de un modelo SVM preentrenado y haciendo uso de descriptores HOG y las muestra por la pantalla junto a su clasificación.

Se usa de la siguiente forma:

```
test_HOG_SVM.py -m <Fichero dat con el modelo SVM>
                -i <Rutas de imágenes a clasificar,...>
```

Con el parámetro *-m* o *–model* se especifica un fichero con un modelo SVM preentrenado que soporte descriptores HOG como entrada.

Por su parte, con el parámetro *-i* o *–images* se especifica las rutas de las imágenes que se desean clasificar.

2.2. Módulos implementados

Todos los *scripts* implementados hacen uso de funciones o clases de una serie de módulos implementados en el directorio *eci*.

- ***lbp.py***: Contiene la implementación de las clases LBPDescriptor y UniformLBPDescriptor, que representan utilidades para la extracción de rasgos basados en LBP y en LBP Uniforme respectivamente.
- ***utils.py***: Contiene diversas funciones de utilidad usadas por los *scripts* ejecutables.
- ***svm_functions***: Contienen diversas funciones que facilitan el entrenamiento y evaluación de modelos basados en SVM.

3. Experimentación y análisis de resultados

Con ayuda de los anteriores *scripts*, se realizan diferentes experimentaciones y se estudia el rendimiento de los modelos generados con la finalidad de encontrar el modelo más óptimo.

En la presente extracción, para cada uno de los métodos de extracción de rasgos, se realizan diferentes baterías de experimentos considerando diferentes parámetros del modelo SVM.

Se describirá, primeramente, las diferentes experimentaciones realizadas y se expondrán, los resultados obtenidos junto a una discusión que determinará el modelo óptimo obtenido.

3.1. Descriptores basados en Histogram of Oriented Gradient

En primer lugar, se hizo uso de descriptores basados en **Histogramas de gradiente orientado** (*Histogram of Oriented Gradient*).

En la primera experimentación, se trabaja con el ***kernel* lineal** de SVM y se evalúa el *accuracy* medio de los modelos mediante ***kfold* estratificado considerando 5 particiones**. Por su parte, se consideran los siguientes valores para el parámetro de regularización: $C \in \{0{,}001, 0{,}01, 0{,}05, 0{,}1, 0{,}5, 1, 1{,}5, 2, 2{,}5, 5\}$.

Los parámetros recogidos en esta experimentación, se recogen en el siguiente fichero *JSON*:

```
{
  "kernel_type": "SVM_LINEAR",
  "Cvalue": [0.01, 0.05]
}
```

Script 3: Contenido del documento JSON "experimentos7_HOG_SVM_kfold_kernel_linear.json" con el conjunto de parámetros probados en esta experimentación.

El *accuracy* medio evaluado en el proceso de validación cruzada, se recoge en la siguiente tabla:

C	accuracy medio	accuraccy mínimo	accuracy máximo
0.001	0.93550	0.92575	0.95592
0.01	0.97262	0.0.95592	0.98376
0.05	**0.97541**	**0.96056**	**0.99072**
0.1	0.97262	0.95824	0.98608
0.5	0.96613	0.94896	0.97680
1	0.96613	0.94896	0.97680
1.5	0.96613	0.94896	0.97680
2	0.96613	0.94896	0.97680
2.5	0.96613	0.94896	0.97680
5	0.96613	0.94896	0.97680

Tabla 1: Accuracy medio, mínimo y máximo obtenido mediante una evaluación con kfold estratificado para diferentes valores de C y considerando un kernel de SVM lineal.

Se aprecia como para valores del parámetro C iguales o superiores a 0.5, los modelos obtenidos presentan rendimientos similares caracterizado por un *accuracy* de 0.96613. **El valor de la contante de regularización C que ofrecen mejores resultados medios se da para C=0.05**.

Finalmente, tomando este mejor parámetro, se realizó una segunda experimentación sobre una partición fija de entrenamiento y test, obteniendo los siguientes resultados:

	accuracy	precission	recall	ROC-AUC
C=0.05	0.96545	0.9753	0.94800	0.96400

Tabla 2: Métricas de accuracy, precission, recall y área bajo la curva ROC del modelo SVM con kernel lineal, C=0.05 sobre la partición fija de entrenamiento y test.

La matriz de confusión de este modelo es la siguiente:

		Clase Real	
		+	-
Clase Predicha	+	474	26
	-	12	588

Tabla 3: Matriz de confusión del modelo con kernel lineal y C=0.05

Por último, se realizó otra batería de experimentos con el ***kernel* de base radial** (*kernel RBF*) y se considerando diferentes parámetros para la

constante de regularización C y para el parámetro *gamma*. La elaboración de los modelos se llevó a cabo mediante una partición fija de *train* y *test* debido al elevado tiempo de ejecución que tomaron los experimentos

En las experimentaciones se probaron los siguientes conjuntos de parámetros:

- Constante de regularización $C \in \{0{,}001, 0{,}01, 0{,}05, 0{,}1, 0{,}5, 1, 2, 5\}$.
- $gamma \in \{1e-3, 1e-2, 0{,}1, 0{,}5, 1, 2, 5, 10\}$.

La parametrización de este conjunto de experimentos se recoge en el siguiente documento JSON:

```
{
  "kernel_type": "SVM_RBF",
  "Cvalue": [0.001, 0.01, 0.05, 0.1, 0.5, 1, 2, 5],
  "gamma": [1e-3, 1e-2, 0.1, 0.5, 1, 2, 5, 10]
}
```

Script 4: Contenido del documento JSON "experimentos8_HOG_SVM_kfold_kernel_rbf.json" con el conjunto de parámetros probados en esta experimentación.

