UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DESARROLLO DE UN MÉTODO PARA SEGMENTACIÓN DE FRAGMENTOS DE ROCAS EN IMÁGENES DIGITALES, USANDO REDES NEURONALES CONVOLUCIONALES Y AUMENTACIÓN DE DATOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

EITAN ARIEL HASSON ARELLANO

PROFESOR GUÍA:
CLAUDIO PÉREZ FLORES

MIEMBROS DE LA COMISIÓN:
ANDRÉS CABA RUTTE
FRANCISCO GALDAMES GRUNBERG

Este trabajo ha sido parcialmente financiado por:
FONDEF ID21I10172
FONDECYT 1191610, ANID
Departamento de Ingeniería Eléctrica de la Universidad de Chile
Centro basal AMTC AFB 220002, ANID

SANTIAGO DE CHILE
2023

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: EITAN ARIEL HASSON ARELLANO
FECHA: 2023
PROF. GUÍA: CLAUDIO PÉREZ FLORES

DESARROLLO DE UN MÉTODO PARA SEGMENTACIÓN DE FRAGMENTOS DE ROCAS EN IMÁGENES DIGITALES, USANDO REDES NEURONALES CONVOLUCIONALES Y AUMENTACIÓN DE DATOS

En el proceso minero es importante conocer la granulometría de las rocas extraídas para saber la calidad de la fragmentación realizada, de esta manera se podrá continuar realizando los siguientes pasos sin inconvenientes ni gastos excesivos de energía, material o dinero. Se desea extraer rocas de un tamaño óptimo para el transporte y procesamiento de los diferentes fragmentos para la obtención final del metal. Hasta hace algunos años no se utilizaban métodos de Inteligencia Artificial para realizar este trabajo, sin embargo los avances realizados en esa área permiten estudiar su uso para medir y evaluar el resultado de la fragmentación del material en los puntos de extracción.

El objetivo principal del trabajo consiste en el desarrollo de un método que sea capaz de detectar y segmentar rocas en una imagen digital, mediante el uso de redes neuronales convolucionales. Además, deberá ser capaz de hacerlo para imágenes que contengan una gran cantidad de rocas de diferentes tamaños dentro de una faena minera. Adicionalmente, se buscará desarrollar y evaluar diferentes metodologías que logren mejorar el desempeño de la segmentación a realizar.

Para cumplir lo anterior se utiliza una base de datos disponible en internet y una red neuronal convolucional de código abierto, llamada Detectron2. Esta red permite modificar varios de sus hiperparámetros, además de una fácil implementación de aumentación de datos. La metodología elegida permite obtener resultados en un formato fácil de evaluar (formato COCO). Así es como se logran probar diferentes métodos de aumentación de datos para ver si los resultados obtenidos mejoran.

Se realizaron múltiples pruebas, entre ellas, la mejor aumentación de datos para utilizar durante el entrenamiento corresponde al que realiza recortes en posiciones aleatorias en una imagen de entrada, ajustado a un rango de tamaño de corte asignado. Este modelo entrega un resultado de precisión media promedio en segmentación cercana al 50 % usando el evaluador COCO, y un resultado de precisión total superior al 80 %, utilizando una evaluación menos exigente implementada durante el desarrollo del trabajo de título.

Finalmente, se considera que se logró cumplir con el objetivo principal, ya que se consigue segmentar satisfactoriamente una gran cantidad de rocas. Gracias a ello, las mineras podrían ser capaces de obtener de forma rápida y eficiente la segmentación de rocas en una imagen, logrando medir los tamaños de cada una. De todas formas el método aún puede mejorar, pues existen más procedimientos de aumentación de datos, hiperparámetros y otros detalles que no se pudieron probar debido a limitaciones de hardware.

Esta memoria es mejor que...

Agradecimientos

Primero que todo, me gustaría agradecer a mi familia. A mi padre, Mario, por seguir trabajando y esforzándose para darme toda mi educación y por facilitarme la introducción al área que sería de mi interés. A mi madre, Rossana, que siempre ha estado ahí para mí, buscando lo mejor para todos. También a mis hermanos, David y Yael, quienes siempre me han ayudado cuando lo necesito. Muchas gracias por ser mi familia y por todo el cariño que me han entregado estos años.

Quiero agradecer a mi profesor, Claudio Pérez, por darme la oportunidad de trabajar con él y por la guía que me entregó durante los últimos años. Muchas gracias a Juan Pablo, quien estuvo siempre disponible para ayudarme y responder todas las dudas que surgieron mientras estuve trabajando con ambos.

Finalmente, quiero agradecer a mis amigos y amigas, especialmente a la Consu por su ayuda. Les agradezco todo lo que me han hecho disfrutar y reír durante los últimos años, tanto fuera como dentro de la universidad. No creo haber podido conocer mejores personas.

Esta memoria de título fue parcialmente financiada por los proyectos FONDEF ID21I10172, y FONDECYT 1191610, ANID, por el Departamento de Ingeniería Eléctrica de la Universidad de Chile y por el centro basal AMTC AFB 220002, ANID.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Descripción del Problema

En el procesamiento de minerales existen varias etapas que van desde la fragmentación de rocas hasta la extracción de minerales. Cada proceso tiene sus propias dificultades, dependencias y costos que hay que tener en cuenta. Algunos de estos procesos, como el transporte, el chancado y la molienda, dependen fuertemente del tamaño de los fragmentos de rocas obtenidos, por lo que, al realizar la tronadura, se busca extraer fragmentos de tamaño óptimo [1, 2]. De esta manera se podrán manipular más fácilmente los fragmentos durante los subsiguientes procesos mineros, evitando la utilización de más energía o tiempo. De ahí que las mineras busquen estimar el tamaño de las rocas extraídas, pues así se sabrá la calidad de la fragmentación realizada, permitiendo implementar optimizaciones y eludir el uso de más energía o tiempo, lo que a su vez logra prevenir el aumento de los costos originales [1–3].

Para poder estimar el tamaño de los fragmentos existen diversos métodos manuales que pueden resultar imprecisos o tomar mucho tiempo [4]. Por otro lado, existen métodos más eficaces, basados en el análisis digital de imágenes. Sin embargo, algunos de estos métodos requieren de un experto que los ponga en funcionamiento, otros necesitan mucho tiempo para ser implementados, pues se basan en una segmentación manual realizada por un experto, etiquetando el contorno de cada roca dentro de una imagen, la cual puede tener mala resolución, baja luminosidad, múltiples rocas superpuestas, entre otros problemas que retrasan el trabajo de segmentación. Es por eso que es necesario utilizar una metodología diferente, la cual sea más veloz y capaz de segmentar rocas, pudiendo así conocer la distribución de tamaño de las que haya dentro de una imagen digital.

En los últimos años la Inteligencia Computacional ha sido estudiada y aplicada a múltiples tareas basadas en la visión computacional, como lo son la clasificación de imágenes, localización de objetos y/o segmentación de los mismos. Estos estudios han logrado buenos resultados, y son constantemente mejorados. A pesar del potencial que se ha mencionado, no existen muchos estudios que analicen la segmentación de rocas mediante la utilización de redes neuronales convolucionales (CNNs) [5]. Si bien hay trabajos que investigan sobre la detección y/u obtención de tamaños de rocas dentro de imágenes digitales con CNNs, se considera que están poco relacionados con el objetivo del presente trabajo de título, por ejemplo, algunos de estos trabajos segmentan granos de minerales microscópicos en una roca. Existen otros artículos más actuales, que si implementan la segmentación de cada instancia

de roca, pero la implementan en imágenes con rocas pequeñas, lo cual es insuficiente para segmentar cada roca en una imagen dentro de un entorno minero.

1.2. Objetivos

1.2.1. Objetivo General

El objetivo principal de esta memoria de título consiste en el desarrollo de un método que sea capaz de detectar y segmentar rocas de múltiples tamaños en imágenes digitales. El método debe ser capaz de realizar la tarea mencionada en ambientes no controlados, como lo son dentro de una faena minera. Se desarrollará con CNNs para evaluar la aplicación de esta tecnología que ha tenido muy buenos resultados en otros ámbitos.

1.2.2. Objetivos Específicos

1. Desarrollar un método de segmentación de rocas para varias escalas de resolución, aplicable a imágenes tomadas a cielo abierto en ambiente no controlado.

2. Explorar combinaciones de hiperparámetros de la CNN para mejorar el desempeño de la segmentación de fragmentos de roca.

3. Desarrollar y aplicar métodos de aumentación de datos para mejorar el desempeño de la segmentación en base a CNNs.

Capítulo 2

Marco Teórico

Durante el desarrollo de esta memoria de título se utilizarán términos asociados a la inteligencia computacional. Es por ello importante describirlos con mayor detalle en la presente sección, para así, una vez se comente respecto a la metodología utilizada, se pueda sólo centrar en explicar la misma.

2.1. Red Neuronal Convolucional (CNN)

Las redes neuronales convolucionales (CNNs) son redes neuronales especializadas, las cuales, en vez de utilizar multiplicación de matrices, utilizan, en al menos una de sus capas, la convolución [6, 7]. Estas redes se utilizan para procesar datos que presentan una estructura cuadriculada, como datos en series de tiempo o imágenes digitales [8]. En la actualidad este tipo de red se ha desarrollado bastante y se utiliza para resolver un gran número de problemas complejos, como lo son la clasificación y/o segmentación de objetos en imágenes y vídeos, el desarrollo de automóviles autónomos, el reconocimiento de voz y diversas aplicaciones dentro de la medicina. Las CNNs pueden verse construidas por una gran cantidad de capas, sin embargo, las más importantes corresponden a las capas convolucionales y las capas de "pooling". A continuación se explicarán de manera general.

