Tarea 3 EL7008 – Primavera 2023
Clasificación de edad usando características tipo HOG y LBP
(esta tarea tiene Nota Doble, equivalente a 2 tareas)

Profesor: Javier Ruiz del Solar
Auxiliar: Patricio Loncomilla

Fecha enunciado: 22 de Septiembre
Fecha entrega: 16 de Octubre

El objetivo de esta tarea es diseñar y construir un sistema de clasificación de edad, que utilice características tipo HOG (Histograms of Oriented Gradients) y tipo LBP (Local Binary Patterns), y clasificadores SVM y *Random Forest*. Se debe utilizar las librerías OpenCV, numpy y scikit-learn, pues contienen varias de las funcionalidades requeridas en la tarea.

Preparación de Conjuntos de Entrenamiento y Prueba

Se debe utilizar las imágenes de la base de datos subida a U-Cursos (600 imágenes de personas), la cual incluye 200 imágenes con personas de 1 año de edad, 200 imágenes con personas de 5 años de edad y 200 imágenes con personas de 28 o más años de edad. Esta base de datos debe ser separada en 60% para entrenamiento, 20% para validación y 20% para prueba.

Extracción de Características

Histograms of Oriented Gradients (HOG) [1]: El método procesa imágenes de un tamaño fijo (ejemplo: 64x128). Se calculan gradientes en la imagen y se dividen espacialmente usando 8x16 celdas. En cada celda se calcula un histograma de orientación del gradiente con 9 componentes (orientaciones). Finalmente se hace una normalización por bloques, y se genera el histograma final concatenando los histogramas. Un ejemplo de histogramas de gradientes orientados se muestra en la figura 1. En el caso de esta tarea, se usarán 8x8 celdas.

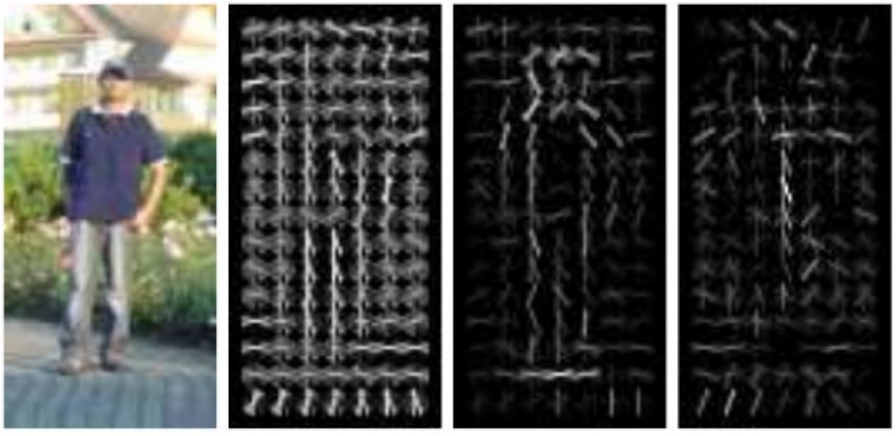

Figura 1: Imagen de prueba, HOG, HOG ponderado por los pesos positivos del SVM, y HOG ponderado por los pesos negativos del SVM (figura extraída del paper original)

[1] Ver paper: Navneet Dalal, Bill Triggs. “Histograms of Oriented Gradients for Human Detection”.

Histogramas LBP[2]: El método extrae las características utilizando histogramas de características LBP. Se definen tres niveles diferentes de localidad: a nivel de píxel, a nivel regional y a nivel global. En el primer nivel de localidad se aplica la transformada LBP sobre la imagen original. El segundo nivel de localidad se obtiene dividiendo la imagen LBP (imagen original con trasformada LBP) en regiones, de cada una de las cuales se extraen histogramas. Estos histogramas se utilizan para una representación eficiente de la información de textura (ver Figura 2). El nivel global de localidad se obtiene concatenando histogramas LBP locales. En este caso deben utilizar las características $LBP_{8,1}^{u2}$ (ver paper[2]) y dividir la imagen en 4x4 regiones.

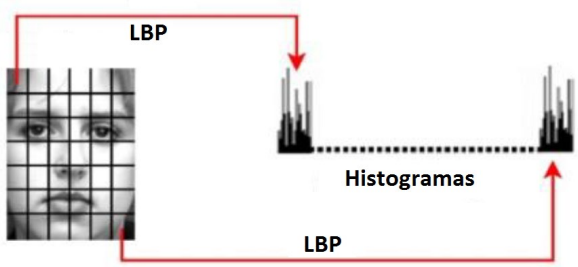

Figura 2: Histogramas LBP.

Clasificación

Se debe entrenar un clasificador SVM y un clasificador *Random Forest* para clasificar los rostros según su edad utilizando características HOG y características LBP. Scikit-learn tiene implementaciones de SVM y de *Random Forest*. Se debe usar *grid search* para buscar los mejores hiperparámetros, tanto para SVM como para *Random Forest*.

Se pide:

Características HOG

1. Implementar una función en python que transforme la imagen a escala de grises, al tipo np.float32.
2. Implementar en python una función que reciba una imagen y calcule sus gradientes (se puede reutilizar código de cálculo de gradientes de la tarea 2).
3. Implementar en python una función que, a partir de los gradientes, calcule las características HOG usando 8x8 celdas (la salida debe ser un arreglo de numpy de dimensión 8x8x9).
4. Implementar en python el *block normalization* para el histograma. Para implementar esto, se deben formar bloques de 2x2 celdas (con traslape). Cada bloque se debe transformar en un vector de 1x36, el cual se debe normalizar. El vector de características final (de tamaño 1x1764) se obtiene concatenando los vectores normalizados de cada bloque, y normalizando el resultado usando norma L2.

