Laboratorio de Análisis y Procesamiento de Imágenes
Módulo IV
Fechas de realización: 26 de abril - 18 de mayo de 2021

Universidad de los Andes

Proyecto 4:

Clasificación Supervisada y No-Supervisada de Imágenes

1. Objetivos

- Comprender el flujo de trabajo en el reconocimiento visual.
- Comprender el valor y objetivo de los procesos de entrenamiento, validación y prueba en la determinación de un modelo óptimo para una tarea particular.
- Realizar una aproximación a un proceso de experimentación en visión por computador, utilizando herramientas clásicas.
- Familiarizarse con las librerías de Python para el procesamiento de imágenes.

2. Parámetros de entrega

Entrega por Brightspace:

- Adjuntar un archivo con los códigos de la siguiente forma:

  main_CódigoEstudiante1_CódigoEstudiante2.py. **Es obligatoria la entrega de un script *.py***. **NO** se recibirán archivos de Jupyter Notebook (extensión *.ipynb*). Su entrega conllevará una penalización sobre la calificación final del proyecto.

  - El código enviado debe tener una longitud máxima de 500 líneas. Se generará una penalización sobre la calificación final de cada entrega por cada 10 líneas adicionales. Esto con el objetivo de promover el uso de estructuras iterativas y operaciones vectoriales, que aumenten la eficiencia de sus códigos (aspecto fundamental en tareas de programación).

- Adjuntar **TODO** el código que se entregó en el ítem anterior, en formato *.txt*. Llámelo de igual manera: *CódigoEstudiante1_CódigoEstudiante2.txt*. Esto con el fin de poder evaluar en Brightspace automáticamente cualquier intento de copia o similitud entre los algoritmos. Cualquier intento de copia será tratado de acuerdo al reglamento de la Universidad. **Aquel grupo que no incluya este ítem en cualquiera de sus entregas tendrá una nota de 0 en la entrega**.

- Adjuntar un único archivo con el informe en PDF con el mismo nombre del primer ítem: *CódigoEstudiante1_CódigoEstudiante2.pdf*. El informe debe presentarse con el formato CVPR (para más información del informe, ver sección de **Parámetros de calificación**).
- En el caso de que no tenga compañero, **DEBE** utilizar un cero como el código de su compañero, i.e. *main_CódigoEstudiante_0.py* para el archivo de código, *CódigoEstudiante_0.pdf* para el informe y *CódigoEstudiante_0.txt* para el archivo de texto.
- **NO** se permiten archivos comprimidos tales como *zip, rar, 7z, tar, gz, xz, iso, cbz*, etc. Aquel grupo que envíe su informe como un archivo comprimido no tendrá calificación.
- No se recibirá ningún archivo por algún medio diferente a Brightspace.
- Este proyecto tiene una duración de tres semanas y está compuesto por tres entregas (una por semana). Estas se realizarán para las siguientes fechas:
  - Entrega 1: Domingo 9 de mayo de 2020
  - Entrega 3: Martes 18 de mayo de 2020

  El plazo de entrega será hasta las 11:59 p.m. de las fechas establecidas. El vínculo para el envío de los documentos dejará de estar disponible luego de esta hora. **No se recibirán documentos o archivos después de la hora máxima de entrega** (tenga en cuenta que la hora queda registrada en el envío de Brightspace). Cada estudiante tiene varios intentos en Brightspace para cada entrega, pueden hacer entregas parciales y se tendrá en cuenta únicamente el último intento para la calificación.
- Cada entrega del presente miniproyecto recibirá una calificación por separado. Estas contarán con los siguientes pesos sobre la nota final del curso:
  - Entrega 1: 4 %
  - Entrega 2: 6 %

3. Reglas generales

- La asistencia a la sección de laboratorio **inscrita** es **obligatoria**. De acuerdo con el Reglamento General de Estudiantes de Pregrado de la Universidad de los Andes, la inasistencia a más del 20 % de las clases de laboratorio resultará en la reprobación de la materia completa (laboratorio y magistral).
- Los informes deben realizarse **únicamente** con la pareja. Esto quiere decir que, aunque es válido discutir los problemas con sus compañeros, la solución y el código deben ser de su completa autoría. Está prohibido copiar literalmente el algoritmo y/o procedimiento desarrollado por otro grupo o tercero. Si llega a obtener un código de Internet, asociado al problema a resolver, este debe estar debidamente **referenciado** y usted debe **entenderlo por completo**. Este último aspecto no es válido para las funciones propias, pues estas deben ser implementaciones concebidas por cada una de las parejas.