2.1.1. Convolución

Previo a explicar las capas convolucionales es importante comentar sobre la operación matemática conocida como convolución. De manera general, la convolución es una operación entre dos funciones que indica la cantidad de superposición entre una función $g(t)$ mientras ésta es movida hacia la segunda función $f(t)$, es decir, corresponde al área del producto entre dos funciones [8]. La operación es la siguiente:

$$f(t) * g(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \tag{2.1}$$

Ecuación 2.1: Operación para calcular la convolución. Ecuación extraída de [8].

Sin embargo, las funciones digitales no son continuas, es por ello que existe la ecuación 2.2, la cual es una discretización de la ecuación 2.1.

$$f(n) * g(n) = \sum_{k=-\infty}^{\infty} f(k)g(n-k) \tag{2.2}$$

Ecuación 2.2: Operación para calcular la convolución con funciones discretas. Ecuación extraída de [8].

Finalmente, debido a que también existen funciones con más de una dimensión, es importante destacar la ecuación 2.3, la cual representa la convolución discreta para 2 dimensiones. Gracias a la propiedad conmutativa [8] de la convolución se pueden utilizar las 2 versiones presentes en la ecuación.

$$f(m,n)*g(m,n) = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} f(j,k)g(m-j,n-k) = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} f(m-j,n-k)g(j,k) \tag{2.3}$$

Ecuación 2.3: Operación para calcular la convolución en funciones discretas con 2 dimensiones. Ecuación extraída de [8].

La estructura de esta ecuación se puede seguir utilizando si se desea aumentar las dimensiones, simplemente se deberán agregar las respectivas sumatorias y variables de acuerdo a las dimensiones añadidas.

2.1.2. Capas Convolucionales

Las capas convolucionales usualmente utilizan una variación de la segunda parte de la ecuación 2.3 conocida como "cross correlation" [8] (ecuación 2.4). Sin embargo, por comodidad, se le llama como la convolución.

$$f(m,n) * g(m,n) = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} f(m+j,n+k)g(j,k) \tag{2.4}$$

Ecuación 2.4: Operación para calcular la "cross correlation". Ecuación extraída de [8].

En el caso de las capas convolucionales, la función $f(m,n)$ corresponde a la imagen de entrada y la función $g(m,n)$ a un filtro, conocido como "kernel" [8]. Es así como el proceso dentro de una capa convolucional corresponde al "kernel" desplazándose por la imagen. Durante el desplazamiento se van calculando los valores de la convolución. Así se van generando una o más "imagenes" de menor tamaño, mejor conocidas como "feature map" [8]. El funcionamiento general se puede resumir en que, cada neurona de la primera capa convolucional se conecta sólo a los píxeles con los que se calcula la convolución, designados como campo receptivo. Este proceso se repite de la misma manera con las siguientes capas, cada píxel de una capa se conecta a un pequeño rectángulo dentro de la capa anterior. Esto hace que la red se logre centrar en características pequeñas de bajo nivel, generando así características de mayor nivel en una nueva imagen [6]. Lo más básico de este proceso se puede observar en la figura 2.1 [6].

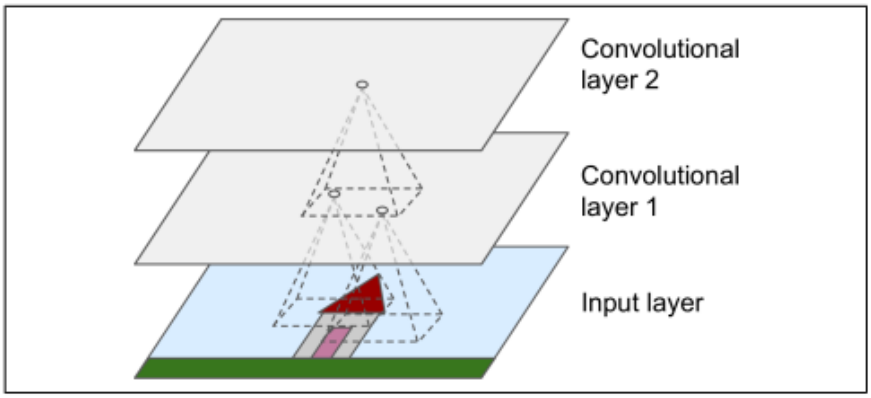
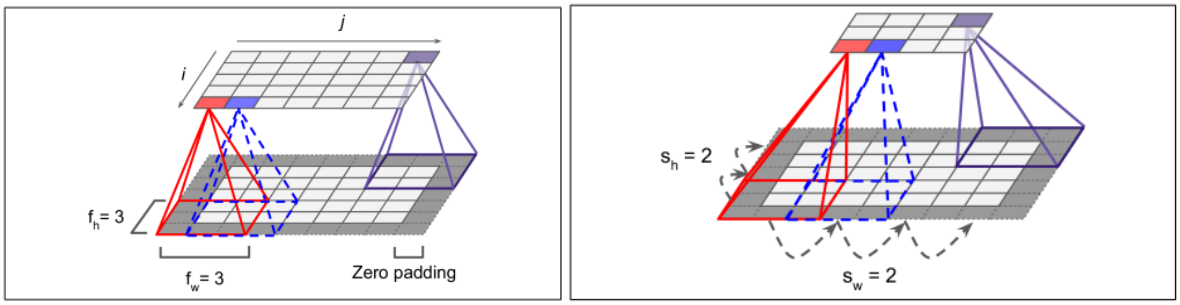


Figura 2.1: Capas Convolucionales con campos receptivos rectangulares. Se puede observar como entre capas cada píxel abarca sólo un área rectangular y, a medida que se profundiza en la red, las características de la imagen original se van detallando más en un sólo píxel. Figura extraída de [6].

Las operaciones realizadas durante la traslación del "kernel" que genera los "feature map" pueden ser modificadas por diversos métodos. Una de ellas se conoce como "padding", observable en la figura 2.2(a) [6]. Esta operación se utiliza para mantener el mismo tamaño entre cada capa, agregando ceros en la entrada o extendiendo los límites de la misma [6, 7]. Otro método popular corresponde al "stride", observable en la figura 2.2(b) [6], que consiste en espaciar los campos receptivos que se generan al realizar la convolución, y logra reducir la complejidad del modelo [6, 7].



(a) "Padding" (b) "Stride"

Figura 2.2: En (a) se observa una representación de "padding" al realizar la convolución, en específico se están considerando los píxeles fuera de la imagen que caen dentro del kernel como 0, de esta manera la salida resulta del mismo tamaño que la entrada. En (b) se observa la representación del "stride", en la cual las flechas indican los "saltos" que da el "kernel" durante la realización de los cálculos de la convolución, de esta manera se reduce el tamaño de la salida. Ambas figuras fueron extraídas de [6].

2.1.3. Capas de "pooling"

El segundo tipo de capa que suele estar presente en la estructura de una CNN corresponde a la capa de "pooling", la cual, en palabras simples, se encarga de reducir el tamaño de la salida de una capa convolucional [6, 7]. Esta reducción de tamaño es útil, pues permite

reducir la carga computacional, el uso de memoria y el número de parámetros de la red, lo cual limita el riesgo de sobre ajuste [6].

Las capas de "pooling", al igual que las convolucionales, tienen sus neuronas conectadas sólo a un número limitado de neuronas de la capa anterior. También se les puede definir el tamaño, "stride" y "padding". Sin embargo, a diferencia de las capas convolucionales, las capas de "pooling" no tienen pesos asociados, sino que aplican una función de agregación. Por ejemplo, se puede utilizar el máximo (conocido como "max pooling") o el promedio (conocido como "average pooling"). De esta manera, si se tuviera una capa de "max pooling" de tamaño 2x2 con "stride" de 2 (como en la figura 2.3), se tomarían 4 píxeles de una imagen, dejando para la salida el que tuviera mayor valor. Esto sería aplicado para toda la imagen de entrada, resultando en una salida de tamaño 4 veces menor [6].

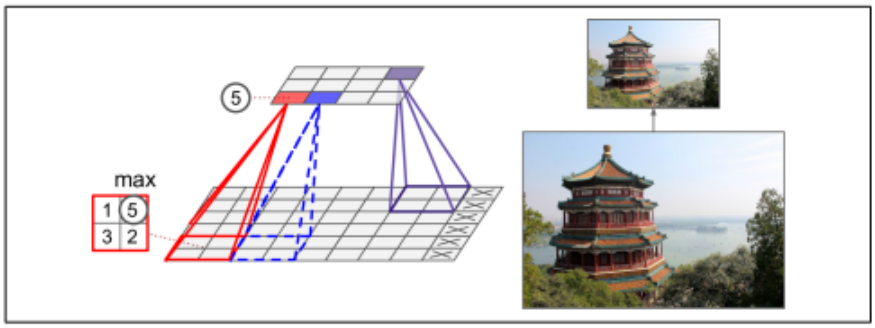


Figura 2.3: Capa de "max pooling" de tamaño 2x2 que utiliza un "stride" de 2, sin "padding". Se muestra como funcionaría el "max pooling" en una imagen, reduciendo su tamaño, pero manteniendo los píxeles de mayor valor dentro de ciertas áreas. Figura extraída de [6].