Los resultados de esta experimentación se recogen en la siguiente tabla:

C	gamma	CCR	precission	recall	ROC
0.001	0.001	0.917273	0.906561	0.912	0.916833
	0.010	0.928182	0.925253	0.916	0.927167
	0.100	0.949091	0.972340	0.914	0.946167
	0.500	0.817273	0.732504	0.942	0.827667
	1.000	0.899091	0.980247	0.794	0.890333
	2.000	0.770000	0.907591	0.550	0.751667
	5.000	0.478182	0.465549	1.000	0.521667
	10.000	0.454545	0.454545	1.000	0.500000
0.010	0.001	0.917273	0.906561	0.912	0.916833
	0.010	0.928182	0.925253	0.916	0.927167
	0.100	0.949091	0.972340	0.914	0.946167
	0.500	0.818182	0.734375	0.940	0.828333
	1.000	0.897273	0.968523	0.800	0.889167
	2.000	0.771818	0.908197	0.554	0.753667
	5.000	0.478182	0.465549	1.000	0.521667
	10.000	0.454545	0.454545	1.000	0.500000
0.050	0.001	0.917273	0.906561	0.912	0.916833
	0.010	0.928182	0.925253	0.916	0.927167

	0.100	0.949091	0.972340	0.914	0.946167
	0.500	0.817273	0.733959	0.938	0.827333
	1.000	0.897273	0.955294	0.812	0.890167
	2.000	0.771818	0.908197	0.554	0.753667
	5.000	0.478182	0.465549	1.000	0.521667
	10.000	0.454545	0.454545	1.000	0.500000
0.100	0.001	0.917273	0.906561	0.912	0.916833
	0.010	0.930000	0.927273	0.918	0.929000
	0.100	0.949091	0.972340	0.914	0.946167
	0.500	0.817273	0.733959	0.938	0.827333
	1.000	0.898182	0.949074	0.820	0.891667
	2.000	0.771818	0.908197	0.554	0.753667
	5.000	0.478182	0.465549	1.000	0.521667
	10.000	0.454545	0.454545	1.000	0.500000
0.500	0.001	0.918182	0.908367	0.912	0.917667
	0.010	0.960000	0.975000	0.936	0.958000
	0.100	0.949091	0.972340	0.914	0.946167
	0.500	0.817273	0.733959	0.938	0.827333
	1.000	0.897273	0.946882	0.820	0.890833
	2.000	0.770909	0.907895	0.552	0.752667
	5.000	0.478182	0.465549	1.000	0.521667
	10.000	0.454545	0.454545	1.000	0.500000
1.000	0.001	0.922727	0.920892	0.908	0.921500
	0.010	0.970000	0.989518	0.944	0.967833
	0.100	0.957273	0.972860	0.932	0.955167
	0.500	0.855455	0.799649	0.910	0.860000
	1.000	0.897273	0.953162	0.814	0.890333
	2.000	0.776364	0.909677	0.564	0.758667
	5.000	0.478182	0.465549	1.000	0.521667
	10.000	0.454545	0.454545	1.000	0.500000
2.000	0.001	0.941818	0.950413	0.920	0.940000
	0.010	0.973636	0.989605	0.952	0.971833
	0.100	0.955455	0.968815	0.932	0.953500
	0.500	0.640909	0.981651	0.214	0.605333
	1.000	0.655455	0.571767	0.964	0.681167
	2.000	0.479091	0.465983	1.000	0.522500
	5.000	0.548182	1.000000	0.006	0.503000
	10.000	0.547273	1.000000	0.004	0.502000
5.000	0.001	0.959091	0.972973	0.936	0.957167
	0.010	**0.977273**	0.989691	0.960	**0.975833**

0.100	0.955455	0.968815	0.932	0.953500
0.500	0.640909	0.981651	0.214	0.605333
1.000	0.655455	0.571767	0.964	0.681167
2.000	0.479091	0.465983	1.000	0.522500
5.000	0.548182	1.000000	0.006	0.503000
10.000	0.547273	1.000000	0.004	0.502000

Tabla 4: Valores medidos de accuracy, precission, recall y área bajo la curva ROC de los modelos obtenidos considerando todas diferentes combinaciones de parámetros C y gamma.

Como se refleja en los experimentos anteriores, **los modelos considerando C=5 y *gamma*=0.01, permiten obtener los valores más altos de CCR (*Correct Classified Rate*) sobre el conjunto de test**, el valor más alto del área bajo la curva ROC, además de valores de precisión y sensibilidad elevados (sólo es superado por algunos modelos con un valor de CCR más reducido y un valor de ROC próximo a 0.5, lo que lleva a considerar que se trata de modelos infra-ajustados que clasifican todas las instancias como pertenecientes a la clase positiva).

Por su parte, considerando un valor de *gamma*=5, se aprecia una tendencia general en el aumento del rendimiento del modelo al hacer incrementar el valor del parámetro C. Por ello, se considera realizar una segunda experimentación manteniendo fijo el valor de *gamma* y considerando valores mayores para el parámetro C.

Para esta segunda experimentación, se considerará el siguiente conjunto e valores del parámetro C: $C \in \{5{,}5, 6, 7, 8, 10, 20\}$.