4. Parámetros de calificación

Resultados:

1. Todos los códigos deben mantener orden y coherencia en la ejecución de comandos, es decir, cada vez que se muestre una figura, el programa debe esperar para que se presione una tecla, para así continuar con la siguiente y así sucesivamente (para esto utilice en Python `input("Press Enter to continue...")`. Si quieren contrastar dos o más imágenes utilicen `subplot`.
2. Toda figura debe estar numerada y debe llevar su título y descripción en el informe.
3. El código debe estar debidamente comentado.
4. NUNCA utilice rutas absolutas para leer o guardar archivos. Este es el error más común en la ejecución de los códigos y será penalizado.
5. Para generar rutas utilice `os.path.join` en Python, ya que puede que corramos los laboratorios usando Linux o Windows y los separadores de archivos cambian dependiendo del sistema operativo.
6. Asuma que dentro de la carpeta de ejecución del código se encuentra el directorio proporcionado (`data_mp4`) con los archivos necesarios para la realización del laboratorio.

   Ejemplo: Dentro del código principal, el estudiante quiere leer la imagen `im.png` que está dentro de una carpeta de imágenes en la misma ruta que el *main*.

   - Forma incorrecta:
     `skimage.io.imread('C:/Estudiante/docs/ElProyecto/ims/im.png')`
   - Forma correcta:
     `skimage.io.imread(os.path.join('ims','im.png'))`

Informe:

Todos los laboratorios deben realizarse en formato $\LaTeX$ o Word, pueden obtener una plantilla del formato en el siguiente vínculo. Este miniproyecto consistirá en la construcción paulatina de un artículo científico, por lo cual, a diferencia de los miniproyectos anteriores, deben contener todas las secciones que conforman un documento como tal. Ver Sección **??** para más información sobre la estructura del informe. Por último, el informe tiene una longitud máxima de 5 páginas (sin incluir Referencias ni Anexos).

Bonos:

Cada pareja ganará puntos que le suben la nota por cumplir la siguiente característica:

1. Desarrollar el informe en LaTeX. Aquellas personas que lo desarrollen en LaTeX, deben escribir al final del informe **Realizado en LaTeX**. De lo contrario no se contará el bono. Los grupos que intenten reproducir la frase en un informe realizado en Word tendrán 0 en la nota de dicho proyecto. Para poder escribir el logo utilice el comando "$\backslash LaTeX$" en su informe de Latex.

Estos puntos se asignarán de acuerdo al criterio de las profesoras.

5. Definición del problema

La clasificación automática de escenas es un problema longevo en el campo de la visión por computador. Esta tarea, eje central del presente miniproyecto, consiste en la asignación de etiquetas a imágenes con escenarios naturales o urbanos según su contenido semántico. Dado este problema de clasificación y una base de datos proporcionada, el objetivo para las próximas semanas será el desarrollo paulatino de un ejercicio que cubra todos los aspectos de un proyecto de investigación en visión por computador.

Este miniproyecto condensará las etapas de justificación, experimentación, discusión y presentación de un proyecto en el área. Buscaremos el desarrollo de un algoritmo de clasificación de imágenes fotográficas en tipos de escenas naturales y urbanas. Para ello, le hacemos entrega de una base de datos con imágenes correspondientes a seis (6) tipos de escena: construcciones, bosque, glaciar, mar, montañas y calle. En la Figura 1 se expone una imagen representativa de cada clase.

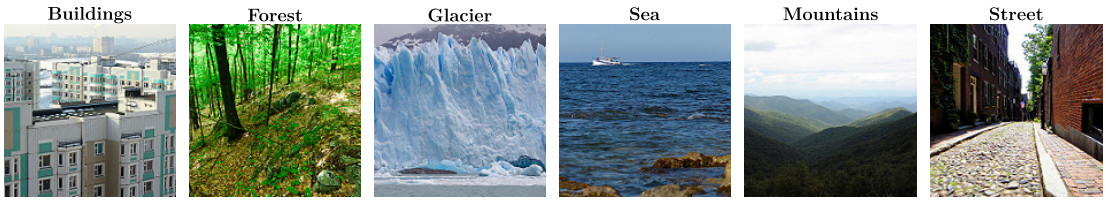

Figura 1: Ejemplos de las imágenes de cada clase dentro de la base de datos *Scene classification*.

