Sistema de clasificación de rostros con el método de Histograma de gradientes orientados

February 2021

Resumen

Se presentarán resultados de la implementación de método de histograma de gradientes orientados para la clasificación de rostros humanos.

1. Introducción

1.1. Detección y reconocimiento facial

La detección facial es el proceso en el que el software determina, mediante algoritmos, si hay rostros humanos en una foto o vídeo. No determina la identidad de una persona, tan solo determina si hay alguna cara. Por otro lado el sistema de reconocimiento facial es una aplicación que se encarga de identificar automáticamente a una persona, el cual realiza un análisis de las características faciales del usuario adquiridas mediante una imagen y comparándolas con una base de datos.

1.2. HOG: Histograma de gradientes orientados

Los algoritmos de procesamiento buscan extraer características al convertir una imagen de tamaño fijo en un vector. Esta transformación sirve para detectar patrones.

Es bastante común que una imagen de entrada posea demasiada información adicional que no es necesaria para la detección de una clase de objeto, por ejemplo un rostro humano. Por lo tanto, el primer paso en proceso de la detección de objetos es simplificar la imagen extrayendo la información importante contenida en esta y dejando de lado el resto. Este paso se llama extracción de características el cual consiste en transformar una imagen de tamaño fijo en un vector de longitud fija, el histograma de gradientes orientados **HOG** es referencia en el campo a partir del trabajo presentado por Dalal-Triggs [2]

1.3. Clasificador lineal SVM

Un modelo SVM es una representación de los vectores HOG como puntos en el espacio, mapeados de modo que los HOG que contienen un rostro humano

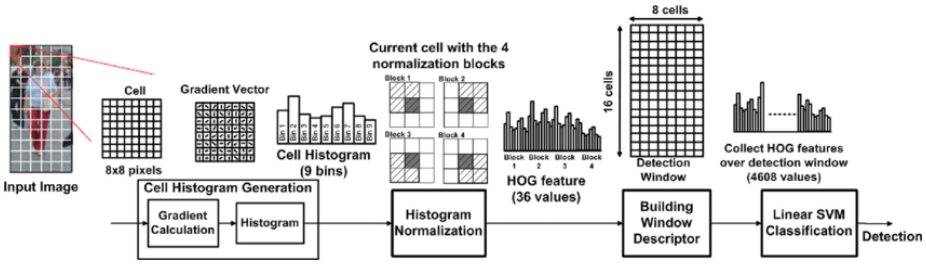

Figura 1: Diferentes etapas de un clasificador de objetos utilizando HOG como vector de características de la imagen

queden separados por el margen más grande posible de aquellos que no. Luego, los nuevos HOG se mapean en ese mismo espacio y se predice a que categoría pertenecen según el lado del espacio que se les asigna.

2. Desarrollo

Usando HOG como vector característico de una imagen , podemos construir un algoritmo de clasificación de rostros simple Scikit-Learn [10] con una SVM como clasificador. Los pasos son los siguientes:

- Obtener un conjunto de imágenes de rostros correspondientes a diferentes clases, es este caso a diferentes individuos.
- Dividir el conjunto en: imágenes de entrenamiento e imágenes de prueba.
- Extraer los vectores HOG de este conjunto de imágenes.
- Entrenar el clasificador SVM con un subconjunto los vectores HOG y sus clases correspondientes.
- Comparar las clases resultado de usar el modelo para clasificar el subconjunto restante de vectores HOG.

2.1. Obtención de los datos

. Scikit-Lear proporciona 2 conjunto de datos como los requeridos para el desarrollo del presente trabajo:

- The Olivetti faces dataset
- The Labeled Faces in the Wild face recognition dataset

Se utilizó el primero que consta de 400 imágenes correspondientes a 40 sujetos o clases mediante la siguiente instrucción:

```
datasets.fetch_olivetti_faces()
```

Figura 2: Imágenes correspondientes al conjunto Olivetti

2.2. Conjunto de entrenamiento y pruebas

Para dividir las imágenes se utilizo la función

```
train_test_split(faces.data,faces.target)
```

de scikit-learn

2.3. Estracción vectores HOG

Para este punto se utilizaron 2 extractores de los vectores HOG, el proporcionado por scikit-learn y el proporcionado por el paquete OpenCV. La razón de lo anterior como se mostrarán en los resultados fue que los vectores HOG obtenidos por medio de scikit-learn fueron clasificados mal en promedio 90 por ciento, por lo que se buscaron alternativas para la extracción del HOG. Cabe mencionar que el conjunto de datos Olivetti están preparados para ser procesados por las funciones de scikit-learn por lo que para ser utilizadas por OpenCv se tuvo que hacer un preprocesamiento que consistió en convertir la imagen de valores de punto flotante a valores entero sin signo de 8 bits.