Existe otra propiedad obtenida a partir de las capas de "pooling", la cual se utiliza si importa más que una característica esté en una imagen, que el dónde está [8]. Ser invariante a la traslación (ver figura 2.4 [6]) significa que, si la entrada es trasladada algunos píxeles a algún lado, la salida de la capa de "pooling" no será muy diferente a la original. De todas formas, ésta propiedad no resulta útil en la tarea actual, debido a que se necesita segmentar cada roca, por lo que una traslación de píxeles debe ser llevada al resultado final, ya que implicaría el límite de una roca diferente. Es debido a ello que la arquitectura de red usada en el presente trabajo no utiliza una gran cantidad de capas de "pooling".

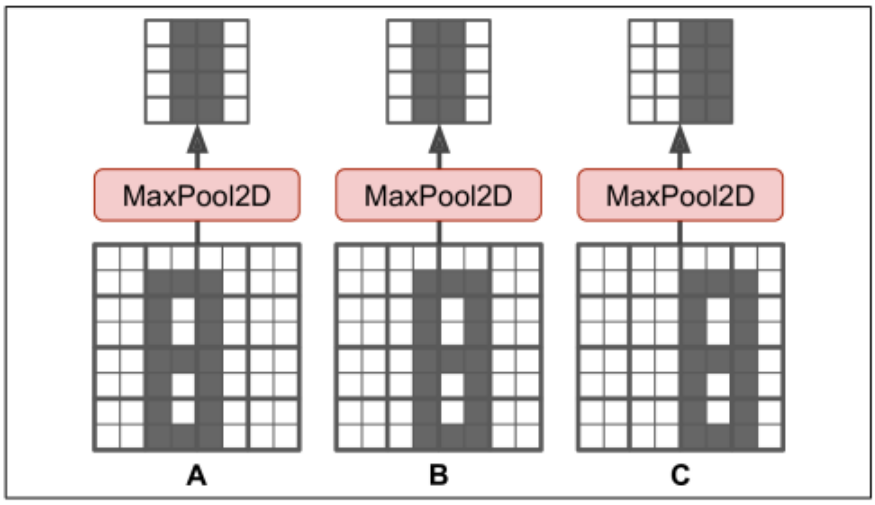


Figura 2.4: Ejemplo de "max pooling", representando la invarianza ante traslaciones. Al mover la figura "A" 1 píxel a la derecha (es decir, "B"), el resultado de realizar "max pooling" es el mismo. A pesar de que el resultado en "C" es diferente, sigue compartiendo una gran cantidad de píxeles con la original. Figura extraída de [6].

2.1.4. Arquitectura de una CNN

Con la ayuda de los conceptos previamente explicados es posible construir lo que suele ser un típico bloque de una CNN, el cual consiste de una capa convolucional, seguida de una función de activación no lineal (usualmente una función conocida como ReLu), concluyendo con una capa de "pooling". Con múltiples de estos bloques, los cuales tienen sus propios parámetros internos (pueden ser iguales o diferentes entre si), es como se suele construir una CNN. Al tener una imagen de entrada, durante su paso a través de la red, va reduciendo su tamaño, sin embargo se vuelve más específica a ciertos detalles dentro de la imagen, generando múltiples "feature maps". Con esto se puede construir la base de una red capaz de clasificar una imagen (ver figura 2.5 [9]). Cuando se considera que se utilizaron suficientes bloques convolucionales se suelen agregar capas de redes neuronales comunes completamente conectadas, hasta llegar a la capa de salida que genera la predicción de la clase mediante una función "softmax".

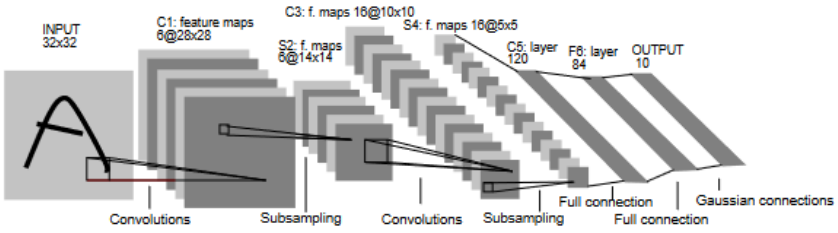


Figura 2.5: Arquitectura de LeNet-5, una CNN creada para detección de dígitos. Cada plano corresponde a un "feature map". En la figura se pueden observar tanto las capas convolucionales como las de "pooling", nombradas como "subsampling". Figura extraída de [9].

2.2. Aumentación de Datos

Para que un modelo basado en inteligencia computacional funcione correctamente es importante que sea capaz de generalizar, es decir, que no se haya sobre ajustado. Un modelo sobre ajustado entregará buenos resultados para el conjunto de entrenamiento, pero para el conjunto de prueba funcionará incorrectamente, pues el modelo aprendió de memoria la mayoría de detalles y características del conjunto de entrenamiento, los cuales no necesariamente se encontrarán en los datos de prueba [10]. Para observar el sobre ajuste durante el entrenamiento del modelo es necesario graficar las curvas de entrenamiento y validación respecto a las épocas, gracias a lo cual se podrá ver el punto en donde el modelo empieza a fallar, mostrando dónde es necesario aplicar diversas técnicas para poder evitarlo. De esta manera se intenta implementar un modelo que logre mantener un descenso en el error de entrenamiento y validación [11].

Un método útil para prevenir el sobre ajuste corresponde a la aumentación de datos (**DA**, de su nombre en inglés: "data augmentation"). Se trata de aumentar la cantidad de datos presentes en el conjunto de entrenamiento a través de diferentes métodos que logren generar nueva información. De esta manera también se logra generar un conjunto de datos que pueda tener características que posiblemente se encuentren en el conjunto de validación o prueba, logrando obtener mejores resultados para los mismos.

Es importante tener en cuenta que los métodos de DA utilizados sean efectivos. Debido a ello es importante realizar un estudio sobre los métodos que se utilizarán y sus respectivos parámetros, además de revisar que las segmentaciones sigan siendo correctas. Existen múltiples métodos de DA, de los cuales algunos resultan útiles y son comúnmente usados para una base de datos de imágenes. A continuación se procederá a mencionar y explicar algunos de los utilizados para la metodología implementada. Considerando la figura 2.6 como la imagen original, se verán ejemplos por cada método de tal modo que las variaciones implementadas queden claras.

Figura 2.6: Imagen extraída de la base de datos editada. Original de [12].

2.2.1. Volteo Vertical/Horizontal

Los métodos de volteo vertical (**VF**, de su nombre en inglés: "vertical flip") y volteo horizontal (**HF**, de su nombre en inglés: "horizontal flip") corresponden a voltear la imagen original en la dirección vertical (ver figura 2.7(a)) y horizontal (ver figura 2.7(b)) respectivamente. Éstos métodos se pueden utilizar por si sólos o de manera combinada (ver figura 2.7(c)), generando una imagen volteada en horizontal y vertical, con lo cual se pueden tener 4 versiones diferentes de una misma imagen, contando la original.

(a) HF

(b) VF

(c) Ambos

Figura 2.7: Resultados de aplicar HF, VF y ambos a la vez a la imagen de la figura 2.6.

2.2.2. Recortes Aleatorios

El método de recortes aleatorios (**RC**, de su nombre en inglés: "random crop") consiste en cortar aleatoriamente un área de la imagen y centrarse sólo en esta área recortada (ver figura 2.8). Es importante evitar generar imágenes muy pequeñas o que no tengan objetos relevantes. Debido a que la imagen recortada es más pequeña que la original, en ciertas ocasiones es importante utilizar un método que la devuelva a un tamaño igual, o casi similar al original [11].

Figura 2.8: Resultados de aplicar RC a la imagen de la figura 2.6. En ellos se puede ver como se hizo un “zoom” a diferentes zonas de la imagen original.

2.2.3. Brillo Aleatorio

El brillo aleatorio (**RB**, de su nombre en inglés: “random brightness”) consiste en cambiarle el brillo a las imágenes dentro de la base de datos. Este cambio de brillo se realiza de manera aleatoria dentro de un rango previamente estudiado, el cual no genere mucha luz u oscuridad que imposibiliten el encontrar el o los objetos dentro de la imagen. En la figura 2.9 se pueden ver ejemplos de una disminución y aumento de brillo de la imagen original en la figura 2.6.

(a) Disminuir Brillo (b) Aumentar Brillo

Figura 2.9: RB aplicado a la imagen de la figura 2.6, en (a) se puede ver una clara reducción del brillo original, mientras que en (b) un aumento del mismo.

2.2.4. Saturación Aleatoria

Al utilizar la saturación aleatoria (**RS**, de su nombre en inglés: "random saturation"), la saturación de las imágenes dentro de la base de datos se ve modificada de forma aleatoria dentro de un rango asignado, el cual se encuentra previamente estudiado para no perder información relevante de cada objeto. En la figura 2.10 se pueden ver ejemplos de una disminución y aumento de saturación de la imagen original en la figura 2.6.

(a)
Disminuir Saturación

(b)
Aumentar Saturación

Figura 2.10: RS aplicado a la imagen de la figura 2.6, en (a) se puede ver una clara reducción de la saturación original, mientras que en (b) un aumento de la misma.

2.2.5. Contraste Aleatorio

El contraste aleatorio (**RCo**, de su nombre en inglés: "random contrast") funciona de forma similar al brillo y la saturación, sin embargo, en este caso se modifica el contraste de las imágenes dentro de la base de datos. Nuevamente es necesario tener los límites del aumento y disminución de contraste previamente estudiados para no perder detalles de cada

objeto que dificulten un correcto entrenamiento de la red. En la figura 2.11 se pueden ver ejemplos de una disminución y aumento de contraste de la imagen original en la figura 2.6.