Durante las próximas semanas, experimentarán con diferentes descriptores y clasificadores para realizar la tarea propuesta y obtener los resultados más favorables. Los descriptores considerarán aspectos como color, textura y forma, y servirán para caracterizar las imágenes en un espacio de dimensión superior. En cada uno de estos espacios, una imagen estará representada por un único punto. Esta información servirá como entrada para cuatro métodos de clasificación en dos contextos diferentes: *K-Means* (no-supervisado) y *Random Forests*, *Support Vector Machines* y *Neural Networks* (supervisado).

El objetivo principal de aprendizaje es realizar un proceso de experimentación exhaustivo sobre los parámetros de los descriptores y clasificadores, utilizando los conjuntos de entrenamiento (*train*) y validación (*val*). Los resultados de cada entrega servirán para determinar una configuración óptima para la clasificación automática de escenas en las imágenes de la base de datos. En la última entrega, harán uso de los resultados parciales de las entregas pasadas para determinar un método final, que evaluarán en el conjunto de prueba (*test*).

Archivos:

Todos los archivos del presente miniproyecto se encuentran en este vínculo. Recuerden incluir `data_mp4` como su directorio base para acceder a los documentos en su interior a través de sus códigos.

Entregables:

Según la complejidad de los algoritmos a desarrollar, los archivos a entregar variarán en cada entrega. Se recomienda hacer una lectura cuidadosa de las secciones 6.3 y 7.3 antes de realizar el envío de sus trabajos. En general, aplican las mismas reglas enunciadas en la Sección 2 sobre el formato de los archivos y su forma de entrega, con ligeras diferencias en los nombres.

6. Procedimiento Entrega 1

Para esta primera entrega, su objetivo será realizar la clasificación de las escenas con un método no supervisado. Para esto, utilizarán descriptores de color de las imágenes y *k-means* como clasificador. Para la representación de las imágenes emplearán tanto el histograma de color concatenado como el conjunto. Estas representaciones servirán de entrada para realizar una clasificación con *k-means* según su definición de clusters.

6.1. Código

En la carpeta proporcionada para este laboratorio encontrarán el *script* principal del miniproyecto (`data_mp4/main.py`). Deberán modificar este archivo para realizar la clasificación de las imágenes usando la representación de histogramas de color conjunto y concatenado, y el clustering con *k-means*. En este archivo encontrarán diferentes tareas por realizar (# TODO), sin embargo, deben tener en cuenta que estas no son las únicas modificaciones que deberán realizar para cumplir con los requisitos de la entrega. Si modifican de manera adecuada y metódica este archivo, podrán realizar más eficientemente la variación de parámetros para ejecutar sus experimentos.
Nota: Tengan en cuenta que este archivo solo lo podrán utilizar para este miniproyecto, por lo que está **prohibido** hacer uso de este archivo para su proyecto final, en otros cursos y por fuera del semestre 2021-10.

Procedimiento

1. Extraigan el archivo main.py de la carpeta `data_mp4` y modifiquen el nombre a `main_Código1_Código2.py`. Como es usual, este será el archivo que van a entregar, por lo cual deberán hacer todos sus procedimientos dentro del mismo.

2. Familiarícense con el archivo y entiendan cada una de las funciones. Fíjense en los parámetros de entrada de cada una de las funciones para que puedan hacer sus respectivas modificaciones.

3. Revisen la documentación de las funciones para realizar los histogramas de color concatenado y conjunto que encontrarán en el archivo `data_mp4/functions.py`. Por otro lado, para inicializar y entrenar un modelo de *k-means*, deben utilizar la función `sklearn.cluster.KMeans`. Asegúrense de entender su documentación antes de utilizarla.

4. Modifiquen la función `train()` y `validate()` para completar los # TODO. Para guardar y leer la matriz de descriptores son libres de escoger tanto la librería como la extensión del archivo (puede ser *.mat, .npy, .npz, .pkl,* etc.). Esto mismo aplica para guardar y leer el modelo entrenado de *k-means*.

5. Para llevar a cabo la clasificación con *k-means* deberá tener en cuenta que:

   - Deben establecer un número de clusters (k) en `parameters['k']` para inicializar su modelo. Este valor debe ser apropiado para encontrar la etiqueta que le corresponde a cada clase.
   - Para asegurar la reproducibilidad de sus modelos, deben incluir una semilla mediante el parámetro de entrada `random_state` de la función `sklearn.cluster.KMeans`.