Los parámetros de esta experimentación se recogen en el siguiente documento JSON:

```
{
  "kernel_type": "SVM_RBF",
  "Cvalue": [5, 5.5, 6, 7, 8, 10, 20],
  "gamma": 1e-2
}
```

Script 5: Contenido del documento JSON "experimentos9_HOG_SVM_kfold_kernel_rbf.json" con el conjunto de parámetros probados en esta experimentación.

	accuracy	precission	recall	ROC-AUC
C=5	0.97727	0.98969	0.96	0.97583
C=5.5	0.97727	0.98969	0.96	0.97583
C=6	0.97727	0.98969	0.96	0.97583
C=7	0.97727	0.98969	0.96	0.97583
C=8	0.97727	0.98969	0.96	0.97583
C=10	**0.97818**	**0.98971**	**0.962**	**0.97683**
C=20	0.97727	0.98969	0.96	0.97583

Tabla 5: Métricas de accuracy, precission, recall y área bajo la curva ROC del modelo SVM con kernel lineal, gamma=0.01 sobre la partición fija de entrenamiento y test.

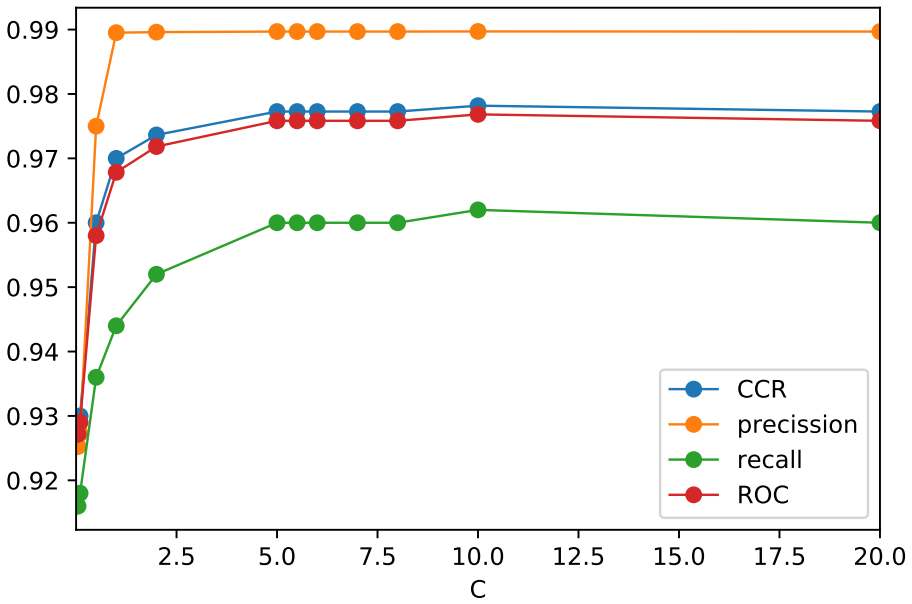

Figura 1: gráfico con la evolución de las métricas CCR, precission, recall y área bajo la curva ROC, al variar el valor del parámetro C manteniendo un valor de gamma=0.01 y considerando un kernel RBF

Para C=10, se obtiene el modelo que ofrece mejor rendimiento de toda la experimentación realizada. Su matriz de confusión, se expone a continuación:

		Clase Real	
		+	-
Clase Predicha	+	481	19
	-	5	595

Tabla 6: Matriz de confusión del modelo con kernel RBF, C=10 y gamma=0.01

Por último, comparamos este modelo con el mejor modelo obtenido considerando el *kernel* lineal:

Parámetros	Tipo de kernel	accuracy	precission	recall	ROC-AUC
C=0.05	lineal	0.96545	0.9753	0.94800	0.96400
C=10, gamma=0.05	RBF	**0.97818**	**0.98971**	**0.962**	**0.97683**

Tabla 7: Comparación de métricas del mejor modelo obtenido con el kernel de base radial (RBF) y el kernel lineal

Se aprecia por consiguiente, que **el modelo que ofrece mejor rendimiento es el modelo con *kernel RBF***.

3.2. Descriptores basados en *Local Binnary Pattern*

A continuación, se repetirá la experimentación analizada en el anterior capitulo para el desarrollo y análisis de modelos haciendo uso de LBP (*Local Binnary Pattern*) como método de extracción de características de las imágenes tratadas. Finalmente, se compararán los resultados obtenidos con los obtenidos con los modelos que que tomaban descriptores HOG.

Primeramente, se realiza una experimentación con modelos SVM de *kernel* lineal considerando el mismo conjunto de valores para la constante de regularización C que en los experimentos anteriores y, nuevamente evaluando como medida de *accuracy*, el CCR medio considerando *kfold* estratificado con 5 particiones.

Se consideran, por tanto, los siguientes valores de C: $C \in \{0{,}001, 0{,}01, 0{,}05, 0{,}1, 0{,}5, 1, 1{,}5, 2, 2{,}5, 5\}$, obteniéndose los siguientes resultados:

C	CCR medio	CCR mínimo	CCR máximo
0.001	0.95452	0.94432	0.96520
0.01	0.95360	0.93968	0.96520
0.05	0.96937	0.96056	0.97912
0.1	0.97123	0.96520	0.97912
0.5	0.98005	0.97448	0.98376
1	0.98329	0.97912	0.98840
1.5	0.98422	0.97679	0.99072
2	0.98470	0.97912	0.99072
2.5	0.98561	0.97912	0.99072
5	**0.98608**	**0.97912**	**0.99072**

Tabla 8: Accuracy medio, mínimo y máximo obtenido mediante una evaluación con kfold estratificado para diferentes valores de C y considerando un kernel de SVM lineal.

El modelo óptimo se obtiene con C=5, sin embargo, se aprecia una tendencia en el crecimiento del CCR medio conforme se incrementa el valor del parámetro C, lo cual lleva a plantear, si para valores de C superiores a 5, el CCR de los modelos se incrementa.