(a)
Disminuir Contraste

(b)
Aumentar Contraste

Figura 2.11: RCo aplicado a la imagen de la figura 2.6, en (a) se puede ver una clara reducción del contraste original, mientras que en (b) un aumento del mismo.

2.2.6. Rotación

Finalmente se tiene la Rotación, en la cual se asigna un rango de ángulos en el cual girar las imágenes de la base de datos. La rotación utilizada no es aleatoria, debido a que se decidió utilizar valores específicos para aumentar los datos. También existe la necesidad de asignar un método de tratar los bordes, entre los cuales se tiene la posibilidad de mantener el borde negro mientras se gira la imagen (ver figura 2.12(a)), reflejar los píxeles para no generar áreas negras (ver figura 2.12(b)), o cortar los bordes que no existen en la imagen girada (ver figura 2.12(c)).

(a) Constante

(b) Reflejo

(c) Cortado

Figura 2.12: Resultados de aplicar una rotación a la imagen de la figura 2.6, estos ejemplos corresponden a una rotación de 20º. En (a) se observa la rotación de la imagen original, manteniendo el tamaño, pero añadiendo un fondo negro constante para rellenar. En (b) se tiene el mismo caso que en (a), pero en vez de rellenar con negro, se reflejan los píxeles de la imagen. En (c), en vez de mantener el tamaño original, se corta el área que no existe en la imagen original, por lo que se reduce el tamaño de la imagen.

2.3. Segmentación de Instancia

Debido a que se busca implementar un método capaz de segmentar cada fragmento de roca dentro de una imagen para poder calcular su tamaño, es necesario detectar cada instancia única y asignarle sus propios datos de posición. En la segmentación de instancia se busca unificar el problema de la detección de objetos en imágenes, realizado mediante el uso de "bounding boxes", y la segmentación semántica de distintas clases en las imágenes, implementada mediante el marcado de cada píxel perteneciente a cada clase. De esta manera se logra definir las distintas instancias de diferentes clases dentro de una imagen [13].

Para entender mejor lo que realiza la segmentación de instancia se procederá a explicar sobre los dos métodos que unifica, la detección y la segmentación semántica.

2.3.1. Detección

La detección de objetos consiste en la tarea de clasificar y localizar múltiples objetos en una imagen [6]. Para ello se predice un "bounding box" que encierre cada objeto dentro de la imagen. Este "bounding box" usualmente consiste en las coordenadas horizontales y verticales del centro de un objeto, además de su altura y ancho.

La métrica que se utiliza para saber que tan bien predice las “bounding boxes” un modelo se llama “Intersection over Union” (IoU), la cual corresponde al área en la que se superponen la “bounding box” predicha con la real, dividido por el área de la unión de las mismas (ver figura 2.13 [6]).

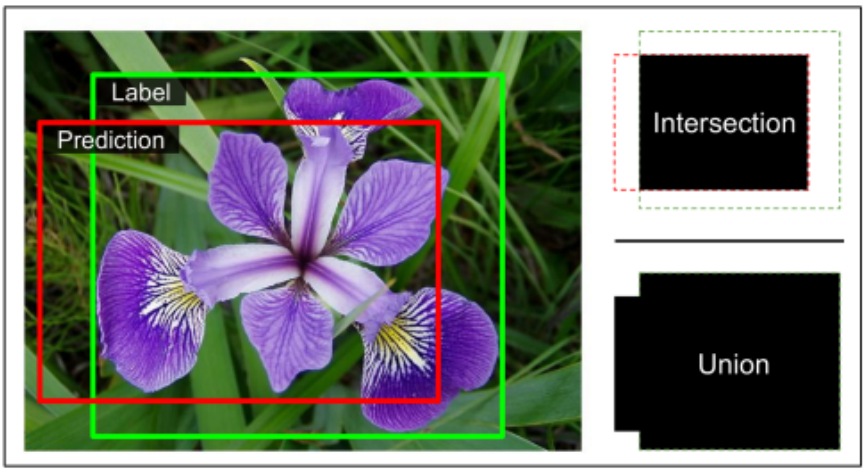


Figura 2.13: A la izquierda se observa un ejemplo de una “bounding box” predicha (rojo) y la real (verde). A la derecha se observa de forma gráfica el cálculo de la IoU. Extraído de [6].

2.3.2. Segmentación semántica

La segmentación semántica consiste en la clasificación por píxel, esta clasificación se realiza según la clase a la que pertenezca el objeto que tenga ese píxel analizado [6]. Sin embargo, objetos diferentes de una misma clase no son diferenciados, por ejemplo, en la figura 2.14 se puede observar como cada persona, edificio, auto o bicicleta es asignado al mismo color, uniendo instancias que se encuentren muy cercanas entre si.

Figura 2.14: Figura que representa la segmentación semántica, a la izquierda está la imagen de entrada, a la derecha la salida del modelo. Extraído de [6].

Aunque no es la única forma de implementar la segmentación de instancia, el método desarrollado une la detección y la segmentación semántica de una forma específica para

poder ejecutarla. En palabras sencillas, la primera parte del proceso intenta detectar todos los objetos dentro de la imagen de entrada, asignándoles una "bounding box" con una clase respectiva. Una vez se tienen detectados los objetos, se puede tratar cada "bounding box" como una tarea de segmentación semántica diferente. Así, se asigna a cada píxel del objeto detectado, la clase asignada a la "bounding box". De esta manera se logrará diferenciar cada objeto, ya que no se encontrarán todos asignados a la misma masa de píxeles, como sucede en la figura 2.14.

Capítulo 3

Estado del Arte

Como se mencionó previamente, hasta la fecha, no se han realizado muchos estudios sobre el uso de CNN's para realizar un análisis de granulometría en la minería [5]. Uno de los métodos más utilizados actualmente corresponde al análisis de imágenes digitales, entre los que se pueden encontrar métodos "hand crafted" (hecho a mano), cómo también métodos completamente automáticos. Los métodos "hand crafted" pueden ser implementados de diferentes maneras, por ejemplo, se pueden utilizar detecciones de borde, implementaciones con filtros gabor, "support vector machines" (SVM) y/o "principal component analysis" (PCA) [14]. Sin embargo, como se dijo previamente, para implementar los métodos nombrados, se necesita de un experto que los conozca, junto a mucho tiempo de prueba para encontrar parámetros eficientes.

Es importante destacar que existen programas que, mediante una imagen digital con fragmentos de rocas, son capaces de detectar los bordes de rocas y entregar detalles del tamaño de las mismas (WipFrag, Fragscan y Split Desktop entre los más conocidos). El problema con estos programas pagados es que, a medida que se complejiza el contenido de la imagen, es menos probable que funcionen bien [5]. Adicionalmente, estos sistemas también requieren de alguien que esté calificado para usarlos, que tenga el conocimiento sobre los parámetros que hay que asignar para optimizar las detecciones de borde que realiza el programa.

Por otro lado, se encuentran los métodos que utilizan CNN's, en los cuales el usuario simplemente tiene que ingresar la imagen y la red le entrega el resultado ya segmentado. Así, ya no es necesaria la intervención de un experto más allá de la implementación y entrenamiento de la red. El uso de CNN's permite realizar una implementación diferente según quien la realice. A continuación se mencionarán diversos estudios realizados sobre la detección y segmentación de rocas en imágenes digitales.

Previamente se mencionó la metodología de la detección de bordes, la cual, aunque funciona correctamente, requiere de un experto para implementar el método. Sin embargo, en el artículo [15] se implementa la detección de bordes mediante el uso de una CNN. En este trabajo se utiliza una red llamada DexiNed para detectar los bordes de las rocas dentro de la imagen. A pesar de ello, sigue siendo necesario realizar tanto un pre procesamiento a las imágenes de entrada, como un post procesamiento al resultado entregado por la red. El pre procesamiento consiste en llevar las imágenes a escala de grises y aplicar un método de ecualización de histogramas para definir mejor cada roca, conocido como CLAHE ("contrast-

limited adaptive histogram equalization"). Por otro lado, el post procesamiento se utiliza para evitar la discontinuidad de bordes o para separar los contornos de rocas que quedaron muy unidas, para lo cual se vuelve a necesitar de un método "hand crafted" implementado por un experto que debe saber que parámetros utilizar para obtener los mejores resultados. En la figura 3.1 se observan algunos resultados gráficos que se obtuvieron con esta metodología.