2.4. Entrenamiento del clasificador SVM

Consistió en instanciar el objeto **svm.SVC** con diferentes kernel y la constante de margen C, posteriormente entrenar el clasificador con el subconjunto

datos de entrenamiento.

2.5. Comparación de resultados

Se invoca al modelo con los vectores HOG del subconjunto pruebas, las clases obtenidas se comparan con las clases originales restando la clase asociada por la SVM con la clase original, aquellas cuyo resultado es **0** corresponden a clasificaciones correctas.

3. Resultados

Los principales valores que se modificaron para la obtención del modelo SVM fueron el porcentaje de vectores utilizados para el entrenamiento train_size y el kernel: {'linear', 'poly', 'rbf', 'sigmoid'}

Kernel	**precisión**	**exactitud**	**Recall**
linear	0.993	0.975	0.975
poly	0.993	0.975	0.975
rbf	0.982	0.975	0.975
sigmoid	0.019	0.062	0.062

Tabla 1: Comparación de diferentes kernel para el entrenamiento del SVM

% Entrenamiento	**precisión**	**exactitud**	**Recall**	**Aciertos**
50	0.967	0.955	0.955	191/200
75	0.979	0.96	0.96	96/100
80	0.993	0.975	0.975	78/80

Tabla 2: Comparación de diferentes porcentajes de vectores utilizados para el entrenamiento del SVM

Método	**precisión**	**exactitud**	**Recall**
Eigenfaces	1	0.94	1
Cnn_aleatorio	0.1588	0.9730	0.3424
HOG[1]	0.995	0.991	0.991
HOG[2]	0.993	0.975	0.975

Tabla 3: Comparación del método HOG con otras implementaciones. HOG[1] 70 %, HOG[2] 80 % de vectores utilizados para entrenamiento

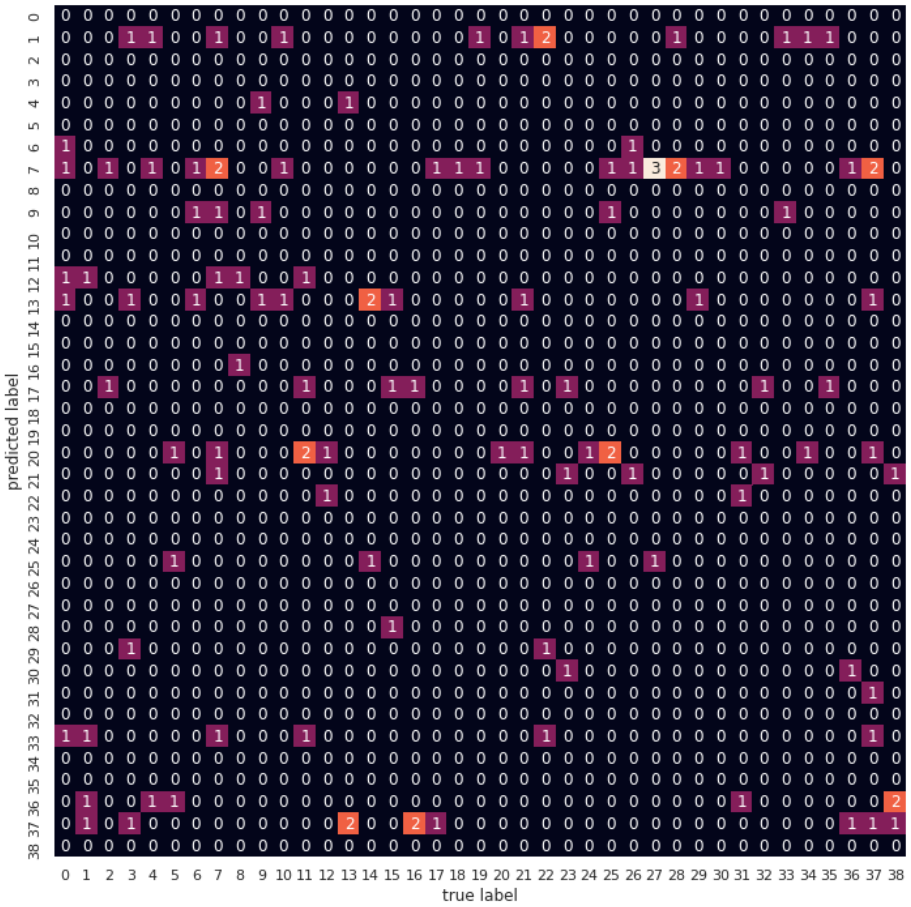

Figura 3: Gráfico "heatmap"de clasificación con kernel sigmoid.

