PRÁCTICA EXTRACCIÓN DE CARACTERÍSTICAS EN IMÁGENES

Máster DATCOM

Extracción de Características en Imágenes

Autor
Alberto Armijo Ruiz

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

24 de febrero de 2019

0.1. Resumen

En esta práctica se realizará un estudio sobre diferentes tipos de descriptores de imágenes; en concreto HOG (Histogram of Gradients) y LBP (Local Binary Patterns). De este último se realizará una modificación, para cuál también se realizará un estudio con diferentes clasificadores. Por último, se describirá el proceso para crear un detector de personas en imágenes con uno de los clasificadores generados en los apartados anteriores y se analizarán los resultados que produce.

Esta práctica ha sido realizada en Python 3 junto con OpenCV. Dentro del fichero comprimido se encontrarán los diferentes archivos con la implementación de los descriptores, los archivos de pruebas, el archivo con los resultados, y el archivo que contiene la implementación del detector de personas. Adicionalmente se proporcionan archivos con los datos generados por los descriptores comprimidos ya que tardan bastante en cargarse de forma normal por si se desea ejecutar las pruebas con los diferentes descriptores.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Evaluación del descriptor HOG

Para esta primera parte de la práctica se realizará un análisis de los resultados obtenidos con el descriptor HOG (Histogram of Gradients), para ello, se definirán diferentes funciones, tanto para entrenar el modelo y obtener resultados como para hacer validación cruzada. El contenido que se va a explicar a continuación se encuentra dentro de los archivos **Extracción de rasgos.py** y **pruebas_hog.py**.

1.1. Lectura de imágenes,creación del modelo y predicción con nuevas imágenes

Lo primero que se debe hacer es leer los datos, para ello se ha creado la función *loadTrainingData()*, esta función se encarga de abrir cada una de las imágenes contenidas en la carpeta de train que se proporciona con la práctica de ejemplo *ECI.Practica*; por cada una de la imágenes se computa se descriptor HOG; para ello se hace uso de la función *cv2.HOGDescriptor().compute()*.

Para esta práctica se está utilizando un descriptor HOG con parámetros por defecto; este descriptor por defecto utiliza un tamaño de ventana de 128x64, bloques de 16x16, desplazamientos de 8x8, ... Con estos parámetros obtenemos por cada ventana 3780 características. Las imágenes que se utilizan en esta práctica son de 128x64, por lo que al utilizar el descriptor sobre estas cada una producirá un vector con 3780 características. Además de computar el descriptor por cada una de las imágenes, se añaden también a un vector la clase a la que pertenece cada imagen; dicho vector contiene unos (imágenes con personas en las fotos) y ceros (imágenes con fondo, sin personas). Una vez se han generado todas las imágenes se devuelve una matriz que contiene todos los descriptores calculados y un vector con la clase de cada imagen.

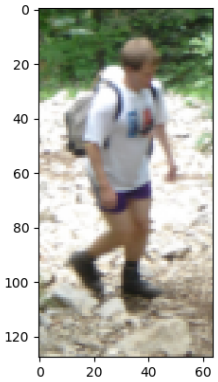
Tras esto, se debe entrenar un modelo para después poder predecir clases de una imagen dada; para ello, se utilizará como clasificador un *SVM* contenido en la librería de *OpenCV*. Este clasificador permite utilizar diferentes tipos de "kernels" que se utilizan

después para realizar transformaciones a los datos y conseguir una mejor separación de estos; por el momento se va a utilizar un kernel lineal, más tarde se realizarán pruebas con diferentes kernels y se analizarán los resultados obtenidos.

Para utilizar este clasificador es necesario utilizar la función *cv.ml.SVM_create()*, esta función nos genera un *SVM* vacío al que le podemos aplicar los parámetros que queramos, por defecto se definirá el tipo de kernel (lineal) y el tipo de SVM (en este caso de clasificación); también se le pueden (o deben) añadir más parámetros dependiendo del tipo de kernel que se utilice. Una vez definidos dichos parámetros del *SVM* se utiliza la función *train()* a la que se le pasa la matriz de descriptores, el vector de clases de las imágenes y además se le añade un parámetro indicando si las filas son los ejemplos o son las columnas (en el caso de esta práctica son las filas las que contienen cada uno de los ejemplos). Todo esto está contenido dentro de la función *train()*, definida dentro de la práctica, a la cual se le pasa la matriz de descriptores, el vector de clases, el tipo de kernel que utilizará el clasificador, y un parámetro llamado "degree" utilizado por algunos kernels.