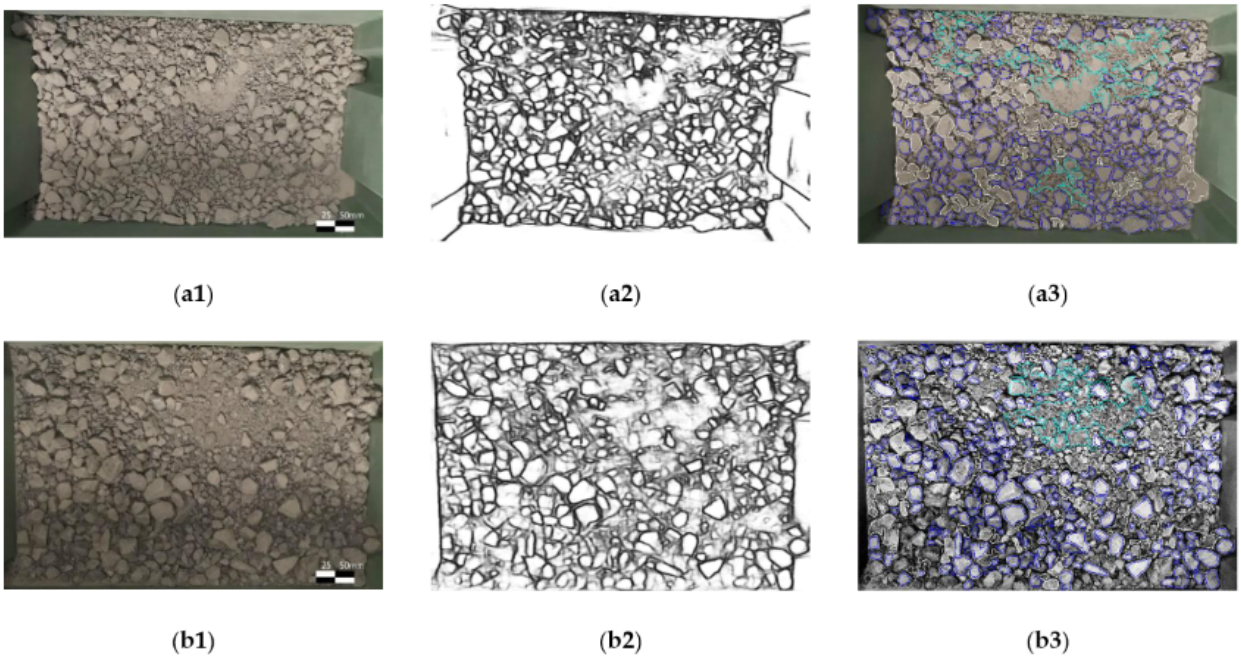


Figura 3.1: Algunos resultados gráficos obtenidos por el método descrito en [15]. (a1) y (b1) corresponden a las imágenes originales, (a2) y (b2) corresponden al mapa de bordes obtenido de cada imagen de entrada y, (a3) y (b3) corresponden al mapa de contorno post procesado, dónde el contorno azul corresponde a rocas singulares, el celeste corresponde a una región con material fino y el blanco, a las rocas que se encontraban conectadas. Figura extraída de [15].

Este método podría resultar ser un posible apoyo durante el trabajo de segmentación de un experto, ya que la generación del mapa de contornos podría considerarse como una segmentación inicial, la cual el experto corregiría según corresponda. Sin embargo, estos resultados de contorno requieren de un post procesamiento extenso implementado separado de la CNN, por lo que podría ser más sencillo para un experto iniciar sin el apoyo de estos resultados. Por lo tanto, este trabajo no se complementa con lo buscado en el presente trabajo, dónde se espera que la CNN sea capaz de segmentar directamente cada instancia de roca dentro de una imagen con el menor error posible.

Existen algunos trabajos que implementan la segmentación de instancia para poder detectar y segmentar cada instancia de roca dentro de una imagen digital. Estos trabajos corresponden a los artículos [5], [16] y [17], los cuales se comentarán a continuación con mayor detalle.

En el artículo [16] implementan la segmentación de instancia para detectar fragmentos de rocas en una imagen mediante el desarrollo de una arquitectura que combina una subred

capaz de realizar la detección de objetos, con otra subred capaz de realizar segmentación semántica. En la figura 3.2 se puede observar un esquemático de la red que se implementa en el artículo. La subred de detección de objetos consiste en un modelo SSD ("single-shot detector") modificado para su implementación. Por otra parte, la subred de segmentación semántica consiste en una arquitectura U-Net modificada, la cual se encarga de segmentar los fragmentos dentro de cada detección realizada por la subred de detección. El mayor problema con esta implementación recae en el hecho de que falta documentación respecto a la red de segmentación semántica, por lo tanto, el método no es replicable. Debido a este motivo, a pesar de cumplir la tarea de segmentar instancias de rocas dentro de las imágenes, no se puede considerar lo suficientemente útil para realizar el presente trabajo.

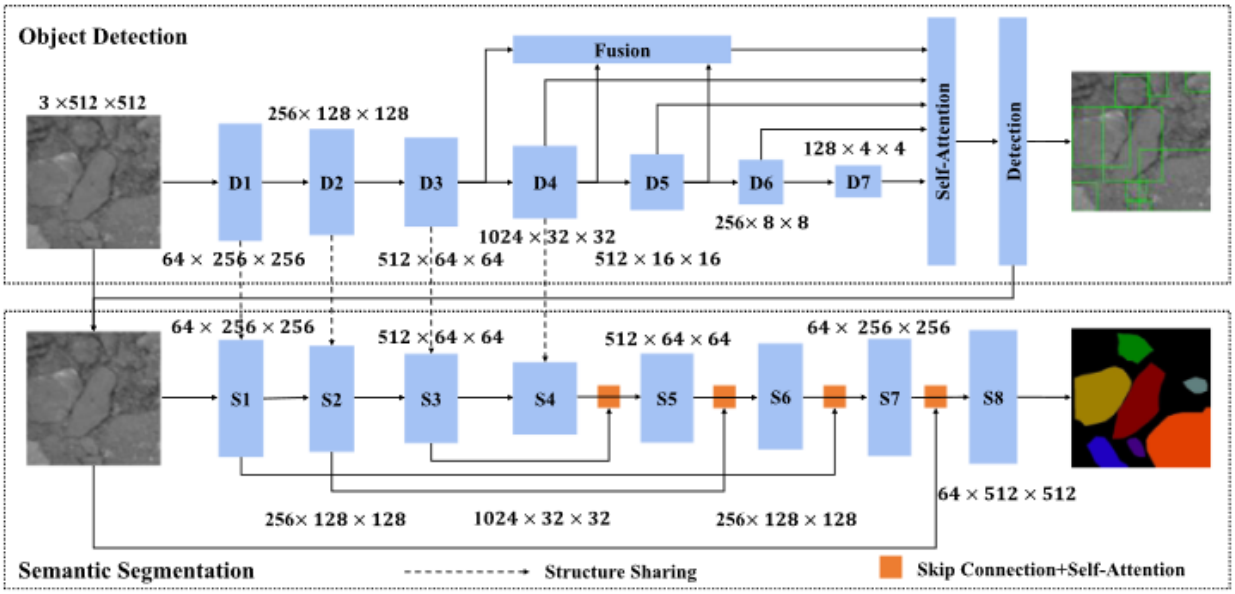


Figura 3.2: Esquemático de la red implementada en [16], los detalles disponibles de la misma se pueden leer en el mismo artículo. Figura extraída de [16].

En el artículo [17] se utiliza la plataforma de código abierto llamada MMDetection [18] para implementar un modelo llamado cascade Mask R-CNN, una extensión del modelo Mask R-CNN. En la figura 3.3 se pueden observar ejemplos de las imágenes utilizadas para entrenar y probar el modelo, dónde se puede ver claramente la gran diferencia con el trabajo que se desea realizar. En las imágenes se pueden observar fragmentos de rocas de tamaños similares y pequeños. Adicionalmente, estas rocas se encuentran sobre una superficie de madera. Estas imágenes no se asimilan a las que se desean utilizar para entrenar y probar el modelo, ya que se desea utilizar imágenes de rocas de múltiples tamaños en entornos poco favorables de una mina.

Figura 3.3: Ejemplos de imágenes utilizadas en [17]. Figura extraída de [17].

Finalmente, en el artículo [5] se utiliza la librería Detectron2, la cual es una librería de código abierto implementada por Facebook AI Research [19]. Detectron2 implementa la arquitectura de red conocida como Mask R CNN, con una gran cantidad de opciones para configurar según lo que se desee implementar, lo que en este caso, al igual que en el presente trabajo, consiste en segmentar cada instancia de roca dentro de una imagen entregada. Sin embargo, el trabajo realizado en [5] no es replicable debido a que los datos de entrenamiento no se encuentran disponibles y falta documentación sobre los hiperparámetros. Por consiguiente, debido a que es el trabajo más similar al que se busca implementar, se decidió utilizar Detectron2 para el desarrollo del método. Así, utilizando la base de datos de la presente memoria, se buscará mejorar los resultados que se obtienen con los hiperparámetros descritos en [5] por medio de diferentes métodos de aumentación de datos y variaciones de hiperparámetros.

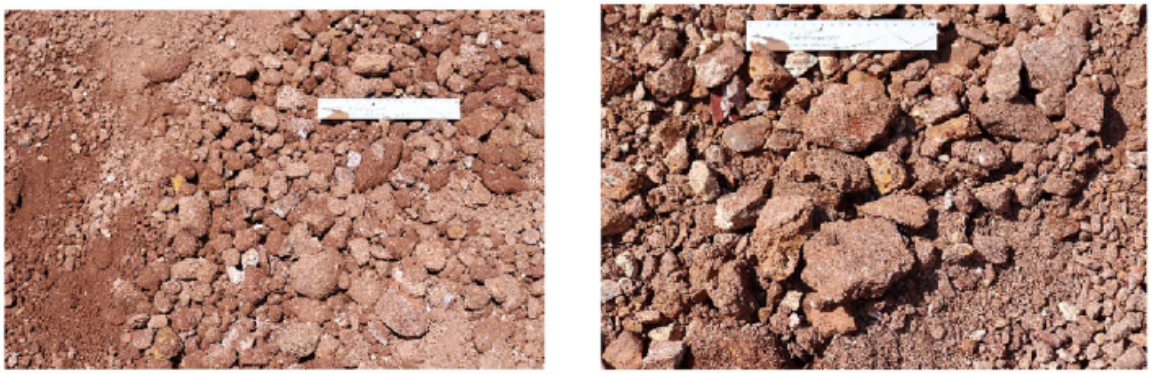


Figura 3.4: Algunos ejemplos de imágenes utilizadas en [5], la escala que comparten ambas imágenes tiene un largo de 93.172 mm. Figuras extraídas de [5].