- Las etiquetas que asigna *k-means* a sus imágenes no necesariamente corresponden con las etiquetas de sus clases originales. Por este motivo, deberán visualizar en un subplot las imágenes de entrenamiento junto con el cluster asignado por *k-means*. Esta implementación deberán hacerla dentro de la función `train()`.
- Para obtener las predicciones en la partición de validación, deberán cargar el modelo entrenado. **NO** deben volver a entrenarlo.
- Para evaluar el modelo, deberán hacer la correspondencia de las etiquetas otorgadas por k-means con las etiquetas asignadas a cada clase en las anotaciones.
- Evaluarán el desempeño de sus modelos mediante una matriz de confusión, la precisión, la cobertura y la f-medida. Para ello, deben hacer uso de las funciones nativas de `sklearn.metrics`.

6. Diseñen una manera de imprimir sus métricas de forma ordenada. En esta impresión deberíamos ver los valores de los parámetros que usaron para extraer sus descriptores y los que usaron para definir su modelo. Hagan uso del diccionario `parameters`.

7. Una vez completen todas las funciones, deben asegurarse que el código funcione. Pueden definir valores arbitrarios en el diccionario `parameters` para probar su código. De ser necesario, solucionen los errores en su código.

8. Una vez su código funcione, pueden proceder a la parte de experimentación. La idea es hacerlos de manera secuencial, por lo que solo van a variar un parámetro a la vez. En esta entrega, el diccionario `parameters` ya tiene todos los parámetros que van a cambiar para la experimentación, por lo que solo deben variar estos valores para completar sus experimentos.
   Nota: Tengan en cuenta que los valores de las llaves pueden ser funciones, strings o números enteros. Deben seleccionar cuidadosamente los valores para estos parámetros. Tengan en cuenta que los deberán justificar en el informe. Los experimentos que deben realizar son:

   - Primero, harán la clasificación de las imágenes utilizando como descriptor el histograma de color conjunto. Con este descriptor probarán dos espacios de color y dos valores para los bins del histograma.
   - Definan un espacio de color y la función para transformar a ese espacio. Les sugerimos hagan uso de la librería `skimage.color`. Otorguen valores para todos los demás parámetros y déjenlos fijos.
     Nota: Los nombres de los archivos que van a guardar deben hacer referencia al experimento que están corriendo. No hay un formato como tal, pero es importante que ustedes sepan a qué experimento corresponde cada archivo, dado que les será útil para realizar las entregas, en especial la entrega final.
   - Corran su código y asegúrense de guardar en archivos separados el modelo entrenado, la matriz de descriptores de las imágenes de entrenamiento y las de validación. Hagan un registro de las métricas obtenidas.
   - Definan otro espacio de color con su respectiva función de transformación y repitan el anterior procedimiento.

- Comparen los resultados y definan con qué espacio de color obtuvieron el mejor resultado. Fijen este valor y ahora varíen el número de bins. Corran esta configuración y obtenga las métricas en la partición de validación.
- Para este punto ya hicieron experimentos con dos números de bins y espacios de color. Ahora repitan este procedimiento, pero utilizando como descriptor el histograma de color concatenado. Igualmente, experimenten con los **mismos** espacios de color y números de bins.
- Comparen el desempeño de los diferentes descriptores y fijen la configuración con los valores que dieron el mejor desempeño.
- Finalmente, cambien el parámetro K del clasificador y dejen fijos los demás valores. Corran el experimento y obtengan las métricas.
  Ayuda: Cada vez que corran *k-means* no olviden hacer la correspondencia de las etiquetas de *k-means* con las etiquetas de las anotaciones para hacer la evaluación.

9. Al finalizar sus experimentos, guarden el mejor modelo de *k-means*, siguiendo las instrucciones de la sección 6.3.

10. Tengan en cuenta cuáles archivos de las matrices de descriptores (tanto de entrenamiento como de validación) otorgaron los mejores resultados. Si prefieren, pueden borrar los demás archivos de experimentación y guardar solamente los de la mejor experimentación. Estos archivos les ahorrarán tiempo para la última entrega.

6.2. Informe

Durante este miniproyecto construirán de forma paulatina un artículo estilo CVPR. Por lo cual, a diferencia de los miniproyectos anteriores, no deben pegar las preguntas al informe ni contestarlas individualmente. En esta entrega, el artículo debe tener el siguiente contenido:

1. **Introducción:** Contexto del problema. Deben hablar sobre la tarea (desde el punto de vista supervisado y no supervisado) y la base de datos. Incluyan estadísticas de la base de datos original y los datos proporcionados para este miniproyecto. (máx. 2 párrafos)

2. **Metodología - Histogramas de Color y K-means:** Breve descripción teórica del histograma de color concatenado y conjunto como descriptores de color de las imágenes. Para el histograma conjunto deben incluir un diagrama de cómo se obtiene el histograma conjunto.