Con esta finalidad, se realiza una segunda experimentación considerando valores superiores para el parámetro C: $C \in \{5, 5{,}5, 6, 7, 8, 10, 20\}$.

Los resultados se recogen en la siguiente tabla:

C	CCR medio	CCR mínimo	CCR máximo
5	**0.98608**	**0.97912**	**0.99072**
5.5	0.98608	0.97912	0.99072
6	0.98608	0.97912	0.99072
7	0.98608	0.97912	0.99072
8	0.98608	0.97912	0.99072
10	0.98608	0.97912	0.99072
20	0.98608	0.97912	0.99072

Tabla 9: Accuracy medio, mínimo y máximo obtenido mediante una evaluación con kfold estratificado para diferentes valores de C superiores a 5 y considerando un kernel de SVM lineal.

Para valores de C superiores a 5, el rendimiento del modelo se muestra invariable con respecto a las métricas obtenidas para C=5. Por consiguiente, se acepta como mejor modelo aquel que considera C=5.

Para terminar con este modelo, se evalúa el rendimiento del modelo entrenado con una partición fija de *train* y *test*:

	accuracy	precission	recall	ROC-AUC
C=5	0.97727	0.97405	0.976	0.97717

Tabla 10: Métricas de accuracy, precission, recall y área bajo la curva ROC del modelo SVM con kernel lineal, C= 0.5 sobre la partición fija de entrenamiento y test.

La matriz de confusión de este modelo es la siguiente:

		Clase Real	
		+	-
Clase Predicha	+	488	12
	-	13	587

Tabla 11: Matriz de confusión del modelo con kernel lineal, C=5

A continuación, se realizó otro conjunto de experimentos con el *kernel* RBF, ejecutando la misma experimentación que en el capítulo anterior. Todos los modelos se desarrollaron y evaluaron usando una partición fija de entrenamiento y test, y se trabajó con el siguiente conjunto de parámetros:

- Constante de regularización $C \in \{0{,}001, 0{,}01, 0{,}05, 0{,}1, 0{,}5, 1, 2, 5\}$.
- $gamma \in \{1e-3, 1e-2, 0{,}1, 0{,}5, 1, 2, 5, 10\}$.

Los resultados obtenidos se recogen en la siguiente tabla:

C	gamma	CCR	precission	recall	ROC
0.001	0.001	0.950000	0.937132	0.954	0.950333
	0.010	0.950000	0.937132	0.954	0.950333
	0.100	0.952727	0.937500	0.960	0.953333
	0.500	0.960000	0.961538	0.950	0.959167
	1.000	0.936364	0.963362	0.894	0.932833
	2.000	0.912727	0.976415	0.828	0.905667
	5.000	0.854545	0.991329	0.686	0.840500
	10.000	0.816364	0.983766	0.606	0.798833

0.010	0.001	0.950000	0.937132	0.954	0.950333
	0.010	0.950000	0.937132	0.954	0.950333
	0.100	0.952727	0.937500	0.960	0.953333
	0.500	0.960000	0.961538	0.950	0.959167
	1.000	0.936364	0.963362	0.894	0.932833
	2.000	0.912727	0.976415	0.828	0.905667
	5.000	0.854545	0.991329	0.686	0.840500
	10.000	0.816364	0.983766	0.606	0.798833
0.050	0.001	0.950000	0.937132	0.954	0.950333
	0.010	0.950000	0.937132	0.954	0.950333
	0.100	0.952727	0.937500	0.960	0.953333
	0.500	0.958182	0.957661	0.950	0.957500
	1.000	0.936364	0.963362	0.894	0.932833
	2.000	0.912727	0.976415	0.828	0.905667
	5.000	0.854545	0.991329	0.686	0.840500
	10.000	0.813636	0.983607	0.600	0.795833
0.100	0.001	0.950000	0.937132	0.954	0.950333
	0.010	0.950000	0.937132	0.954	0.950333
	0.100	0.960909	0.950690	0.964	0.961167
	0.500	0.964545	0.971370	0.950	0.963333
	1.000	0.936364	0.963362	0.894	0.932833
	2.000	0.912727	0.976415	0.828	0.905667
	5.000	0.854545	0.991329	0.686	0.840500
	10.000	0.813636	0.983607	0.600	0.795833
0.500	0.001	0.950000	0.937132	0.954	0.950333
	0.010	0.948182	0.936884	0.950	0.948333
	0.100	0.976364	0.977823	0.970	0.975833
	0.500	0.978182	0.987705	0.964	0.977000
	1.000	0.960000	0.976987	0.934	0.957833
	2.000	0.912727	0.976415	0.828	0.905667
	5.000	0.854545	0.991329	0.686	0.840500
	10.000	0.813636	0.983607	0.600	0.795833
1.000	0.001	0.950000	0.937132	0.954	0.950333
	0.010	0.955455	0.951904	0.950	0.955000
	0.100	0.980000	0.981855	0.974	0.979500
	0.500	0.983636	0.987854	0.976	0.983000
	1.000	0.969091	0.987448	0.944	0.967000
	2.000	0.934545	0.961207	0.892	0.931000
	5.000	0.909091	0.973934	0.822	0.901833

	10.000	0.872727	0.978723	0.736	0.861333
2.000	0.001	0.949091	0.937008	0.952	0.949333
	0.010	0.964545	0.958250	0.964	0.964500
	0.100	0.983636	0.985887	0.978	0.983167
	0.500	**0.988182**	**0.995927**	0.978	**0.987333**
	1.000	0.972727	0.989583	0.950	0.970833
	2.000	0.932727	0.961039	0.888	0.929000
	5.000	0.902727	0.982801	0.800	0.894167
	10.000	0.710909	0.613184	0.986	0.733833
5.000	0.001	0.947273	0.936759	0.948	0.947333
	0.010	0.973636	0.970060	0.972	0.973500
	0.100	0.985455	0.987903	0.980	0.985000
	0.500	0.987273	0.993902	0.978	0.986500
	1.000	0.971818	0.989562	0.948	0.969833
	2.000	0.932727	0.961039	0.888	0.929000
	5.000	0.902727	0.982801	0.800	0.894167
	10.000	0.710909	0.613184	0.986	0.733833

Tabla 12: Valores medidos de accuracy, precission, recall y área bajo la curva ROC de los modelos obtenidos considerando todas diferentes combinaciones de parámetros C y gamma.