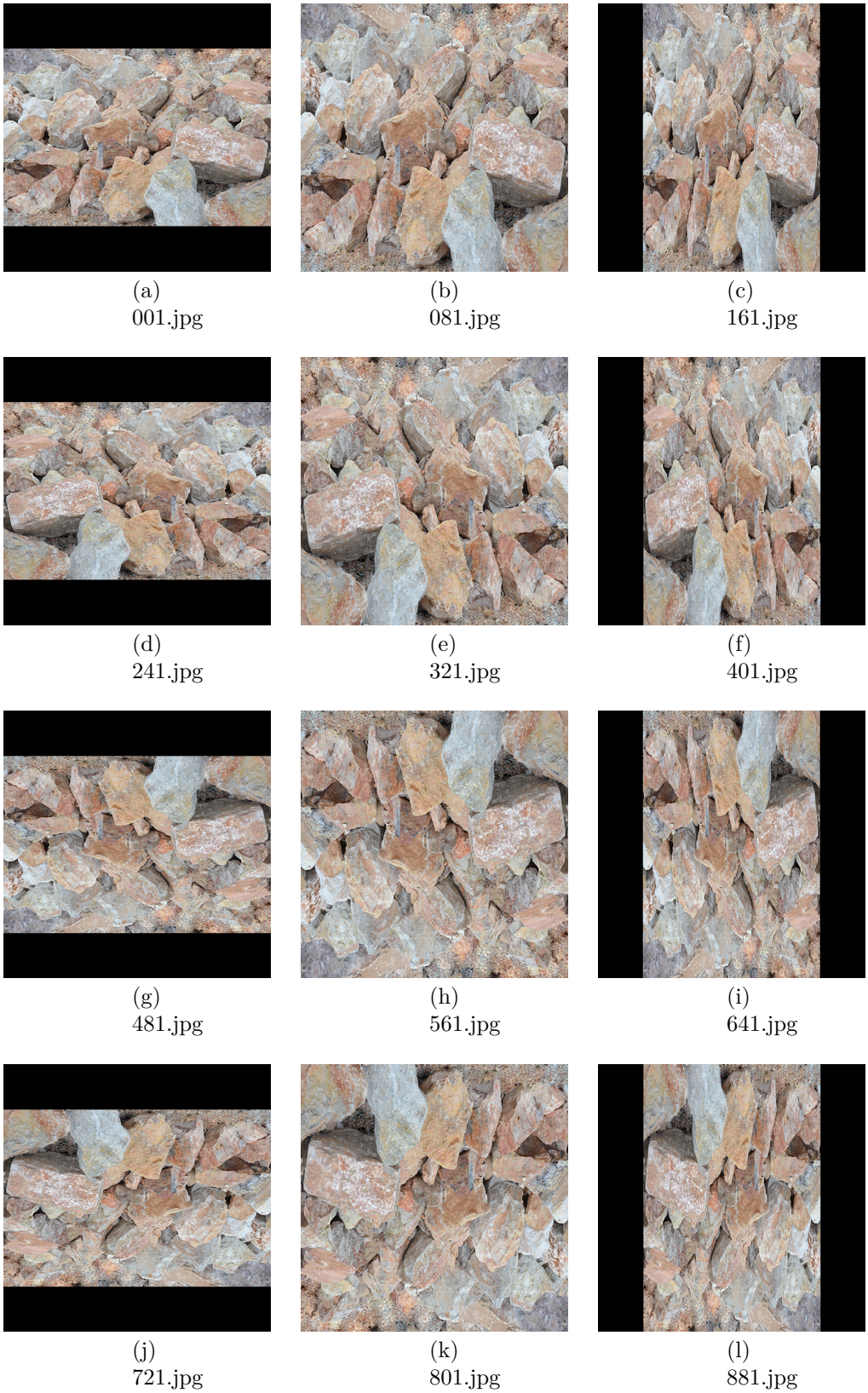
Capítulo 4

Metodología

4.1. Base de Datos

Para poder realizar el presente trabajo es importante tener una base de datos de imágenes de fragmentos de rocas segmentadas. Lamentablemente estos datos son realmente escasos, por lo que es necesario utilizar métodos de DA. La base de datos utilizada corresponde a una llamada "Data for: Rock fragment image segmentation combining CNN and watershed algorithm" [12], la cual se encuentra disponible en internet.

Esta base de datos corresponde a 960 imágenes de 512x512 píxeles con diferentes montones de fragmentos de rocas ya segmentadas vistos de diferentes ángulos. Sin embargo, realizando una revisión de las mismas, se encontró que las imágenes correspondían a solamente 80 imágenes originales, a las que ya se les había aplicado diversas combinaciones de métodos de DA. Específicamente se utilizaron VF, HF y aumento del alto o disminución del ancho, manteniendo el tamaño original. En la figura 4.1 se puede observar en (a) una de las imágenes originales, mediante la cual se pueden obtener las demás. En la primera columna ((a), (d), (g) y (j)) se pueden ver la imagen original en (a) y sus respectivas variaciones en HF, VF y ambas juntas. En la segunda columna ((b), (e), (h) y (k)) se tiene la misma imagen de su respectiva fila, pero con un estiramiento al alto de la imagen original, el cual elimina las franjas negras. En la tercera columna ((c), (f), (i) y (l)) se tiene lo mismo que en la segunda columna, pero se realiza un achatamiento respecto al ancho.



(a) 001.jpg (b) 081.jpg (c) 161.jpg

(d) 241.jpg (e) 321.jpg (f) 401.jpg

(g) 481.jpg (h) 561.jpg (i) 641.jpg

(j) 721.jpg (k) 801.jpg (l) 881.jpg

Figura 4.1: Ejemplo de la primera de las 80 imágenes que fue editada para obtener las 960 imágenes originales de la base de datos en [12].

Adicionalmente se pudo observar que habían instancias de rocas dentro de la base de datos que no habían sido segmentadas, por lo que se decidió utilizar las primeras 80 imágenes de la base de datos, añadiendo las segmentaciones faltantes.

Dentro de estas 80 imágenes revisadas existe una cantidad total de 6141 fragmentos de rocas, de los cuáles, según características internas del método utilizado para analizar los resultados (características de COCO), 4135 son fragmentos pequeños, 1844 corresponden a fragmentos medianos y el resto, 162, son rocas grandes. En la figura 4.2 se pueden apreciar estas características mencionadas de manera gráfica. En la figura se observa claramente que la base de datos no se encuentra balanceada respecto a la cantidad de distintos tamaños, lo cual podría generar posibles problemas durante el entrenamiento de la red.

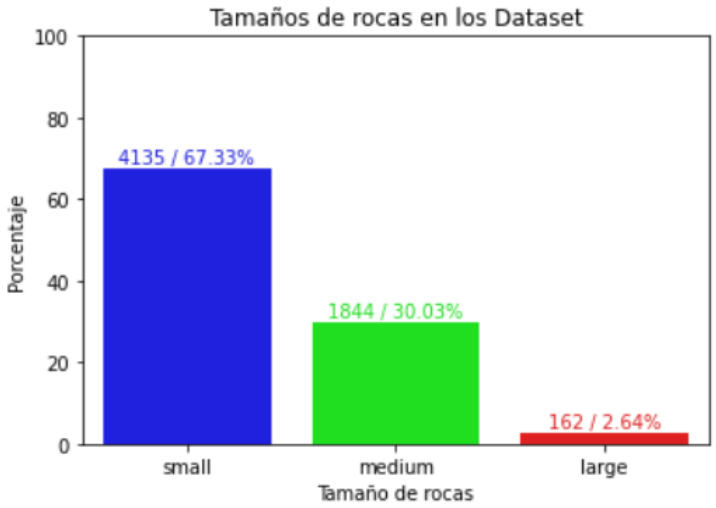


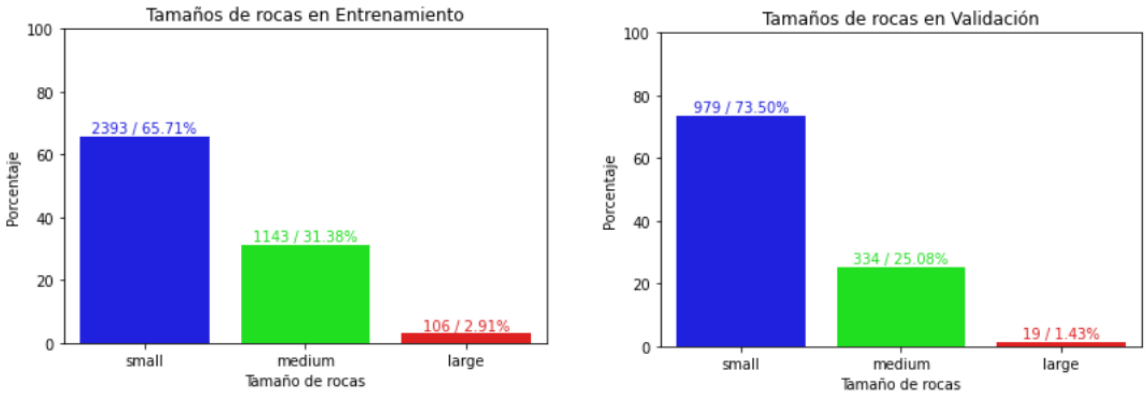
Figura 4.2: Gráfico que indica, según porcentaje, la cantidad de rocas de distintos tamaños que hay dentro de la Base de Datos. Sobre cada barra se encuentra el valor total de rocas, acompañado del porcentaje exacto.