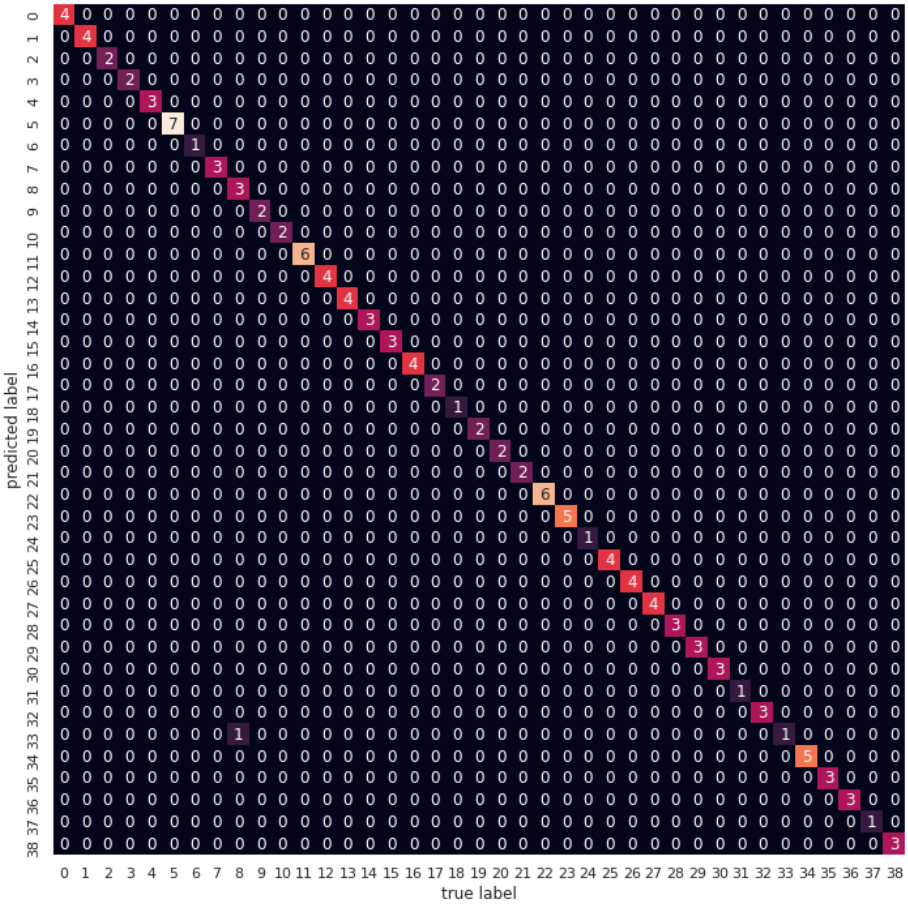

Figura 4: Gráfico "heatmap"de clasificación con kernel linear.

4. Conclusiones

5. Bibliografía

Referencias

[1] Lubna, M. F. Khan and N. Mufti, Çomparison of various edge detection filters for ANPR,"2016 Sixth International Conference on Innovative Computing Technology (INTECH), Dublin, 2016, pp. 306-309, doi: 10.1109/INTECH.2016.7845061. Michel Goossens, Frank Mittelbach, and Alexander Samarin.

[2] Navneet Dalal, Bill Triggs. Histograms of Oriented Gradients for Human Detection. InternationalConference on Computer Vision and Pattern Recognition (CVPR '05), Jun 2005, San Diego, UnitedStates. pp.886–893, 10.1109/CVPR.2005.177. inria-00548512

[3] A Discriminatively Trained, Multiscale, Deformable Part Model P. Felzenszwalb, D. McAllester, D. Ramanan IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008

[4] P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, September 2010

[5] P. Viola and M. Jones, Robust real-time face detection,"Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, Vancouver, BC, Canada, 2001, pp. 747-747, doi: 10.1109/ICCV.2001.937709.

[6] Yang, Ming-Hsuan and Kriegman, David and Ahuja, Narendra. (2002). Detecting Faces in Images: A Survey. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 24. 34 - 58. 10.1109/34.982883.

[7] Suleiman, Amr and Sze, Vivienne. (2016). An Energy-Efficient Hardware Implementation of HOG-Based Object Detection at 1080HD 60 fps with Multi-Scale Support. Journal of Signal Processing Systems. 84. 10.1007/s11265-015-1080-7.

[8] Python is a programming language that lets you work more quickly and integrate your systems more effectively.
`https://www.python.org/`

[9] The fundamental package for scientific computing with Python
`https://numpy.org/`

[10] Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning.
`https://scikit-learn.org/`