Una vez se ha entrenado el modelo, podemos utilizarlo para predecir la clase de una imagen. Para ello se ha creado la función *test()*, a la cual se le pasa una imagen y el clasificador que queremos utilizar. Para la imagen dada se computa su descriptor HOG; tras esto, se utiliza la función *clasficador.predict()* para obtener la predicción; este método nos devuelve una tupla, de la cual nos interesa el segundo valor de esta, en la cual se encuentra un vector con la clase predicha para la imagen; si se utilizara la función *clasificador.predict()* sobre varios descriptores, este vector contendría la clase predicha para cada uno de los descriptores.

El proceso de crear leer las imágenes, crear un modelo y realizar una prueba con una imagen esta contenido dentro de la función *ejemploClasificadorImagenes()* dentro de este ejemplo se utiliza como test una imagen de una persona, por lo que el resultado de la predicción debe de ser uno. A continuación se mostrará la imagen utilizada y una foto con la salida de la función.




```
Leidas 1916 imágenes de entrenamiento -> positivas
Leidas 2390 imágenes de entrenamiento -> negativas
Clasificador entrenado
Predicción: (0.0, array([[1.]], dtype=float32))
Leidas 500 imágenes de entrenamiento -> positivas
Leidas 600 imágenes de entrenamiento -> negativas
```

Figura 1.1: Ejemplo clasificación con SVM lineal

1.2. Cálculo de medidas de interés para HOG

En este apartado, se describirá el proceso realizado para obtener medidas de interés sobre el clasificador, para ello se cargará un conjunto de imágenes de test y se creará una función para calcular diferentes medidas de interés.

Para cargar el conjunto de imágenes de test se realiza el mismo proceso que en el apartado anterior, pero en vez de computar el descriptor para estas imágenes se guardan las imágenes; esto se realiza en la función *loadTestData()*. Tras esto se utiliza la función *test()* sobre cada una de las imágenes y se obtienen las predicciones del clasificador. Una vez se han obtenido las predicciones estás se guardan en un vector y se utiliza la función *calculateMetrics()* para obtener las diferentes medidas.

La función *calculateMetrics()* se encarga de obtener medidas de interés para los datos que se predicen, dichas medidas son *F1-score*, *Accuracy*, *Precision*, *True Negative Rate*, *True Positive Rate*.A esta función se le debe pasar la predicción hecha por el clasificador y la clase real de los datos. Dichas medidas miden lo siguiente:

- **Accuracy:** mide el porcentaje de predicciones correctas sobre el total de predicciones realizadas.
- **Precision:** mide el porcentaje de predicciones positivas reales sobre el total de predicciones positivas realizadas, es decir, el porcentaje de imágenes que se han

clasificado como positivas que realmente son positivas.

- **True Positive Rate:** mide el porcentaje de predicciones positivas hechas sobre el total de datos positivos que hay, es decir, el porcentaje de imágenes positivas que se han clasificado como positivas.
- **True Negative Rate:** mide el porcentaje de predicciones negativas hechas sobre el total de datos negativos que hay, es decir, el porcentaje de imágenes negativas que se han clasificado como negativas.
- **F1-score:** está medida calcula una proporción entre *Precision* y *True Positive Rate*, indica la calidad de la predicción de la clase positiva, en nuestro caso sería detectar a una persona en la imagen. Esta medida decrece sin cualquiera de las dos medidas anteriores decrece.

Estas medidas nos serán útiles para conocer mejor como se está ajustando el clasificador entrenado con los datos. Los resultados obtenidos para nuestro conjunto de test son los siguientes.

Figura 1.2: Resultados de test para HOG

Como se puede ver, los resultados son bastante buenos, obteniendo un 96 % aproximado de acierto, como se puede ver en el resto de medidas, el clasificador ha obtenido mejores resultados clasificando imágenes como fondo que como persona, aunque realmente no hay mucha diferencia.