   Pueden guiarse por las siguientes preguntas: ¿Qué es un histograma concatenado? ¿Qué es un histograma conjunto? ¿Cómo se obtienen estos histogramas? ¿Cómo se incluye el número de bins para formarlos?

   Breve descripción teórica de *k-means*.

   Recuerden que la metodología debe redactarse desde la teoría. No se trata de resumir el flujo de trabajo que realizaron en código, sino de justificar los histogramas de color como un descriptor útil para la caracterización de imágenes.

3. **Experimentos - Histogramas de Color y K-means:**

- Breve explicación de la función de los parámetros con los que experimentaron (`parameters['space'`
(`parameters['bins']` y `parameters['k']`).
- Justificación y explicación de los valores que tomaron para estos parámetros.
- Reporte completo de los resultados de clasificación en el conjunto de validación. Deben presentar una tabla con los 7 resultados de los experimentos. Esta tabla debe ser autocontenida, clara y organizada, y debe contar con un *caption* de referencia. El experimento con mejores resultados debe estar resaltado en **negrilla**.
- Breve explicación de los resultados en la tabla: ¿Qué configuración de parámetros funcionó mejor? ¿Cuál es el efecto en las métricas de evaluación de variar el histograma de color? ¿Cuál es el efecto en las métricas de evaluación de variar el espacio de color y el número de bins? ¿Cuál es el efecto en las métricas de evaluación de modificar el número de clusters de *k-means*?

4. **Discusión - Histogramas de Color y K-means:**

- Breve explicación de las diferencias entre el histograma de color conjunto y el concatenado. Desde la teoría ¿Por qué cree que una representación fue mejor que la otra?
- Explicación de los resultados obtenidos. Desde la teoría, ¿por qué considera que su mejor configuración de parámetros obtuvo los mejores resultados?
- ¿Es el color un buen descriptor para clasificar las imágenes de la base de datos? Explique las ventajas y desventajas de ambos descriptores (histogramas concatenados y conjuntos) para la base de datos utilizada.
- ¿Es la clasificación no supervisada/k-means útil en este contexto? Explique las ventajas y desventajas de este clasificador para la base de datos utilizada.
- ¿Qué cambios haría al método para obtener mejores resultados? ¿Por qué?

6.3. Entregables

Para esta entrega, ustedes deberán enviar:

- El informe del laboratorio: *Código1_Código2.pdf*.
- Un archivo con el mejor modelo entrenado de k-means (aquel con el que hayan obtenido los mejores resultados en validación en esta entrega). Puede ser un archivo de extensión *.mat*, *.npy*, *.npz*, *.pkl*, etc. y debe tener como nombre *best_model_E1_Código1_Código2*.
- Un archivo *main_Código1_Código2.py* que contenga el código base para entrenar y clasificar las imágenes de entrenamiento y validación por medio del método de los histograma de color.

  Tenga en cuenta que este archivo **NO DEBE ENTRENAR EL MODELO NI GUARDAR LOS DESCRIPTORES DE VALIDACIÓN**. Debe tener el parámetro `perform_train = False` y `action = None`. El diccionario `parameters` debe tener la configuración del mejor experimento. Al hacer esto, el archivo debe cargar el mejor modelo que se adjunta como entregable y a partir de este, clasificar las imágenes de validación de la base de datos.
- El archivo *Código1_Código2.txt* con una copia del código.

7. Procedimiento Entrega 2

En esta entrega, estudiarán una segunda y tercera combinación de herramientas para la clasificación de imágenes de escenas naturales y urbanas. En primer lugar, utilizarán el histograma de textones como descriptor para caracterizar las imágenes y el algoritmo de *Support Vector Machines* (SVM). Luego, utilizarán el histograma de gradientes orientados y el método *Random Forests*.

La idea general es modificar el script principal (`main_Código1_Código2.py`) para adaptar las funciones para el cálculo de los descriptores, entrenamiento y validación a estos nuevos contextos. Utilicen el modelo proporcionado como una guía, pues seguirán la misma lógica general. Sin embargo, tengan en cuenta que los detalles de implementación y parámetros cambiarán (en particular, noten las entradas diferentes que debe tener el diccionario `parameters`). Por otro lado, el material de trabajo para esta entrega es similar al material para la entrega anterior. Además de la base de datos, necesitarán el archivo `textons.py` y el banco de filtros dentro de `filterbank.mat`, ambos disponibles dentro del directorio base `data_mp4`.

