Tarea 4 EL7008 – Primavera 2021
Clasificación de edad usando LBP y redes neuronales

Profesor: Javier Ruiz del Solar
Auxiliar: Patricio Loncomilla

Fecha enunciado: 14 de Octubre
Fecha entrega: 27 de Octubre

El objetivo de esta tarea es diseñar y construir un sistema de clasificación de edad, que utilice características tipo *Histogramas LBP* y redes neuronales como clasificador estadístico. La base de datos a usar es la misma de la tarea 3. Se debe programar el cálculo de los histogramas LBP usando Cython.

Preparación de Conjuntos de Entrenamiento y Test

En esta tarea se debe utilizar las imágenes de la base de datos subida a U-Cursos (600 imágenes), la cual incluye 200 imágenes de personas con edades en el rango 1-4 años, 200 imágenes con personas en el rango 5-27 años y 200 imágenes de personas con 28+ años. Esta base de datos debe ser separada en 60% para entrenamiento, 20% para validación y 20% para prueba.

Extracción de Características

Histogramas LBP[1]: El método extrae las características utilizando histogramas de imágenes 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) del rostro 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 1). El nivel global de localidad, es decir, descripción del rostro, se obtiene concatenando histogramas LBP locales. En este caso deben utilizar las características $LBP_{8,1}^{u2}$ (ver paper[1]) y dividir en 4 regiones correspondientes a los cuatro cuadrantes de las caras.

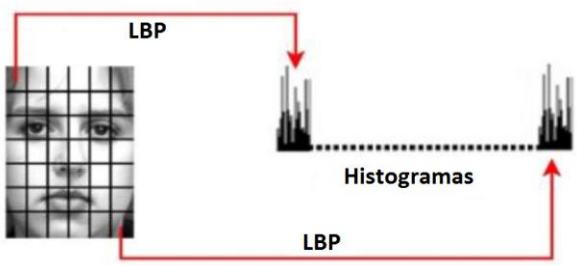

Figura 1: Histogramas LBP.

[1] Ver paper: T. Ahonen, A. Hadid, M. Pietikainen, “Face recognition with local binary patterns”

Clasificación

Se debe entrenar redes neuronales para diferenciar distintos rangos de edad utilizando las características LBP. Se usará la biblioteca *pytorch* para entrenar las redes neuronales. Las redes deben tener una única capa oculta, con función de activación ReLU, y una capa de salida con 3 neuronas y función de activación softmax. La función de *loss* (pérdida) es entropía cruzada. El optimizador que se debe usar es *Adam*. La función *softmax* está implícita al usar la función de pérdida *CrossEntropyLoss* de pytorch (no se debe agregar *softmax* a la salida de la red).

Se pide:

1. Implementar en Cython 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 si lo estima adecuado. Mostrar en el informe un ejemplo de una cara a la que se le aplique la transformada implementada.
2. Implementar en Cython el cálculo de histogramas LBP (ver el paper para más detalles), utilizar las características LBP uniformes y dividir las imágenes en 4 regiones (división de 2x2 regiones).
3. Extraer los Histogramas LBP uniformes de cada cara del conjunto de entrenamiento, validación y prueba. Estos histogramas son el resultado de la concatenación de los histogramas de los cuatro cuadrantes. Se debe usar un *StandardScaler* para normalizar los histogramas.
4. Entrenar una red neuronal con las características extraídas a cada ejemplo (cara) del conjunto de entrenamiento. Se debe usar el conjunto de validación para prevenir sobreajuste de la red. Probar cuatro configuraciones: dos diferentes números de neuronas en la capa oculta, y dos diferentes *learning rate* iniciales del optimizador. Para cada red a ser entrenada:
    a. Calcule y grafique el *loss* de entrenamiento y el de validación, así como el tiempo de procesamiento requerido. Ambas curvas deben mostrarse en un mismo gráfico, colocando las leyendas adecuadas.
    b. Genere la matriz de confusión normalizada y el accuracy normalizado, usando el conjunto de entrenamiento. Se debe mostrar la matriz de confusión con distintos colores o tonos de gris según el valor del accuracy. La matriz se debe calcular usando scikit-learn, usando la opción normalize = "true".
    c. Genere la matriz de confusión normalizada y el accuracy normalizado, usando el conjunto de validación. Se debe mostrar la matriz de confusión con distintos colores según el valor del accuracy.
5. Usando la mejor red encontrada en validación (aquella con mayor accuracy en validación), calcule la matriz de confusión normalizada y el accuracy normalizado, usando el conjunto de prueba.
6. Repetir los puntos 1-5, usando características $LBP_{12,2}^{u2}$ (ver paper[1]).
7. Analizar los resultados obtenidos con los clasificadores entrenados. ¿Qué tan bien funciona cada clasificador? ¿Cómo se pueden mejorar los resultados? ¿Son apropiadas las características LBP para este problema? ¿Cómo se pueden mejorar los resultados?
8. Documentar cada uno de los pasos anteriores en el informe

El código entregado debe ejecutar un entrenamiento y mostrar la evaluación del clasificador entrenado. Se debe entregar el informe y el notebook con el código. El notebook debe funcionar en colaboratory. Se debe activar el uso de GPU en colaboratory para acelerar el proceso de entrenamiento. Se debe entregar las instrucciones para su ejecución en un archivo README.txt.

En relación al entrenamiento, se debe evitar el sobreajuste (overfitting) de la red. Para esto se debe detener el entrenamiento cuando el *loss* de validación comience a aumentar, mientras el de entrenamiento siga bajando. Como las mediciones de *loss* pueden tener fluctuaciones (ruido), en lapsos cortos de tiempo, la implementación de esta funcionalidad debe realizarse en forma robusta. El código debe guardar *checkpoints* para poder recuperar las redes correspondientes a distintas épocas de entrenamiento. Se debe recuperar la red que obtenga el menor *loss* de validación. Además, se debe indicar el tiempo de entrenamiento necesario para poder obtener dicha red.

Los informes, los códigos (en formato .ipynb) y el archivo README.txt deben ser subidos a U-Cursos hasta las 23:59 horas del día miércoles 27 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 db_tarea_3_2021.zip
```

Nota: El informe de la tarea en formato PDF debe ser subido a turnitin, con el código agregado en un anexo.