1.3. Pruebas con HOG

En este apartado se compararán diferentes clasificadores a través de las medidas de interés descritas en el apartado anterior. Para este apartado se han utilizado todas las imágenes, en total 5406, y se ha utilizado validación cruzada para crear diferentes conjuntos de validación, de esta forma se pueden ver resultados más generales que no dependan solamente de los resultados de un único conjunto de validación y estar más seguros de la calidad del clasificador.

Para realizar validación cruzada se ha creado la función *crossValidation()* esta función tiene como parámetros la matriz de datos con los descriptores, el vector de clases de dichos casos y el número de validaciones a realizar, por defecto este último parámetro es 5, el cual es el número de validaciones que se han utilizado para realizar el estudio de los diferentes clasificadores. También tiene parámetros adicionales para poder definir las características del clasificador. Dentro de esta función se realiza el siguiente proceso por cada validación.

1. Se crea un conjunto de datos de validación. Para ello se utiliza la función de la librería *sklearn.train_test_split()*; dicha función nos divide el conjunto de datos en un conjunto de datos de train y otro de test; para ello elige de forma aleatoria cada vez un conjunto de datos para test y forma el conjunto de train sin incluir dichos datos. Para el estudio se ha utilizado un tamaño de train del 80 % de los datos y un 20 % para test.

2. Se crea un modelo y se entrena con el conjunto de datos de test.

3. Se obtienen las predicciones de este modelo para los datos de test, se calculan la medidas anteriormente descritas y se guardan.

4. Se actualiza el mejor modelo obtenido hasta el modelo, para ello se utiliza la medida *Accuracy*; esto nos será útil para guardar el mejor modelo encontrado para la validación y utilizarlo más tarde si es necesario.

Ahora, pasaremos a comentar los resultados obtenidos por el descriptor HOG con diferentes kernels de SVM y variando algunos parámetros. Los clasificadores que se han entrenado son los siguientes: SVM con kernel lineal, SVM con kernel radial (RBF), SVM con kernel polinómico con grados 2, 3, 4 y 5, SVM con kernel sigmoidal, SVM con kernel Chi Cuadrado y SVM con kernel Inter. Para cada uno de los modelo se han realizado 5 validaciones. Los resultados son los siguientes:

Validation	**F1**	**accuracy**	**precision**	**truenegativerate**	**truepositiverate**
0	0.955533	0.960259	0.966527	0.973019	0.944785
1	0.960566	0.963956	0.979381	0.982699	0.942460
2	0.959474	0.965804	0.962637	0.972756	0.956332
3	0.953751	0.958410	0.960663	0.967905	0.946939
4	0.970854	0.973198	0.979716	0.982759	0.962151

Cuadro 1.1: Validación con SVM kernel lineal

Validation	**F1**	**accuracy**	**precision**	**truenegativerate**	**truepositiverate**
0	0.853179	0.882625	0.960938	0.975042	0.767152
1	0.834146	0.874307	0.957983	0.975767	0.738661
2	0.850123	0.887246	0.971910	0.983974	0.755459
3	0.854237	0.880776	0.952141	0.968013	0.774590
4	0.865106	0.888170	0.948655	0.964646	0.795082

Cuadro 1.2: Validación con SVM kernel RBF

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.963441	0.968577	0.976035	0.981997	0.951168
1	0.969008	0.972274	0.985294	0.988136	0.953252
2	0.973601	0.976895	0.976695	0.981878	0.970526
3	0.974039	0.976895	0.979123	0.983278	0.969008
4	0.968442	0.970425	0.983968	0.985891	0.953398

Cuadro 1.3: Validación con SVM kernel polinómico grado 2

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.978238	0.980591	0.989518	0.991582	0.967213
1	0.979955	0.983364	0.988764	0.992051	0.971302
2	0.968153	0.972274	0.976445	0.981878	0.960000
3	0.973577	0.975970	0.979550	0.982964	0.967677
4	0.974722	0.976895	0.993814	0.994810	0.956349

Cuadro 1.4: Validación con SVM kernel polinómico grado 3

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.970684	0.975046	0.984581	0.988618	0.957173
1	0.969325	0.972274	0.983402	0.986348	0.955645
2	0.981289	0.983364	0.995781	0.996633	0.967213
3	0.973188	0.975046	1.000000	1.000000	0.947776
4	0.972860	0.975970	0.991489	0.993266	0.954918