Ya que en esta entrega se trabajará con histogramas de textones, es **extremadamente recomendable** que empiecen el laboratorio con anterioridad puesto que cada prueba puede tomar una cantidad considerable de tiempo. Por otra parte, es necesario que su computador tenga una memoria RAM mayor a 4 GB. Si usted o su compañero no lo tienen, se solicita notificar con tiempo.

7.1. Código

7.1.1. Textones y SVM

En esta ocasión, deben implementar las funciones necesarias para obtener los descriptores de textura. Para ello, deberán contar con las siguientes funciones implementadas. Estas funciones están anidadas y funcionan de manera secuencial. Se recomienda desarrollarlas en el orden propuesto y siguiendo las instrucciones dentro de cada una:

(I) `calculateFilterResponse_Código1_Código2()`

(II) `calculateTextonDictionary_Código1_Código2()`

(III) `calculateTextonHistogram_Código1_Código2()`

Procedimiento

1. Abra y explore el archivo `textons.py`. Allí encontrará enunciada una serie de pasos lógicos para desarrollar las funciones (I) y (II).

2. Copie y pegue las funciones y sus instrucciones a su script principal (`main_Código1_Código2.py`).

3. Desarrolle la función (I) siguiendo las instrucciones en el script. Tenga en cuenta las características generales de la función, descritas a continuación:

   Sintaxis:

`CalculateFilterResponse_Código1_Código2(img_gray, filters)`

Parámetros de entrada:

- `img_gray`: Una imagen en escala de grises.
- `filters`: Un banco de filtros para calcular la cross-correlación.

Respuesta:

- `resp`: Una matriz con la respuesta de la cross-correlación de la imagen con todos los filtros. Esta matriz es de dimensión $(M \times N) \times \#filtros$, donde M y N son las dimensiones de la imagen de entrada.

4. Desarrolle la función (II). Esta función calculará el diccionario de textones a partir de las respuestas de las imágenes de entrenamiento a la cross-correlación con el banco de filtros. Las características generales de esta función son:

   Sintaxis:

   `CalculateTextonDictionary_Código1_Código2(images_train, filters, parameters)`

   Parámetros de entrada:

   - `images_train`: Una lista con las imágenes de entrenamiento.
   - `filters`: Un banco de filtros para calcular la cross-correlación.
   - `parameters['k']`: El tamaño del diccionario de textones (el número de clusters para K-Means).
   - `parameters['dict_name']`: El nombre del archivo *.mat* donde va guardar el diccionario de textones.

5. Realice la implementación de la función (III). Esta función hará uso del diccionario de textones obtenido con la función (II) para obtener el histograma de textones de una imagen. Deberá emplear la distancia 'L2' para asignar a cada pixel de la imagen de entrada, el textón más cercano. La función debe tener las siguientes características:

   Sintaxis:

   `CalculateTextonHistogram_Código1_Código2(img_gray, centroids)`

   Parámetros de entrada:

   - `img_gray`: Una imagen en escala de grises.
   - `centroids`: Los centroides del diccionario de textones.

   Respuesta:

   - `hist`: El histograma de textones de la imagen de entrada.

6. Ahora utilizarán las funciones (I), (II) y (III) para obtener los histogramas de textones de un conjunto de datos. Para ello, deben llamar estas implementaciones dentro de la función `calculate_descriptors()` en su script principal y realizar algunas modificaciones adicionales. Al finalizar su edición, la nueva función de obtención de descriptores debe tener las siguientes características:

   Sintaxis:

   ```
   calculate_descriptors(data, parameters, calculate_dict)
   ```

   Parámetros de entrada:

   - `data`: Un conjunto de imágenes.
   - `parameters`: El diccionario de parámetros de experimentación.
   - `calculate_dict`: Un parámetro booleano que controle la creación de un diccionario de textones. Al tomar el valor `True`, la función de `calculate_descriptors()` debe hacer uso de las funciones (I) y (II) para calcular y almacenar de forma local un diccionario de textones determinado. En el caso contrario, la función debe cargar un diccionario ya almacenado y utilizarlo para calcular los histogramas de textones del conjunto de imágenes de entrada. Es decir, solo hará uso de las funciones (I) y (III). Recuerde que el nombre del diccionario estará determinado por el parámetro `parameters['dict_name']` y debe tener la extensión *.mat*.

   Respuesta:

   - `descriptor_matrix`: Un arreglo con los histogramas de textones de las imágenes de entrada.

   Nota: Noten que, aunque modifiquen la función de la entrega anterior, estarán siguiendo la misma lógica. El objetivo de obtener un arreglo con los descriptores de las imágenes de entrada se mantiene, pero se cambian los histogramas de color por los histogramas de textones.