El modelo que ofrece mejor rendimiento global es quel que considera C=2 y *gamma*=0.5.

Su matriz de confusión es la siguiente:

		Clase Real	
		+	-
Clase Predicha	+	489	11
	-	2	598

Tabla 13: Matriz de confusión del modelo con kernel RBF, C=2 y gamma=0.5

Finalmente comparamos este modelo con el mejor modelo obtenido considerando el *kernel* lineal:

Parámetros	Tipo de kernel	accuracy	precission	recall	ROC-AUC
C=5	lineal	0.98608	0.97912	0.99072	
C=2, gamma=0.5	RBF	0.988182	0.995927	0.978	0.987333

Tabla 14: Comparación de métricas del mejor modelo obtenido con el kernel de base radial (RBF) y el kernel lineal

Nuevamente el *kernel* RBF supera en rendimiento al *kernel* lineal.

Por último, comparamos estos resultados con los obtenidos usando descriptores basados en HOG:
En primer lugar, para las experimentaciones basadas en *kernel* lineal y comparando los CCR medios:

C	descriptores HOG	descriptores LBP
0.001	0.93550	0.95452
0.01	0.97262	0.95360
0.05	0.97541	0.96937
0.1	0.97262	0.97123
0.5	0.96613	0.98005
1	0.96613	0.98329
1.5	0.96613	0.98422
2	0.96613	0.98470
2.5	0.96613	0.98561
5	0.96613	0.98608

Tabla 15: Comparativa entre los CCR medios de los modelos SVM basados en descriptores HOG y descriptores LBP, considerando para ambos casos un kernel lineal y el mismo conjunto de parámetros C.

Considerando la misma parametrización en los modelos SVM, **se aprecia que los modelos basados en LBP, ofrecen, generalmente, los mejores modelos óptimos**.

Por último, para el caso del *kernel* RBF, comparamos en la siguiente tabla los CCR obtenidos:

C	gamma	descriptores HOG	descriptores LBP
0.001	0.001	0.917273	0.950000
	0.010	0.928182	0.950000
	0.100	0.949091	0.952727
	0.500	0.817273	0.96
	1.000	0.899091	0.936364

	2.000	0.770000	0.912727
	5.000	0.478182	0.952727
	10.000	0.454545	0.816364
0.010	0.001	0.917273	0.95
	0.010	0.928182	0.95
	0.100	0.949091	0.952727
	0.500	0.818182	0.96
	1.000	0.897273	0.936364
	2.000	0.771818	0.912727
	5.000	0.478182	0.854545
	10.000	0.454545	0.816364
0.050	0.001	0.917273	0.95
	0.010	0.928182	0.95
	0.100	0.949091	0.952727
	0.500	0.817273	0.958182
	1.000	0.897273	0.936364
	2.000	0.771818	0.912727
	5.000	0.478182	0.854545
	10.000	0.454545	0.813636
0.100	0.001	0.917273	0.95
	0.010	0.930000	0.95
	0.100	0.949091	0.960909
	0.500	0.817273	0.964545
	1.000	0.898182	0.936364
	2.000	0.771818	0.912727
	5.000	0.478182	0.854545
	10.000	0.454545	0.813636
0.500	0.001	0.918182	0.95
	0.010	0.960000	0.948182
	0.100	0.949091	0.976364
	0.500	0.817273	0.978182
	1.000	0.897273	0.96
	2.000	0.770909	0.912727
	5.000	0.478182	0.854545
	10.000	0.454545	0.813636
1.000	0.001	0.922727	0.95
	0.010	0.970000	0.955455
	0.100	0.957273	0.98
	0.500	0.855455	0.983636
	1.000	0.897273	0.969091

	2.000	0.776364	0.934545
	5.000	0.478182	0.909091
	10.000	0.454545	0.872727
2.000	0.001	0.941818	0.949091
	0.010	0.973636	0.964545
	0.100	0.955455	0.983636
	0.500	0.640909	0.988182
	1.000	0.655455	0.972727
	2.000	0.479091	0.932727
	5.000	0.548182	0.902727
	10.000	0.547273	0.710909
5.000	0.001	0.959091	0.947273
	0.010	0.977273	0.973636
	0.100	0.955455	0.985455
	0.500	0.640909	0.987273
	1.000	0.655455	0.971818
	2.000	0.479091	0.932727
	5.000	0.548182	0.902727
	10.000	0.547273	0.710909

Tabla 16: Comparación de CCR de los modelos basados en descriptores HOG y los modelos basados en descriptores LBP

En los anteriores resultados, se aprecia de forma clara que, en la inmensa mayoría de las configuraciones, los modelos que hace uso de descriptores LBP supera en rendimiento al modelo que hace uso de descriptores HOG.