Cuadro 1.5: Validación con SVM kernel polinómico grado 4

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.970982	0.975970	0.981941	0.987281	0.960265
1	0.973568	0.977819	0.988814	0.991948	0.958785
2	0.971904	0.975046	0.991507	0.993243	0.953061
3	0.972399	0.975970	0.982833	0.986799	0.962185
4	0.966135	0.968577	0.981781	0.984266	0.950980

Cuadro 1.6: Validación con SVM kernel polinómico grado 5

Validation	**F1**	**accuracy**	**precision**	**truenegativerate**	**truepositiverate**
0	0.530194	0.575786	0.531828	0.614865	0.528571
1	0.531154	0.575786	0.521042	0.602990	0.541667
2	0.617252	0.446396	0.446396	0.000000	1.000000
3	0.535865	0.593346	0.541578	0.643449	0.530271
4	0.520686	0.560998	0.503906	0.578773	0.538622

Cuadro 1.7: Validación con SVM kernel sigmoidal

Validation	**F1**	**accuracy**	**precision**	**truenegativerate**	**truepositiverate**
0	0.698479	0.597043	0.539530	0.246503	0.990196
1	0.651129	0.557301	0.489059	0.250401	0.973856
2	0.661839	0.568392	0.500000	0.256911	0.978587
3	0.670968	0.575786	0.506494	0.253682	0.993631
4	0.684507	0.585952	0.524838	0.251701	0.983806

Cuadro 1.8: Validación con SVM kernel Chi Cuadrado

Validation	**F1**	**accuracy**	**precision**	**truenegativerate**	**truepositiverate**
0	0.970854	0.973198	0.977733	0.981067	0.964072
1	0.975242	0.978743	0.997797	0.998353	0.953684
2	0.976999	0.980591	0.986726	0.990338	0.967462
3	0.977459	0.979667	0.987578	0.989813	0.967546
4	0.988004	0.989834	0.993421	0.995169	0.982646

Cuadro 1.9: Validación con SVM kernel Inter

Capítulo 2

Clasificación con descriptor LBP

2.1. Pruebas con LBP

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.722057	0.740296	0.717092	0.751724	0.727092
1	0.724521	0.747689	0.699805	0.745033	0.751046
2	0.750000	0.767098	0.720000	0.754591	0.782609
3	0.752753	0.771719	0.743083	0.779287	0.762677
4	0.705385	0.742144	0.685832	0.754019	0.726087

Cuadro 2.1: Validación con SVM kernel lineal

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.609646	0.439002	0.438483	0.001645	1.0
1	0.620778	0.450092	0.450092	0.000000	1.0
2	0.612821	0.441774	0.441774	0.000000	1.0
3	0.637681	0.468577	0.468085	0.001736	1.0
4	0.611432	0.440850	0.440333	0.001650	1.0

Cuadro 2.2: Validación con SVM kernel RBF

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.757515	0.776340	0.744094	0.780405	0.771429
1	0.742911	0.748614	0.711957	0.723958	0.776680
2	0.752437	0.765250	0.735238	0.760757	0.770459
3	0.738144	0.765250	0.720322	0.771757	0.756871
4	0.754297	0.775416	0.732809	0.774086	0.777083

Cuadro 2.3: Validación con SVM kernel polinómico grado 2

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.762548	0.772643	0.745283	0.765625	0.780632
1	0.769691	0.786506	0.745174	0.778894	0.795876
2	0.753651	0.766174	0.706204	0.733002	0.807933
3	0.763674	0.788355	0.728346	0.777778	0.802603
4	0.768145	0.787431	0.750000	0.787625	0.787190

Cuadro 2.4: Validación con SVM kernel polinómico grado 3

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.347555	0.420518	0.352321	0.484034	0.342916
1	0.359743	0.447320	0.365217	0.519737	0.354430
2	0.288248	0.406654	0.311005	0.518395	0.268595
3	0.727823	0.750462	0.691571	0.736928	0.768085
4	0.373253	0.419593	0.349533	0.434146	0.400428

Cuadro 2.5: Validación con SVM kernel polinómico grado 4

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.613709	0.442699	0.442699	0.0	1.0
1	0.612821	0.441774	0.441774	0.0	1.0
2	0.624285	0.453789	0.453789	0.0	1.0
3	0.623410	0.452865	0.452865	0.0	1.0
4	0.624285	0.453789	0.453789	0.0	1.0