Durante la revisión de estas imágenes se encontraron algunas que tenían similitudes entre sí. Sin embargo, debido a que tenían algunas diferencias respecto al ángulo y "zoom" en las que éstas fueron tomadas, se consideraron útiles para utilizar durante el entrenamiento de la red. Otra decisión tomada durante la revisión de las imágenes corresponde a la eliminación de las franjas negras en las imágenes. Estas franjas negras podrían añadir un comportamiento indeseado a la red durante el entrenamiento de la misma. Gracias a estos cambios quedan 80 imágenes de 339x512 como base de datos a usar durante el entrenamiento, validación y prueba del modelo a utilizar. En la figura 4.3 se muestra un ejemplo comparando una imagen original con la versión editada obtenida.

(a) Original (b) Editada

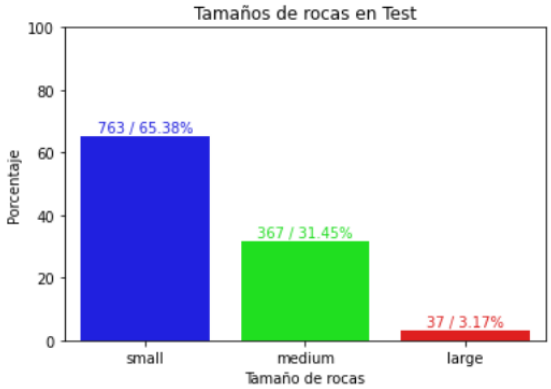
Figura 4.3: Ejemplo de la edición de la imagen 001.jpg de la base de datos.

Finalmente, es necesario separar los datos para generar la base de entrenamiento, validación y prueba. Para ello se decidió utilizar una división de 60 %/20 %/20 % respectivamente. Con esta división quedan las primeras 48 imágenes para el conjunto de entrenamiento, las siguientes 16 imágenes para el conjunto de validación y las últimas 16 imágenes para el conjunto de prueba. Es importante revisar que los fragmentos no queden desbalanceados dentro de estos conjuntos generados. Para el conjunto de entrenamiento se obtienen 3642 fragmentos, donde 2393 son rocas pequeñas, 1143 rocas medianas y 106 rocas grandes, para el conjunto de validación se tienen 1332 fragmentos de rocas, donde 979 son rocas pequeñas, 334 son rocas medianas y 19 son rocas grandes y, para el conjunto de prueba hay 1167 fragmentos de rocas, donde 763 son rocas pequeñas, 367 rocas medianas y 37 rocas grandes. En la figura 4.4 se observan estos detalles mencionados. Se ve claramente que el porcentaje de tamaños original mostrado en la figura 4.2 se mantiene relativamente similar dentro de cada conjunto generado. Dado que en cantidad de rocas total se respeta la separación de 60 %/20 %/20 %, se considera que los conjuntos generados se encuentran correctamente balanceadas según los datos iniciales.



(a) Entrenamiento

(b) Validación



(c) Prueba

Figura 4.4: Gráficos que indican, según porcentaje, los distintos tamaños de rocas dentro de cada conjunto de datos generado. Sobre cada barra se encuentra el valor total de rocas, acompañado del porcentaje exacto.

Es importante mencionar que, durante el desarrollo, se decidió utilizar un método que modifica el tamaño de las imágenes de entrenamiento, de manera tal que el lado de menor tamaño aumente a 800 píxeles, siempre y cuando el lado mayor no sobrepase los 1333 píxeles. Por lo tanto, las imágenes de entrada resultan en un tamaño de 800x1208 píxeles. Los tamaños de las rocas también se ven afectadas, lo cual se puede observar en la figura 4.5, dónde se muestra una clara disminución en la cantidad de rocas de tamaño pequeño, mientras que las de tamaño medio y grande aumentan, dejando el conjunto de entrenamiento más balanceado que el original. Se utilizará ese conjunto de entrenamiento editado para realizar los entrenamientos de cada experimento, a menos que se indique lo contrario.

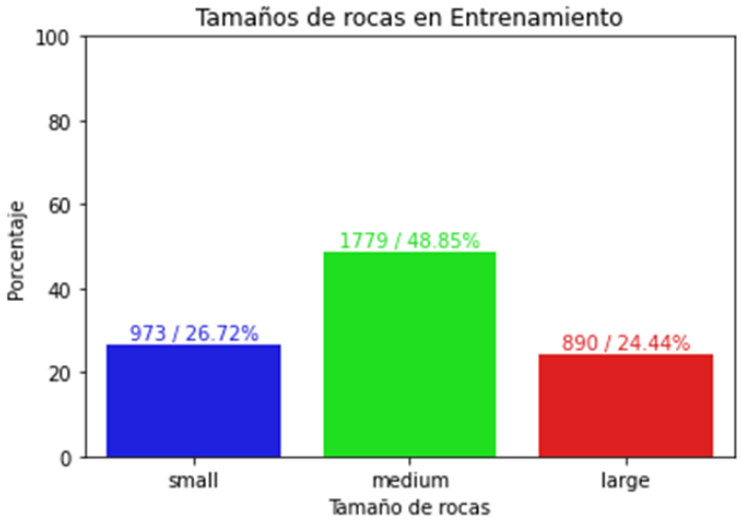


Figura 4.5: Gráfico que indica, según porcentaje, los distintos tamaños de rocas que hay dentro del conjunto de entrenamiento una vez se aumentan a un tamaño de 800x1208. Sobre cada barra se encuentra el valor total de rocas, acompañado del porcentaje exacto.

4.2. Detectron2

Detectron2 es una librería implementada por Facebook AI Research la cual otorga algoritmos estado del arte de detección y segmentación [19]. Esta librería se puede utilizar para el estudio de múltiples tareas de visión computacional, como lo son la detección, la segmentación de instancia, la detección de puntos clave en personas y la segmentación panóptica. La librería es capaz de analizar tanto imágenes como videos. Existen múltiples configuraciones pre-entrenadas que son entregadas dentro del código para facilitar el entrenamiento del usuario. La gran mayoría de estas redes pre-entrenadas fueron entrenadas con el dataset "Common Objects in Context" (COCO).

4.2.1. Estructura general

Detectron2 tiene múltiples configuraciones para las diversas tareas que puede realizar, entre ellas se encuentra la opción de elegir que tipo de redes pre-entrenadas se desea utilizar. Debido a que el presente trabajo consiste en implementar un método de segmentación de instancia, se utilizará una arquitectura llamada Mask R-CNN. La arquitectura de Mask R-CNN se puede separar en 3 partes diferentes. Un "backbone", encargado de obtener los "feature maps" que contengan la información relevante de una imagen. Una "region proposal network" (RPN), encargada de proponer múltiples regiones que podrían ser de interés. La última parte se llama "region of interest Heads" (ROI Heads), la cual se encarga de filtrar las múltiples regiones propuestas por la RPN para entregar el resultado final con los objetos dentro de la imagen detectados, segmentados y clasificados.

Si bien, hay múltiples opciones para elegir dentro de los "backbones" de la red, se elige el que mejores resultados de precisión y velocidad entrega según [20], un FPN-ResNet "backbone" pre-entrenado con la base de datos COCO. Este "backbone" se compone de la arquitectura "Feature Pyramid Network" (FPN), postulada en [21], la cual propone una

arquitectura con estructura piramidal, encargada de extraer las características de una imagen durante diversos niveles del procesamiento de la misma. En la figura 4.6 se observa la estructura general de la arquitectura.

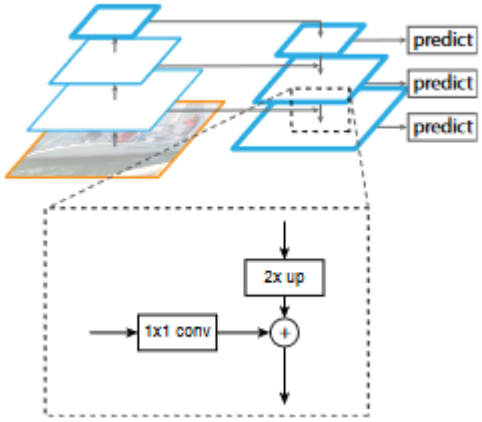


Figura 4.6: Estructura general de una FPN. Se hace un acercamiento que muestra la forma en que se unen las conexiones laterales con el camino hacia abajo de la arquitectura, los cuales se suman. Figura extraída de [21].

Como se puede observar en la figura 4.6, la FPN contiene 2 "caminos" en los que se procesa la imagen, un "camino hacia arriba" y un "camino hacia abajo". En el "camino hacia arriba" se extraen los "feature maps" en múltiples escalas de tamaño. En el presente trabajo, esta parte de la red consiste de una ResNet, arquitectura postulada en [22]. Por otro lado, el "camino hacia abajo", iniciado con el "feature map" específico generado en el último nivel del "camino hacia arriba", se encarga de sobre muestrear el mismo. Así logra generar "feature maps" de mayor calidad, los cuales, según se ve en la figura 4.6, se ven mejorados al unirlos con los "feature maps" del "camino hacia arriba". Es así como, en la configuración utilizada, se generan 5 "feature maps" de mejor calidad que se utilizarán en el siguiente paso.