Se ha podido comprobar por tanto, que de forma general, **la extracción de rasgos mediante descriptores LBP permite obtener modelos con mejor rendimiento que haciendo uso de descriptores HOG**.

3.3. Descriptores basados en *Local Binnary Pattern* Uniforme

Se repite la misma experimentación, usando descriptores basados en LBP Uniforme.

Para el *kernel*, nuevamente se elaboran modelos para diferentes valores de la constante de regularización C y se evalúa el rendimiento de dichos modelos mediante *kfold* estratificado con 5 particiones.

Nuevamente, el conjunto de valores de C a probar es el siguiente:
$C \in \{0{,}001, 0{,}01, 0{,}05, 0{,}1, 0{,}5, 1, 1{,}5, 2, 2{,}5, 5\}$.

Los resultados de esta experimentación se recogen a continuación:

C	CCR medio	CCR mínimo	CCR máximo
0.001	0.94060	0.91879	0.95128
0.01	0.95035	0.93735	0.95824
0.05	0.96473	0.95591	0.97448
0.1	0.97077	0.96288	0.97912
0.5	0.97540	0.96984	0.98376
1	0.97958	0.97448	0.98608
1.5	0.98422	0.97912	0.98840
2	0.98470	0.97680	0.99072
2.5	**0.98422**	**0.97912**	**0.98840**
5	0.98329	0.97680	0.98840

Tabla 17: Accuracy medio, mínimo y máximo obtenido mediante una evaluación con kfold estratificado para diferentes valores de C y considerando un kernel de SVM lineal.

El modelo que presenta un mayor rendimiento es el que considera C=2.5.

Para acabar, se evalúa el modelo ante una partición única de entrenamiento y test, obteniendo los siguientes resultados:

	accuracy	precission	recall	ROC-AUC
C=2.5	0.97818	0.976	0.976	0.978

Tabla 18: Métricas de accuracy, precission, recall y área bajo la curva ROC del modelo SVM con kernel lineal, C= 0.25 sobre la partición fija de entrenamiento y test.

Su matriz de confusión es la siguiente:

		Clase Real	
		+	-
Clase Predicha	+	488	12
	-	12	588

Tabla 19: Matriz de confusión del modelo con kernel lineal, C=2.5

Por último, para el *kernel* RBF, se repite nuevamente las experimentaciones llevadas a cabo en los anteriores capítulos: Se elaborarán diferentes modelos considerando diferentes valores para los parámetros C y *gamma* y se evaluará el rendimiento de estos modelos considerando una partición fija de *train* y *test*.

El conjunto de valores para los parámetros C y *gamma* que se van a probar se recoge a continuación:

- Constante de regularización $C \in \{0{,}001, 0{,}01, 0{,}05, 0{,}1, 0{,}5, 1, 2, 5\}$.
- $gamma \in \{1e-3, 1e-2, 0{,}1, 0{,}5, 1, 2, 5, 10\}$.

Los resultados de los experimentos se muestran a continuación:

C	gamma	CCR	precission	recall	ROC
0.001	0.001	0.939091	0.922027	0.946	0.939667
	0.010	0.940000	0.922179	0.948	0.940667
	0.100	0.942727	0.932673	0.942	0.942667
	0.500	0.943636	0.946939	0.928	0.942333
	1.000	0.926364	0.956427	0.878	0.922333
	2.000	0.910000	0.971765	0.826	0.903000
	5.000	0.851818	**0.997050**	0.676	0.837167
	10.000	0.816364	0.980645	0.608	0.799000
0.010	0.001	0.939091	0.922027	0.946	0.939667
	0.010	0.940000	0.922179	0.948	0.940667
	0.100	0.942727	0.932673	0.942	0.942667
	0.500	0.943636	0.946939	0.928	0.942333
	1.000	0.926364	0.956427	0.878	0.922333
	2.000	0.910000	0.971765	0.826	0.903000
	5.000	0.851818	**0.997050**	0.676	0.837167
	10.000	0.818182	0.983871	0.610	0.800833
0.050	0.001	0.939091	0.922027	0.946	0.939667
	0.010	0.940000	0.922179	0.948	0.940667
	0.100	0.944545	0.932939	0.946	0.944667
	0.500	0.946364	0.947262	0.934	0.945333
	1.000	0.926364	0.956427	0.878	0.922333
	2.000	0.910000	0.971765	0.826	0.903000
	5.000	0.851818	**0.997050**	0.676	0.837167
	10.000	0.818182	0.983871	0.610	0.800833
0.100	0.001	0.939091	0.922027	0.946	0.939667
	0.010	0.940000	0.922179	0.948	0.940667