Cuadro 2.6: Validación con SVM kernel polinómico grado 5

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.619898	0.449168	0.449168	0.0	1.0
1	0.626032	0.455638	0.455638	0.0	1.0
2	0.609254	0.438078	0.438078	0.0	1.0
3	0.607465	0.436229	0.436229	0.0	1.0
4	0.629512	0.459335	0.459335	0.0	1.0

Cuadro 2.7: Validación con SVM kernel sigmoidal

Validation	**F1**	**accuracy**	**precision**	**truenegativerate**	**truepositiverate**
0	0.619017	0.448244	0.448244	0.000000	1.0
1	0.626429	0.456562	0.456059	0.001698	1.0
2	0.625478	0.456562	0.455051	0.005076	1.0
3	0.612323	0.441774	0.441258	0.001653	1.0
4	0.609646	0.439002	0.438483	0.001645	1.0

Cuadro 2.8: Validación con SVM kernel Chi Cuadrado

Validation	**F1**	**accuracy**	**precision**	**truenegativerate**	**truepositiverate**
0	0.922912	0.933457	0.907368	0.929374	0.938998
1	0.926625	0.935305	0.920833	0.937500	0.932489
2	0.924731	0.935305	0.932755	0.949429	0.916844
3	0.934322	0.942699	0.928421	0.944535	0.940299
4	0.939516	0.944547	0.939516	0.948805	0.939516

Cuadro 2.9: Validación con SVM kernel Inter

Capítulo 3

Clasificación con descriptor LBP Uniforme

3.1. Pruebas con LBP Uniforme

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.865234	0.872458	0.848659	0.863793	0.882470
1	0.872093	0.878004	0.837989	0.851789	0.909091
2	0.844534	0.856747	0.788390	0.817447	0.909287
3	0.867133	0.877079	0.841085	0.862647	0.894845
4	0.859556	0.877079	0.856842	0.888525	0.862288

Cuadro 3.1: Validación con SVM kernel lineal

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.642812	0.474122	0.473636	0.001754	1.0
1	0.629512	0.459335	0.459335	0.000000	1.0
2	0.618135	0.447320	0.447320	0.000000	1.0
3	0.589693	0.418669	0.418131	0.001587	1.0
4	0.614001	0.444547	0.443003	0.004967	1.0

Cuadro 3.2: Validación con SVM kernel RBF

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.865385	0.870610	0.830258	0.842466	0.903614
1	0.877953	0.885397	0.851145	0.867797	0.906504
2	0.870010	0.882625	0.853414	0.878939	0.887265
3	0.869295	0.883549	0.846465	0.876020	0.893390
4	0.848849	0.860444	0.820116	0.845000	0.879668

Cuadro 3.3: Validación con SVM kernel polinómico grado 2

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.870293	0.885397	0.845528	0.877023	0.896552
1	0.882061	0.890018	0.854127	0.872054	0.911885
2	0.877228	0.885397	0.847036	0.865546	0.909651
3	0.862000	0.872458	0.850099	0.870968	0.874239
4	0.864097	0.876155	0.827184	0.854337	0.904459

Cuadro 3.4: Validación con SVM kernel polinómico grado 3

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.869059	0.875231	0.820513	0.835846	0.923711
1	0.883813	0.891867	0.849237	0.868114	0.921325
2	0.859330	0.864140	0.825368	0.836489	0.896208
3	0.875764	0.887246	0.838207	0.864600	0.916844
4	0.847695	0.859519	0.804183	0.831148	0.896186

Cuadro 3.5: Validación con SVM kernel polinómico grado 4

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.627774	0.457486	0.457486	0.0	1.0
1	0.621656	0.451017	0.451017	0.0	1.0
2	0.616368	0.445471	0.445471	0.0	1.0
3	0.619898	0.449168	0.449168	0.0	1.0
4	0.619898	0.449168	0.449168	0.0	1.0

Cuadro 3.6: Validación con SVM kernel polinómico grado 5

Validation	**F1**	**accuracy**	**precision**	**truenegativerate**	**truepositiverate**
0	0.640704	0.471349	0.471349	0.0	1.0
1	0.644110	0.475046	0.475046	0.0	1.0
2	0.592062	0.420518	0.420518	0.0	1.0
3	0.643260	0.474122	0.474122	0.0	1.0
4	0.602067	0.430684	0.430684	0.0	1.0