[2] Ver paper: T. Ahonen, A. Hadid, M. Pietikainen, "Face recognition with local binary patterns"

5. Extraer características HOG de cada imagen del conjunto de entrenamiento. Se debe aplicar un StandardScaler (usando scikit-learn) a las características para normalizarlas. El StandardScaler se debe entrenar usando sólo las características de entrenamiento.
6. Entrenar un SVM (usando scikit-learn) con las características extraídas a cada imagen del conjunto de entrenamiento. Se debe elegir un kernel y usar *grid search*, usando el conjunto de validación para encontrar los mejores hiper parámetros (se debe usar PredefinedSplit).
7. Entrenar un clasificador *Random Forest* (usando scikit-learn). Se debe usar *grid search*, usando el conjunto de validación para encontrar los mejores hiper parámetros.
8. Realizar las pruebas correspondientes con el conjunto de prueba. Calcular la matriz de confusión sin y con normalización del clasificador SVM y del *Random Forest*.

Características LBP

9. Implementar una función que reciba una imagen y retorne la imagen con la transformada LBP uniforme $\left(LBP_{8,1}^{u2}\right)$. Se puede subdividir esta función en otras menores (ej. Determinación de si un patrón LBP es uniforme, cálculo de una lista para mapear patrones LBP a patrones uniformes, comparación de píxeles centrales v/s sus vecinos recorriendo la imagen) si lo estima adecuado. El recorrido en la imagen se debe implementar en Cython. Mostrar en el informe un ejemplo de una cara a la que se le aplique la transformada implementada.
10. Implementar en Cython el cálculo de histogramas LBP (ver el paper para más detalles), utilizar las características LBP uniformes y dividir las imágenes en 16 regiones (división de 4x4 regiones).
11. Implementar el cálculo de las características LBP. Los vectores de características finales se obtienen concatenando los histogramas LBP de las 16 regiones, y normalizando el resultado usando norma L2.
12. Extraer características LBP de cada imagen del conjunto de entrenamiento. Se debe aplicar un StandardScaler (usando scikit-learn) a las características para normalizarlas. El StandardScaler se debe entrenar usando sólo las características de entrenamiento.
13. Entrenar un SVM (usando scikit-learn) con las características extraídas a cada imagen del conjunto de entrenamiento. Se debe elegir un kernel y usar *grid search*, usando el conjunto de validación para encontrar los mejores hiper parámetros (se debe usar PredefinedSplit).
14. Entrenar un clasificador *Random Forest* (usando scikit-learn). Se debe usar *grid search*, usando el conjunto de validación para encontrar los mejores hiper parámetros.
15. Realizar las pruebas correspondientes con el conjunto de prueba. Calcular la matriz de confusión sin y con normalización del clasificador SVM y del *Random Forest*.
16. Repetir los puntos 9-15, usando características $LBP_{12,2}^{u2}$ (ver paper[1]).

Análisis de resultados:

17. Analizar los resultados obtenidos. ¿Qué tan bien funcionan las características HOG en esta aplicación? ¿Qué tan bien funcionan las características $LBP_{8,1}^{u2}$ y las características $LBP_{12,2}^{u2}$? ¿Qué tan bien funciona cada clasificador para cada tipo de característica? ¿Son apropiadas las características usadas para resolver este problema? ¿Cómo se pueden mejorar los resultados?
18. Documentar cada uno de los pasos anteriores en el informe

El código entregado debe ejecutar el entrenamiento, usando los tres tipos de característica (HOG, $LBP_{8,1}^{u2}$ y $LBP_{12,2}^{u2}$), usando tanto SVM como *Random Forest* con *grid search* y debe calcular las matrices de confusión (sin y con normalización) correspondientes a cada uno de estos casos usando el conjunto de prueba. Se debe usar un jupyter notebook y el código debe correr en colaboratory.

Los informes (en formato PDF), los códigos (en formato .ipynb) y el archivo README.txt deben ser subidos a U-Cursos hasta las 23:59 horas del día lunes 16 de Octubre.

Importante: La evaluación de esta tarea considerará el correcto funcionamiento del código, la calidad de los experimentos realizados y de su análisis, las conclusiones, así como la prolijidad y calidad del informe entregado.

Nota: En colaboratory, se recomienda subir el .zip con las imágenes y ejecutar:

```
!unzip imagenes_tarea3_2023_edad.zip
```

Nota: El informe de la tarea en formato PDF debe ser subido a turnitin, con el código agregado en un anexo, en modo texto (no como captura de imagen). Para asegurar que el informe pueda ser subido a turnitin, no se permite generar el informe directamente a partir del notebook.

Nota: El *grid search* se debe realizar usando el conjunto de validación, no *cross validation*. Además, se debe reentrenar con el conjunto de entrenamiento una vez encontrados los mejores hiper parámetros.

Nota extra: Dado que la ejecución del código de HOG puede ser lenta en Python, se recomienda usar numba o bien cython en las funciones que hacen operaciones píxel a píxel. Numba se usa del siguiente modo:

```
from numba import jit

@jit(nopython=True)
def computeHog(gradx, grady):
  # Cuerpo de la función en Python normal
```