7. Modifiquen la función `train()` para entrenar un modelo de *Support Vector Machines* (SVM) utilizando los histogramas de textones como descriptores de las imágenes. Al inicializar el modelo, deben establecer su parámetro 'kernel' igual a `parameters['kernel']`. Conserven la(s) línea(s) que implementaron previamente para guardar el modelo con el nombre dado por el parámetro `parameters['model_name']`.

8. Verifiquen la función `validate()` y asegúrense que funcione para su nueva combinación de descriptor y clasificador.

9. Por último, verifique el diccionario `parameters` y, si no lo han hecho anteriormente, modifiquen sus entradas para ajustar los parámetros de experimentación a su nuevo descriptor y clasificador. Descarte las entradas que hagan referencia a la entrega anterior. **Ojo:** No deben eliminar las entradas `'model_name'`, `'train_descriptor_name'` ni `'val_descriptor_name'` pues las necesitarán para poder almacenar sus descriptores y modelos entrenados.

10. Realicen 2 experimentos variando el parámetro `parameters['k']`. Recuerde que este valor se refiere al tamaño del diccionario de textones utilizado para la construcción de sus descriptores.

    - Para estos experimentos, la variable de control será el parámetro `parameters['kernel']` del clasificador. Deben elegir un valor para este parámetro según la documentación oficial del clasificador y mantenerlo fijo durante estos experimentos.

- Para la realización de estos experimentos, establezcan el valor el parámetro `calculate_dict` igual a `True`.
- Recuerden establecer un nombre al diccionario de textones y descriptores de entrenamiento y validación, que hagan referencia al experimento. Esto será fundamental para evitar confusiones en adelante y mantener el orden en sus archivos.

11. Repita los experimentos anteriores, ahora fijando el parámetro `parameters['kernel']` igual a la función `GeneralizedHistogramIntersection()` definida dentro del archivo `pykernels/regular.py`.

    - Para importar la función `GeneralizedHistogramIntersection()` dentro de su *script* principal, utilice el siguiente comando:

      ```
      from data_mp4.pykernels.regular import GeneralizedHistogramIntersection()
      ```

      Esta implementación pertenece a la librería *pykernels*, realizada por Wojciech Marian Czarnecki y Katarzyna Janocha. La fuente oficial se encuentra en el siguiente vínculo.
    - Para estos experimentos, contarán con dos diccionarios de textones creados en el numeral anterior (uno para cada valor de `parameters['k']`). Por tanto, deben establecer su parámetro `calculate_dict` igual a `False` y asegurarse de cargar el diccionario que corresponda al experimento que realizarán.

12. Al finalizar este procedimiento, tendrá resultados cuantitativos para 4 experimentos. Contará con un registro de métricas para poder reportar en el informe (Sección **Experimentos - Textones y SVM**).

7.1.2. HOG-RF

1. Modifiquen las funciones del *script* principal para obtener los descriptores de forma y el modelo de clasificación por RF. Para RF utilicen la implementación `RandomForestClassifier` de `sklearn.ensemble` y para HOG la función de `skimage.feature.hog`. Lean detenidamente ambas documentaciones y familiarícense con los parámetros de entrada. Asegúrense de comprender su significado desde la teoría.

2. Para el conjunto de experimentos HOG-RF, son libres de escoger dos parámetros de HOG y dos parámetros de RF. Modifiquen las llaves del diccionario `parameters` para que incluyan estos parámetros.

3. Igual que en las entregas pasadas, deben realizar sus experimentos de manera secuencial, fijando primero el clasificador y luego el descriptor. Por cada parámetro, deben completar experimentos con 2 valores diferentes. Como en la entrega pasada, deben obtener los resultados en validación y guardar de forma local, la mejor configuración de descriptores y modelo.

4. Al finalizar este procedimiento, tendrá resultados cuantitativos para 6 experimentos. Contará con un registro de métricas para poder reportar en el informe (Sección **Experimentos - HOG y RF**).

7.1.3. Método Final

En esta sección, utilizarán la mejor configuración de descriptores y clasificadores desarrollados hasta el momento, para obtener un método final. Realizarán la combinación de todos los descriptores con todos los clasificadores (equivalente a 9 experimentos). Para este punto, ya habrán corrido 3 de ellos (Color-K-Means, Textones-SVM y HOG-RF). Ahora, deberán correr las 6 combinaciones que restan.