Cuadro 3.7: Validación con SVM kernel sigmoidal

Validation	**F1**	**accuracy**	**precision**	**truenegativerate**	**truepositiverate**
0	0.621173	0.451017	0.450509	0.001681	1.0
1	0.608360	0.437153	0.437153	0.000000	1.0
2	0.618135	0.447320	0.447320	0.000000	1.0
3	0.625556	0.455638	0.455134	0.001695	1.0
4	0.625159	0.454713	0.454713	0.000000	1.0

Cuadro 3.8: Validación con SVM kernel Chi Cuadrado

Validation	**F1**	**accuracy**	**precision**	**truenegativerate**	**truepositiverate**
0	0.924025	0.931608	0.912779	0.928453	0.935551
1	0.928355	0.934381	0.918164	0.930743	0.938776
2	0.937033	0.945471	0.940043	0.954248	0.934043
3	0.938525	0.944547	0.952183	0.960818	0.925253
4	0.922441	0.930684	0.915811	0.931894	0.929167

Cuadro 3.9: Validación con SVM kernel Inter

Capítulo 4

Clasificación con combinación de descriptores

4.1. Pruebas con LBP y HOG combinados

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.723044	0.757856	0.702259	0.767255	0.745098
1	0.705179	0.726433	0.680769	0.722408	0.731405
2	0.721408	0.736599	0.716505	0.745645	0.726378
3	0.715875	0.740296	0.700990	0.747492	0.731405
4	0.727459	0.754159	0.712851	0.763245	0.742678

Cuadro 4.1: Validación con SVM kernel lineal

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.620778	0.450092	0.450092	0.000000	1.0
1	0.619898	0.449168	0.449168	0.000000	1.0
2	0.644514	0.475970	0.475486	0.001761	1.0
3	0.611432	0.440850	0.440333	0.001650	1.0
4	0.612323	0.441774	0.441258	0.001653	1.0

Cuadro 4.2: Validación con SVM kernel RBF

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.734486	0.758780	0.730769	0.775717	0.738241
1	0.745174	0.756007	0.704380	0.727273	0.790984
2	0.726688	0.764325	0.698969	0.769716	0.756696
3	0.741107	0.757856	0.706215	0.740433	0.779626
4	0.748996	0.768946	0.741551	0.779287	0.756592

Cuadro 4.3: Validación con SVM kernel polinómico grado 2

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.772139	0.788355	0.740458	0.773710	0.806653
1	0.787234	0.796673	0.749540	0.769882	0.828921
2	0.756320	0.777264	0.712381	0.755663	0.806034
3	0.785855	0.798521	0.736648	0.764415	0.842105
4	0.743083	0.759704	0.716190	0.749580	0.772074

Cuadro 4.4: Validación con SVM kernel polinómico grado 3

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.367816	0.440850	0.375267	0.506734	0.360656
1	0.314894	0.404806	0.341014	0.503472	0.292490
2	0.707269	0.724584	0.683112	0.717428	0.733198
3	0.365011	0.456562	0.378924	0.539867	0.352083
4	0.382353	0.456562	0.367677	0.499200	0.398249

Cuadro 4.5: Validación con SVM kernel polinómico grado 4

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.619898	0.449168	0.449168	0.0	1.0
1	0.596628	0.425139	0.425139	0.0	1.0
2	0.628644	0.458410	0.458410	0.0	1.0
3	0.615483	0.444547	0.444547	0.0	1.0
4	0.604771	0.433457	0.433457	0.0	1.0

Cuadro 4.6: Validación con SVM kernel polinómico grado 5

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.621656	0.451017	0.451017	0.0	1.0
1	0.624285	0.453789	0.453789	0.0	1.0
2	0.618135	0.447320	0.447320	0.0	1.0
3	0.626032	0.455638	0.455638	0.0	1.0
4	0.605670	0.434381	0.434381	0.0	1.0

Cuadro 4.7: Validación con SVM kernel sigmoidal

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.610148	0.439002	0.439002	0.0	1.0
1	0.590228	0.418669	0.418669	0.0	1.0
2	0.614597	0.443623	0.443623	0.0	1.0
3	0.610148	0.439002	0.439002	0.0	1.0
4	0.619898	0.449168	0.449168	0.0	1.0