	0.100	0.954545	0.942913	0.958	0.954833
	0.500	0.954545	0.957317	0.942	0.953500
	1.000	0.925455	0.954348	0.878	0.921500
	2.000	0.910000	0.971765	0.826	0.903000
	5.000	0.851818	**0.997050**	0.676	0.837167
	10.000	0.818182	0.983871	0.610	0.800833
0.500	0.001	0.939091	0.922027	0.946	0.939667
	0.010	0.945455	0.929688	0.952	0.946000
	0.100	0.970000	0.966068	0.968	0.969833
	0.500	0.979091	0.985743	0.968	0.978167
	1.000	0.960000	0.973029	0.938	0.958167
	2.000	0.910000	0.971765	0.826	0.903000
	5.000	0.851818	**0.997050**	0.676	0.837167
	10.000	0.817273	0.980707	0.610	0.800000
1.000	0.001	0.939091	0.922027	0.946	0.939667
	0.010	0.950909	0.935547	0.958	0.951500
	0.100	0.979091	0.977956	0.976	0.978833
	0.500	0.983636	0.987854	0.976	0.983000
	1.000	0.968182	0.983368	0.946	0.966333
	2.000	0.932727	0.963043	0.886	0.928833
	5.000	0.915455	0.988010	0.824	0.907833
	10.000	0.870909	0.978610	0.732	0.859333
2.000	0.001	0.939091	0.920388	0.948	0.939833
	0.010	0.960000	0.957831	0.954	0.959500
	0.100	0.981818	0.983871	0.976	0.981333
	0.500	0.986364	0.993890	0.976	0.985500
	1.000	0.972727	0.989583	0.950	0.970833
	2.000	0.930909	0.964912	0.880	0.926667
	5.000	0.910909	0.990244	0.812	0.902667
	10.000	0.629091	0.551339	**0.988**	0.659000
5.000	0.001	0.945455	0.929688	0.952	0.946000
	0.010	0.971818	0.968064	0.970	0.971667
	0.100	0.984545	0.989858	0.976	0.983833
	0.500	**0.987273**	0.993902	0.978	**0.986500**
	1.000	0.972727	0.989583	0.950	0.970833
	2.000	0.930909	0.964912	0.880	0.926667
	5.000	0.910909	0.990244	0.812	0.902667
	10.000	0.629091	0.551339	**0.988**	0.659000

Del anterior conjunto de experimentos, se observa que **el modelo con C=5 y *gamma*=0.5 ofrece valores óptimos en CCR y área bajo**

la curva ROC. Si bien existen algunos modelos con valores más altos de *precission* y *recall*, sus valores de más bajos de CCR y AUC-ROC, nos llevan a considerar que se trata de modelos con poca capacidad predictiva hacia la clase negativa y con mayor tendencia a clasificar patrones en la clase positiva.

El modelo óptimo se obtiene con C=5 y *gamma*=0.5. No obstante, manteniendo *gamma*=0.5, se observa una tendencia en el incremento del CCR al aumentar el valor de C, por lo que se plantea realizar una segunda experimentación manteniendo fijo el valor de *gamma* y considerando valores superiores para C.

Se considera utilizar el siguiente conjunto de parámetros:

- $C \in \{5{,}5, 6, 6{,}5, 7, 8, 10, 20\}$
- *gamma*=0.5

Se obtuvieron los siguientes resultados:

C	CCR	precission	recall	ROC
5.5	0.98727	0.99390	0.978	0.9865
6	0.98727	0.99390	0.978	0.9865
6.5	0.98727	0.99390	0.978	0.9865
7	0.98727	0.99390	0.978	0.9865
8	0.98727	0.99390	0.978	0.9865
10	0.98727	0.99390	0.978	0.9865
20	0.98727	0.99390	0.978	0.9865

Tabla 21: Accuracy medio, mínimo y máximo obtenido mediante una evaluación con kfold estratificado para diferentes valores de C superiores a 5 y considerando un kernel de SVM lineal.

Al incrementar el valor de C, los modelos obtenidos ofrecen rendimientos similares al modelo con C=5. Por lo tanto, **se tomará como modelo óptimo el obtenido para C=5 y *gamma*=0.5**.

Comparamos este mejor modelo obtenido con el mejor modelo obtenido con el *kernel* lineal:

Parámetros	Tipo de kernel	accuracy	precission	recall	ROC-AUC
C=2.5	lineal	0.97818	0.976	0.976	0.978
C=5, gamma=0.5	RBF	0.987273	0.993902	0.978	0.986500

Tabla 22: Comparación de métricas del mejor modelo obtenido con el kernel de base radial (RBF) y el kernel lineal

El modelo con *kernel* RBF muestra un rendimiento superior en todas las métricas respecto del modelo con *kernel* lineal.

Por último, comparamos el rendimiento de estos modelos con los modelos basados en descriptores LBP básico:
Para el *kernel* lineal, comparamos los CCR medios obtenidos en las diferentes experimentaciones:

C	descriptores LBP	descriptores LBP Uniforme
0.001	0.95452	0.94060
0.01	0.95360	0.95035
0.05	0.96937	0.96473
0.1	0.97123	0.97077
0.5	0.98005	0.97540
1	0.98329	0.97958
1.5	0.98422	0.98422
2	0.98470	0.98470
2.5	0.98561	0.98422
5	0.98608	0.98329

Tabla 23: Comparativa entre los CCR medios de los modelos SVM basados en descriptores LBP y descriptores LBP uniformes, considerando para ambos casos un kernel lineal y el mismo conjunto de parámetros C.

