Tarea 3

Segmentación de Imágenes

Profesor: Claudio Pérez F.
Auxiliar: Diego Maureira
Ayudantes: Juan Pablo Pérez, Jhon Pilataxi, Jorge Zambrano

1. Objetivo

El objetivo de esta tarea es comparar el desempeño de un enfoque tradicional de segmentación de imágenes basado en filtros y operaciones morfológicas, con otro enfoque basado en el uso de Redes Neuronales Convolucionales (CNN), evaluando la calidad de los resultados y las ventajas/desventajas de cada método.

2. Descripción

La segmentación de imágenes es una tarea fundamental en el campo de la visión computacional, ya que permite dividir una imagen en sus distintas regiones o componentes relevantes, facilitando el análisis y procesamiento posterior. Existen diversos enfoques para realizar la segmentación, los que van desde métodos clásicos basados en técnicas de procesamiento digital de imágenes, hasta enfoques más recientes basados en CNNs o arquitecturas de *Transformers*.

En esta experiencia, se compararán dos enfoques diferentes para la segmentación: uno basado en un método manual utilizando filtros y operaciones morfológicas, y otro mediante una CNN. La primera actividad les permitirá entender con mayor profundidad cómo funcionan las técnicas tradicionales, ya que tendrán que mejorar el algoritmo de segmentación creado en la Tarea 2. En contraste, con el segundo enfoque, entrenarán una CNN y evaluarán su capacidad para realizar segmentación de forma automática, lo que les permitirá observar la diferencia en la eficacia y la eficiencia de ambos métodos.

3. Mejora de algoritmo *handcrafted* de segmentación de grietas

En esta primera actividad usted utilizará como base el algoritmo de segmentación de grietas en concreto que ya implementó en la Tarea 2 del curso. En aquella experiencia utilizó un filtro Canny para la detección de bordes, y luego, generó la segmentación de las zonas

agrietadas en la imagen mediante la aplicación de operaciones morfológicas.

El objetivo de esta sección es generar mejoras en la forma de segmentación de su algoritmo, utilizando todo lo que ha aprendido en el curso hasta el momento. Como base, puede utilizar un notebook de Colab que quedará disponible en el siguiente link: Tarea3Tutorial_Handcrafted. Recuerde generar una copia del notebook en su drive (File -> Save a copy in Drive) y **no** ejecutar directamente el del link.

A continuación se les dará una lista de posibles formas de mejora de su algoritmo, puede elegir una o más de ellas y aplicarlas, o incluso, investigar y elegir otras alternativas.

1. **Ajustar el tipo e intensidad del suavizado**: Existen muchos tipos de suavizado de imágenes y al utilizar cada uno de ellos también se puede ajustar la intensidad con la que se aplica. Todas estas variaciones provocan cambios en los resultados obtenidos por los algoritmos. Este link le podría ser de utilidad para revisar distintos tipos de suavizados con ejemplos de uso.

2. **Agregar o modificar la umbralización de las imágenes**: Usted podría incluso no utilizar detección de bordes en su algoritmo y reemplazarla por un método de umbralización suficientemente bien ajustado posterior al suavizado de la imagen. Este link le ayudará a expandir la gama de posibles formas de ejecutar una umbralización sobre una imagen.

3. **Refinar parámetros de detección de bordes**: Si opta por mantener la detección de bordes, un aspecto clave en el cual fijarse son los umbrales de Canny, debido a que pueden afectar significativamente los resultados. Experimente con diferentes valores para encontrar los que mejor se adapten a las imágenes.

4. **Mejorar operaciones morfológicas**: Tanto si utiliza la umbralización como la detección de bordes (o incluso, una combinación de ambas), ajustar el tamaño o la forma del elemento estructurante para erosionar y/o cerrar mejor las grietas, se transforma en un proceso de alta importancia. En este link encontrará más información sobre estos métodos de OpenCV.

5. **Filtrar contornos**: Además de utilizar operaciones morfologicas, se podría implementar también un filtrado de contornos mediante el área que posee cada uno. Para esto, primero deberá identificar cada contorno en su máscara tentativa, para esto se puede apoyar de la siguiente documentación. Una vez obtenidos los contornos, se podrían filtrar por el área que abarcan en la imagen o, en lugar de usar un área fija como umbral, considerar analizar la forma del contorno o usar criterios de filtrado más sofisticados para determinar si es una grieta o no.

6. **Hacer una búsqueda de los mejores parámetros de las funciones usadas**: Para esto se puede apoyar de metodologías de optimización de parámetros, tales como la búsqueda de grilla o cualquier otra.

Finalmente, reporte el *mean Accuracy* y el *mean IoU* en el conjunto de *test_small* y el tiempo de inferencia promedio por imagen, tanto de su algoritmo original como de su nueva versión. No se preocupe si es que sus métricas no mejoran, se evaluará con mayor ponderación la metodología seguida y la argumentación de cada una de las decisiones tomadas por usted.

4. Entrenamiento de una CNN

El entrenamiento de una CNN es un proceso clave en el aprendizaje profundo, ya que permite a la red aprender a identificar patrones y características específicas en las imágenes, lo que la capacita para realizar tareas como la segmentación de manera automática. A diferencia de los métodos tradicionales de segmentación, que dependen de filtros y reglas predefinidas, las CNNs aprenden de los datos, a través de la optimización de sus parámetros (llamados también pesos o *weights*) mediante el uso del algoritmo de retropropagación (*back-propagation*) y algún optimizador. Durante el entrenamiento, la red ajusta sus parámetros en función de los ejemplos que le son presentados, minimizando una función de pérdida que cuantifica la diferencia entre las predicciones de la red y las etiquetas reales (conjunto que es llamado *ground truth*).