NO deben experimentar con los parámetros de los descriptores/clasificadores. Deben únicamente obtener las métricas de validación con cada combinación. Por este motivo, van a escoger la configuración con los mejores resultados de las secciones anteriores. Esto implica utilizar los archivos con los descriptores de color, textura y forma con mejor desempeño y con cada uno, entrenar un modelo de K-Means, SVM y RF con los mejores parámetros determinados. A continuación, encuentran un paso a paso:

1. Creen un nuevo archivo llamado `exps_Código1_Código2.py`.
2. Para este archivo, no necesitarán una función que extraiga los descriptores, pues harán uso de los archivos que guardaron en esta y la entrega anterior. Por cada descriptor, deberían tener la matriz de descriptores, tanto para entrenamiento como para validación, con la configuración de mejor desempeño.
3. Carguen las matrices de descriptores de entrenamiento y de validación de cada descriptor.
4. Realicen las combinaciones descriptor-clasificador que restan. Entrenen los clasificadores con las matrices de descriptores de entrenamiento y obtengan las predicciones con las matrices de descriptores de validación.
5. Con las predicciones y las anotaciones que tienen disponibles, calculen el desempeño de clasificación de cada combinación.
6. Al finalizar este proceso, contarán con resultados cuantitativos para 9 métodos diferentes. Este registro de métricas debe ser reportado en el informe (Sección **Experimentos - Método Final**. Recuerde que estos resultados se obtienen a partir de las imágenes de validación y les permitirán seleccionar el método con mejor desempeño (su método final).
7. Guarden este modelo con el nombre final_model_Código1_Código2.mat (o la extensión de su elección).
8. Creen una función llamada `test()` que permita extraer una matriz de descriptores, usando el descriptor de su método final. Cargue el mejor modelo y obtenga las predicciones sobre los datos de prueba (*test*). Esta función debe evaluar el método en este conjunto e imprimir de forma organizada los resultados (incluyendo las métricas, el nombre del descriptor, el nombre del clasificador y los valores de los parámetros de cada uno).
9. Al hacer su entrega, esta es la única función del código que debe correr. Haga el uso de booleanos como en el ejemplo del archivo que les dimos, en vez de comentar la parte de experimentación.

7.2. Informe

En esta entrega reportarán los resultados de los experimentos realizados en las secciones 7.1.1, 7.1.2 y 7.1.3. Para cada conjunto de experimentos, deben agregar una subsección correspondiente dentro de la sección **Experimentos**, que tenga un reporte completo de los resultados de clasificación en el conjunto de validación. En cada subsección, deben presentar una tabla con los resultados de los experimentos. Esta tabla debe ser autocontenida, clara y organizada, y debe contar con un *caption* de referencia. El experimento con mejores resultados debe estar resaltado en **negrilla**. Al final de estas secciones, enuncien los resultados del método final en el conjunto de prueba.

Deben realizar una discusión final que resuma su proceso de experimentación con todos los descriptores y clasificadores vistos en las últimas semanas. Para ello, utilice como guía las siguientes preguntas:

- En el contexto del laboratorio y la base de datos empleada, explique las ventajas y desventajas de los descriptores y clasificadores utilizados.
- ¿Cuál fue la combinación descriptor-clasificador con mejores resultados en validación? Explique las razones por las que considera que este método funciona mejor para este problema.
- ¿Cómo se comparan los resultados en el conjunto de validación frente a los resultados de prueba? Explique las razones por las que estas difieren o podrían diferir.
- ¿Qué limitaciones tiene el método propuesto? ¿Qué características de la base de datos dificultan la tarea?
- De acuerdo con las características de la base de datos, ¿qué mejorías podrían realizar en las metodologías empleadas para mejorar sus resultados?

7.3. Entregables

Para este laboratorio deberán entregar:

- El informe del laboratorio: *Código1_Código2.pdf*
- El archivo de código principal (`main...`) configurado para entrenar su mejor modelo de **Textones-SVM**. Recuerden que este archivo debe incluir las implementaciones (I), (II) y (III) de la Sección 7.1.1.
- El archivo de código donde corrieron todos los experimentos y hacen la evaluación del método final: *exps_Código1_Código2.py*

  En contraste con el ítem anterior, este archivo **NO DEBE ENTRENAR EL MODELO NI GUARDAR NINGÚN DESCRIPTOR**. Debe estar configurado para solo mostrar la evaluación en el conjunto de prueba cuando se clasifica con el mejor método.
- El archivo con el modelo final entrenado (un archivo .mat, .npy, .npz, .pkl, *etc.*): *final_model_Código1_Código2.npy*, *etc.*
- Un archivo de texto plano con **TODOS** los códigos del laboratorio: *Código1_Código2.txt*.