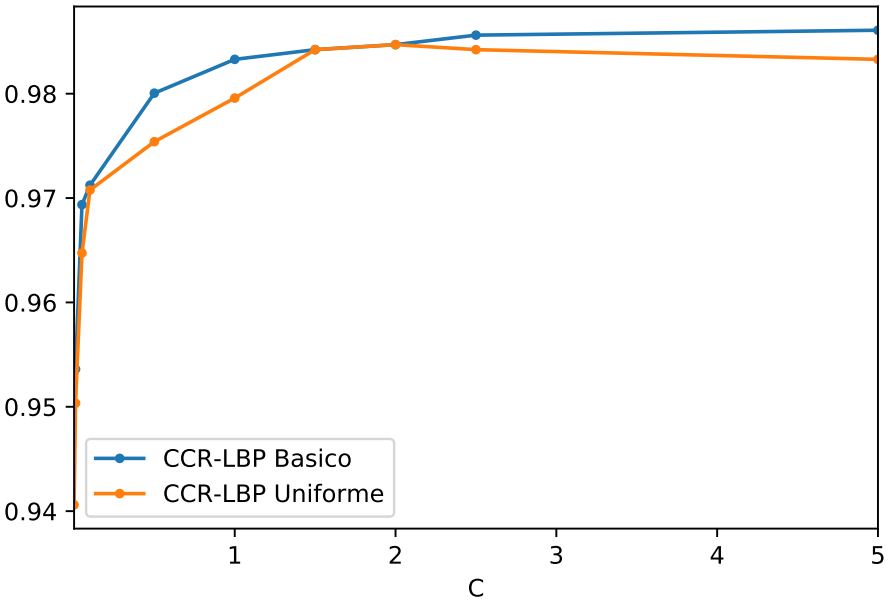

Figura 2: Gráfico con la variacion de los CCR medios de los modelos basados en descriptores LBP y de los modelos basados en LBP Uniforme al variar el valor de C

Se aprecia claramente que **el modelo basado en descriptores LBP básico ofrece, de forma general, un rendimiento algo superior a los modelos basados en LBP Uniforme**.

Por su parte, para el *kernel* RBF, comparamos los CCR de las pruebas ejecutadas para ambos tipos de modelos:

C	gamma	descriptores LBP	descriptores LBP Uniforme
0.001	0.001	0.950000	0.939091
	0.010	0.950000	0.940000
	0.100	0.952727	0.942727
	0.500	0.96	0.943636
	1.000	0.936364	0.926364
	2.000	0.912727	0.91
	5.000	0.952727	0.851818
	10.000	0.816364	0.816364
0.010	0.001	0.95	0.939091
	0.010	0.95	0.940000

	0.100	0.952727	0.942727
	0.500	0.96	0.943636
	1.000	0.936364	0.926364
	2.000	0.912727	0.91
	5.000	0.854545	0.851818
	10.000	0.816364	0.818182
0.050	0.001	0.95	0.939091
	0.010	0.95	0.94
	0.100	0.952727	0.944545
	0.500	0.958182	0.946364
	1.000	0.936364	0.926364
	2.000	0.912727	0.91
	5.000	0.854545	0.851818
	10.000	0.813636	0.818182
0.100	0.001	0.95	0.939091
	0.010	0.95	0.94
	0.100	0.960909	0.954545
	0.500	0.964545	0.954545
	1.000	0.936364	0.925455
	2.000	0.912727	0.91
	5.000	0.854545	0.851818
	10.000	0.813636	0.818182
0.500	0.001	0.918182	0.939091
	0.010	0.960000	0.945455
	0.100	0.949091	0.97
	0.500	0.817273	0.979091
	1.000	0.897273	0.96
	2.000	0.770909	0.91
	5.000	0.478182	0.851818
	10.000	0.454545	0.817273
1.000	0.001	0.922727	0.939091
	0.010	0.970000	0.950909
	0.100	0.957273	0.979091
	0.500	0.855455	0.983636
	1.000	0.897273	0.968182
	2.000	0.776364	0.932727
	5.000	0.478182	0.915455
	10.000	0.454545	0.870909
2.000	0.001	0.941818	0.939091
	0.010	0.973636	0.96

	0.100	0.955455	0.981818
	0.500	0.640909	0.986364
	1.000	0.655455	0.972727
	2.000	0.479091	0.930909
	5.000	0.548182	0.910909
	10.000	0.547273	0.629091
5.000	0.001	0.959091	0.945455
	0.010	0.977273	0.971818
	0.100	0.955455	0.984545
	0.500	0.640909	0.987273
	1.000	0.655455	0.972727
	2.000	0.479091	0.930909
	5.000	0.548182	0.910909
	10.000	0.547273	0.629091

Tabla 24: Comparación del CCR de los modelos basados en LBP básico y LBP Uniforme ante diferentes valores de C y gamma.

En la anterior comparación, se observa que, de forma general, **el rendimiento de los modelos basados en LBP supera al de los modelos basados en LBP Uniforme** aunque, para valores de C iguales o superiores a 1, aparecen muchos modelos basados en LBP Uniforme que superan el rendimiento de los modelos LBP básicos. No obstante, **el mejor modelo obtenido en la experimentación con descriptores LBP básico supera al mejor modelo obtenido con descriptores LBP Uniforme**.

4. Test del mejor modelo

Para finalizar, se desea realizar una demostración de la capacidad de predicción del mejor modelo obtenido, realizando una clasificación de imágenes del conjunto de test seleccionadas aleatoriamente.

El mejor modelo que se obtuvo durante la experimentación es el siguiente:

- Descriptores basados en LBP.
- *Kernel* SVM de base radial (RBF).
- C=2.

- $gamma$=0.5.

El resultado de la clasificación realizada sobre el conjunto de imágenes se muestra a continuación sobre cada imagen:

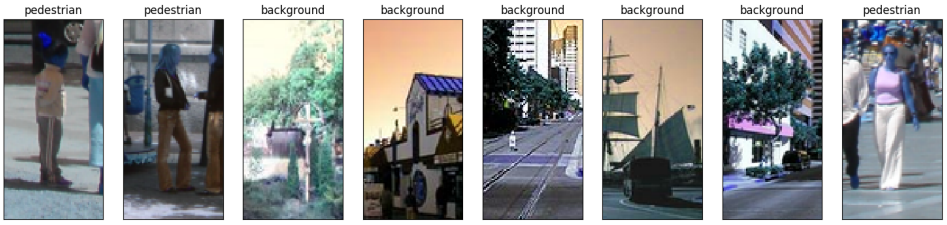

Figura 3: Ejemplo de clasificación del modelo SVM