En este apartado, ustedes entrenarán una CNN en un framework especializado en la segmentación semántica de imágenes, llamado MMSegmentation. Para esto, usarán las mismas imágenes de grietas en concreto utilizadas en la Tarea 2, pero esta vez un subconjunto más amplio (*test_small*). Al final de esta sección, se espera que cada uno de ustedes obtenga una red entrenada, capaz de segmentar grietas de la forma más robusta posible sobre nuevas imágenes y que sean capaces de evaluar su rendimiento utilizando métricas estándar como el *Accuracy* y el *IoU*. Además, visualizarán y analizarán las salidas intermedias o *feature maps* de la red entrenada.

Para cumplir los objetivos anteriores, deberá seguir los siguientes pasos:

1. Clone el archivo de Tarea3Tutorial_DL (File -> Save a copy in Drive) que corresponde a un tutorial de uso de MMSegmentation adaptado por el equipo docente para esta experiencia.

2. Ejecute el punto 1 y 2. Explique que significa la información que le entrega la salida del comando *nvidia-smi*.

3. Ejecute el punto 3.

4. Abra el link hacia la base de datos que utilizará: crack_segmentation_dataset. Haga un *shortcut* de esta base en su Drive (Organize -> Add shortcut -> All locations -> My Drive o donde quiera). Este paso es necesario debido a que originalmente sería difícil acceder desde Colab a una carpeta de Drive que fue compartida con usted.

5. Ejecute el punto 4. Esto iniciará el montaje de su cuenta de Drive en el notebook de Colab. Gracias a esto podrá acceder directamente a la base de datos del punto anterior.

6. Ejecute el punto 5. wandb es una poderosa herramienta que permite la visualización de las curvas de entrenamiento y estadísticas de evaluación de los modelos y, además, da la posibilidad de observar los feature maps obtenidos en inferencia. Siga las instrucciones de las líneas que se presentan en pantalla, en específico este es un paso clave:

```
wandb: You can find your API key in your browser here: https://wandb.ai/authorize

```

   Para completar correctamente este paso tendrá que crearse una cuenta en la página.

7. Lea con atención cada celda y comentario del punto 6. Cada línea de código en este punto define algún funcionamiento del futuro entrenamiento de su modelo, incluida la arquitectura del mismo. La celda en donde se define el dataset (CrackDataset) se puede ejecutar sólo una vez sin que arroje error, en otro caso recibirá el siguiente mensaje:

```
KeyError: 'CrackDataset is already registered in dataset at ___main___'

```

   Si esto pasa, no se preocupe y siga ejecutando el resto del celdas.

   Por otro lado, debe ajustar a su conveniencia la cantidad de iteraciones máximas que entrenará su modelo, para esto tiene que modificar la siguiente línea de código

```
cfg.train_cfg.max_iters = cfg.train_cfg.val_interval * num_epochs

```

   Donde la variable *num_epochs* representa la cantidad de vueltas al conjunto de entrenamiento a las que se expondrá el modelo. En general un buen número es *num_epochs* = 12, pero debido a la limitación de entrenamiento de Colab podría disminuirlo (ojalá no sean menos de 5 épocas).

8. Ejecute el punto 7 y deje entrenar su modelo. Por cada fase de evaluación aparecerá en pantalla un resumen de las métricas de interés por clase y promedio. Además se explicitará si el *checkpoint* de esa evaluación superó a alguno anterior y, en ese caso, se guardará como la nueva mejor configuración de pesos. Por otro lado, todas estas estísticas serán mostradas también en la página de su proyecto en la aplicación de wandb. Comente que observa en la evolución de las métricas.

9. Al finalizar el entrenamiento, el mejor checkpoint estará guardado en el *path* definido, y el nombre del mismo indicará la iteración de evaluación donde se logró el máximo rendimiento ¿Cuáles son las métricas de validación del mejor modelo obtenido? ¿En que iteración se dió el máximo rendimiento del modelo, coincide con la última iteración? Argumente por qué.

10. Ejecute el punto 8 ¿Cómo es visualmente la segmentación obtenida?

11. Copie el archivo *feature_map_visual.py* entregado con la tarea, en la carpeta de MMSegmentation y ejecute el punto 9. Luego de que termine la ejecución, vaya a su página principal de wandb, seleccione el proyecto y observe la visualización de las 4 salidas de la red utilizada ¿Qué característica tiene cada una de las salidas?

5. Actividad Final: Comparación de rendimiento

En este sección se hará el análisis y comparación de los resultados obtenidos por los dos tipos de modelos implementados por usted, para esto deberá seguir los siguientes pasos:

1. Entregue una tabla que compare las métricas de Acc, mIoU y tiempo de inferencia para su modelo *handcrafted* original, modificado y la red convolucional entrenada. Comente respecto a las diferencias de los valores obtenidos.

2. Entregue visualizaciones de la segmentación hecha por los modelos para algunos ejemplos que encuentre significativos. Comente sobre las diferentes cualidades de segmentación.

3. Concluya sobre cuál es el mejor modelo a su juicio y por qué.

6. Entregables

- Presente un reporte del trabajo. Incluya en el informe todo lo solicitado en el enunciado de la tarea.

- El código también debe ser entregado, asegúrese de que pueda ser ejecutado por los revisores para comprobar el algoritmo.

7. Recordatorio

- El reporte es individual.

La fecha de entrega de trabajo será el día 24/10/2024 a las 23:59 hrs por medio de U-cursos.