Cuadro 4.8: Validación con SVM kernel Chi Cuadrado

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.936475	0.942699	0.930754	0.943049	0.942268
1	0.942553	0.950092	0.944563	0.957447	0.940552
2	0.936992	0.942699	0.933198	0.944257	0.940816
3	0.930818	0.939002	0.928870	0.943894	0.932773
4	0.942249	0.947320	0.958763	0.965517	0.92629

Cuadro 4.9: Validación con SVM kernel Inter

4.2. Pruebas con LBP Uniforme y HOG combinados

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.872763	0.881701	0.859100	0.877342	0.886869
1	0.843198	0.860444	0.825203	0.859247	0.861996
2	0.870352	0.880776	0.849020	0.871022	0.892784
3	0.866397	0.878004	0.864646	0.886248	0.868154
4	0.851020	0.865065	0.834000	0.862126	0.868750

Cuadro 4.10: Validación con SVM kernel lineal

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.626429	0.456562	0.456059	0.001698	1.0
1	0.606061	0.435305	0.434783	0.001634	1.0
2	0.606959	0.436229	0.435708	0.001637	1.0
3	0.611432	0.440850	0.440333	0.001650	1.0
4	0.602456	0.431608	0.431082	0.001623	1.0

Cuadro 4.11: Validación con SVM kernel RBF

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.863354	0.878004	0.842424	0.872340	0.885350
1	0.851365	0.864140	0.820663	0.848185	0.884454
2	0.856863	0.865065	0.830798	0.848639	0.884615
3	0.865580	0.878004	0.838264	0.864909	0.894737
4	0.874239	0.885397	0.858566	0.881271	0.890496

Cuadro 4.12: Validación con SVM kernel polinómico grado 2

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.895382	0.897412	0.869963	0.874780	0.922330
1	0.863859	0.871534	0.833648	0.850847	0.896341
2	0.880553	0.888170	0.864341	0.880342	0.897384
3	0.872319	0.884473	0.837255	0.864600	0.910448
4	0.892754	0.897412	0.871698	0.882149	0.914851

Cuadro 4.13: Validación con SVM kernel polinómico grado 3

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.882704	0.890943	0.855491	0.873950	0.911704
1	0.864754	0.878004	0.819417	0.850242	0.915401
2	0.870466	0.884473	0.831683	0.863344	0.913043
3	0.886228	0.894640	0.847328	0.867550	0.928870
4	0.874146	0.880776	0.840525	0.855932	0.910569

Cuadro 4.14: Validación con SVM kernel polinómico grado 4

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.624285	0.453789	0.453789	0.0	1.0
1	0.637280	0.467652	0.467652	0.0	1.0
2	0.599353	0.427911	0.427911	0.0	1.0
3	0.619898	0.449168	0.449168	0.0	1.0
4	0.649189	0.480591	0.480591	0.0	1.0

Cuadro 4.15: Validación con SVM kernel polinómico grado 5

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.620778	0.450092	0.450092	0.0	1.0
1	0.600259	0.428835	0.428835	0.0	1.0
2	0.617252	0.446396	0.446396	0.0	1.0
3	0.624285	0.453789	0.453789	0.0	1.0
4	0.610148	0.439002	0.439002	0.0	1.0

Cuadro 4.16: Validación con SVM kernel sigmoidal

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.617252	0.446396	0.446396	0.0	1.0
1	0.633838	0.463956	0.463956	0.0	1.0
2	0.634700	0.464880	0.464880	0.0	1.0
3	0.632975	0.463031	0.463031	0.0	1.0
4	0.592062	0.420518	0.420518	0.0	1.0

Cuadro 4.17: Validación con SVM kernel Chi Cuadrado

Validation	F1	accuracy	precision	truenegativerate	truepositiverate
0	0.940239	0.944547	0.927308	0.936968	0.953535
1	0.921466	0.930684	0.918580	0.935644	0.924370
2	0.933761	0.942699	0.918067	0.937299	0.950000
3	0.931452	0.937153	0.914851	0.927731	0.948665
4	0.934120	0.938078	0.920543	0.929432	0.948104

Cuadro 4.18: Validación con SVM kernel Inter

Capítulo 5

Detector de personas multiescala