Una vez se tienen los "feature maps" extraídos de la FPN, la RPN se encarga de proponer un máximo de 1000 regiones de interés (ROIs), correspondientes a posibles "bounding boxes" que contengan objetos importantes dentro de la imagen. Esta red consiste de 3 capas convolucionales, una capa convolucional de 3x3, una capa convolucional encargada de dibujar una "bounding box" y otra encargada de clasificar. En la figura 4.7 se puede observar un dibujo de la estructura general de esta red. Se puede observar que la red recorre todo el "feature map" analizado y, mediante "anchor boxes" de tamaños fijos, se encarga de asignar múltiples detecciones según sea pertinente.

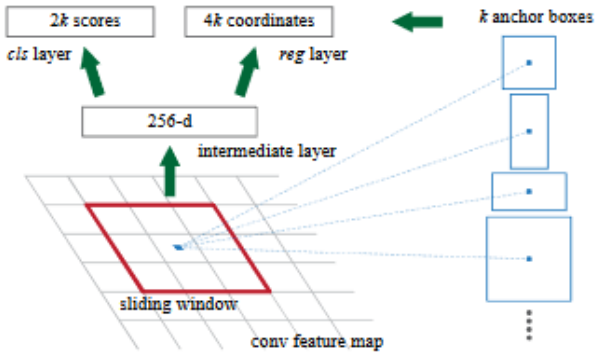


Figura 4.7: Estructura general de una RPN, se muestra en un punto fijo. Figura extraída de [23].

Una vez se hayan procesado todos los "feature maps", se calculan parámetros que les asignan un puntaje, seleccionando los 1000 mejores dentro de los obtenidos.

Por último se encuentra ROI Heads, la cual se encarga de finalizar con las predicciones de clase y "bounding box", y predice la máscara por cada ROI. De las 1000 regiones obtenidas en el proceso anterior, puesto en palabras simples, ROI Heads se encarga de filtrar los objetos detectados que tengan peor puntaje según cálculos internos que realiza. De esta manera, después de eliminar "bounding boxes" que se encuentren superpuestas, se obtendrá la imagen con sus respectivas detecciones, clasificaciones y segmentaciones encontradas, según las características internas de toda la red.

4.2.2. Métricas de evaluación

Las métricas de evaluación que se utilizan en este algoritmo corresponden a las métricas de COCO, es decir, "average precision" (AP) @$[IoU = 0.5 : 0.95]$, correspondiente a la precisión promediada sobre los rangos de IoU asignados dentro del paréntesis, AP @$[IoU = 0.5]$ (AP50) y AP@$[IoU = 0.75]$ (AP75). Es importante destacar que estas métricas se evalúan sobre todos los objetos dentro de la base de datos y entregan resultados entre 0 a 100. Existen también métricas que se centran en objetos pequeños, medianos o grandes dentro de la base de datos. Los límites de tamaño corresponden a los de COCO [1]. Los objetos pequeños tienen un área menor a 32x32 píxeles, los objetos de tamaño medio tienen área mayor a 32x32 y menor a 96x96, y los de tamaño grande son de área mayor a 96x96. Por lo tanto también existen las métricas de AP @$[IoU = 0.5 : 0.95]$ con áreas pequeñas (APs), AP @$[IoU = 0.5 : 0.95]$ con áreas medianas (APm) y AP @$[IoU = 0.5 : 0.95]$ con áreas grandes (APl).

Se utilizará también, para comparar algunos resultados, la prueba de Tukey. Esta prueba se encarga de comparar todos los resultados de 2 en 2, midiendo que tan diferentes son entre si. Esta diferencia la indica mediante un valor. Si el valor resulta mayor a 0.05, se considera que los resultados son suficientemente similares, y no se rechaza la hipótesis de que sean similares. En cambio, si el valor es menor a 0.05, la hipótesis de similitud si es rechazada. De esta manera, la prueba de Tukey se utilizará para ver, con mayor detalle, la similitud entre los experimentos realizados.

[1] https://cocodataset.org/detection-eval

Se implementó también un cálculo de precisión sobre las predicciones realizadas. Se obtiene calculado de los Verdaderos Positivos y los Falsos Positivos de una predicción. Para ello se comparan las áreas de las predicciones realizadas con las rocas originales. Si existe un alto nivel de superposición (70 %, asignado durante el desarrollo del trabajo) entre los píxeles de cada una comparado a la intersección, se considera la predicción como un Verdadero Positivo, si no se cumple en ningún caso, se asigna la predicción como un Falso Positivo. Debido a que no se centra tanto en la exactitud de los píxeles de la segmentación, se considera una métrica menos exigente que COCO. Esta métrica implementada también entrega sus resultados entre 0 y 100.

4.3. Metodología general

La metodología para poder cumplir el objetivo general es bastante simple, pero es necesario realizar múltiples repeticiones. Para generar un caso base, se eligen parámetros que serán los iniciales. Este caso base se realizará sin ningún método de DA y se entrenará con la base de entrenamiento por defecto. Para poder comprender mejor su comportamiento, será necesario realizarlo varias veces, para poder calcular un promedio y desviación estándar de sus resultados.

Una vez se tengan los resultados iniciales, simplemente será necesario ir probando diferentes configuraciones. Se probarán diferentes parámetros de la red, diferentes metodologías de DA y de edición de la base de entrenamiento. Todos los experimentos realizados se deberán repetir una cierta cantidad de veces. Durante el desarrollo se decidió repetir una cantidad de 5 veces por experimento.

Durante la realización de los diferentes experimentos realizados será importante ver los resultados obtenidos, para compararlos con el caso base y entre sí. De esta manera se desarrollará un método que sea mejor que los demás para la segmentación de múltiples rocas de diferentes tamaños dentro de imágenes digitales. La figura 4.8 muestra lo que se explicó de forma gráfica.

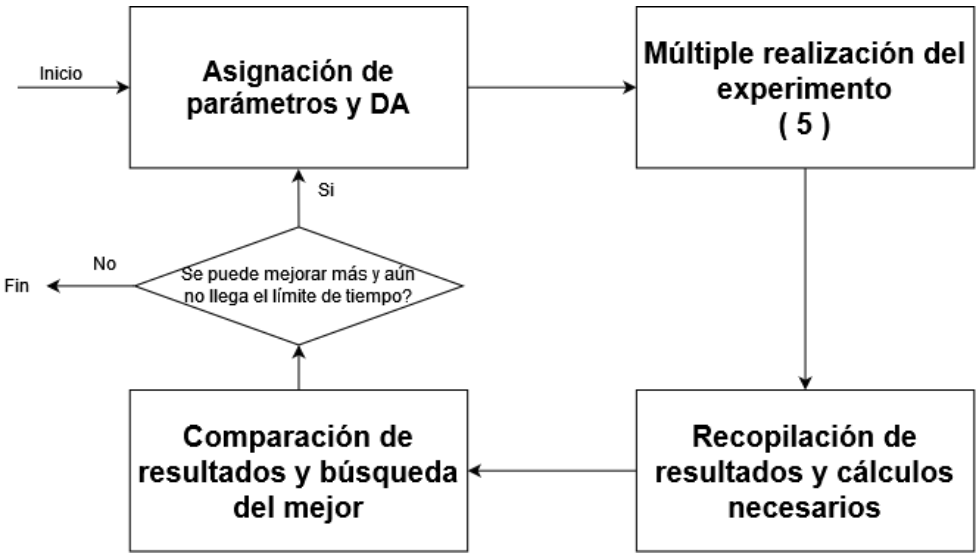


Figura 4.8: Diagrama de la metodología implementada.

Capítulo 5

Resultados y Discusión

En el presente capítulo se mostrarán algunos de los resultados obtenidos de los múltiples experimentos realizados. Estos resultados incluyen un caso base sin DA, los mejores resultados obtenidos y variaciones implementadas de los mismos resultados presentados. La gran mayoría de los experimentos se realizaron con características similares, modificando ciertos detalles para buscar mejoras de los resultados obtenidos.

Inicialmente se realizaron experimentos variados dónde se modificaban los hiperparámetros del modelo. Se editó la cantidad de iteraciones, el modelo pre-entrenado, la tasa de aprendizaje, entre otros. A continuación se indican los valores probados de los hiperparámetros.

- Iteraciones: [350, 550, 800, 1000, 4000]
- Tasa de aprendizaje: [0.0025, 0.005, 0.025, 0.05]
- "Batch" de imágenes: [2, 4]
- "Batch" de instancias: [128, 512]
- Modelo pre-entrenado: [Mask RCNN 50, Mask RCNN 101]
- Con "Scheduler": [Si, No]

Después de realizar múltiples pruebas se decidió utilizar las siguientes características para un caso base y sus futuras variaciones. Cada experimento es realizado un número de 5 veces para detallar mejor sus resultados presentados. Se utilizan 4000 iteraciones con una tasa de aprendizaje de 0.05. Se inicializa el modelo con pesos pre-entrenados. Para ello se utiliza un archivo de modelos pre-entrenados entregados por los desarrolladores de Detectron2. Se eligió el archivo *COCO-InstanceSegmentation/mask_rcnn_R_101_FPN_3x.yaml*. El entrenamiento usa un "scheduler" el cual actualiza la tasa de aprendizaje cada 400 iteraciones, utilizando un gamma de 0.1, momentum de 0.9 y "weight decay" de 0.0001. Se utiliza un tamaño de "batch" de 4 imágenes, dentro de las cuales se utiliza un "batch" de 512 instancias. Es importante recordar que se emplea un método con el cual se ajustan las imágenes de entrada a un tamaño de 800x1208. El hardware para entrenar los modelos es el entregado por google colab pro, haciendo que cada experimento demore aproximadamente 2 horas en

entrenar.

La primera sección muestra los resultados del caso base (Sin DA) junto a los resultados de realizar experimentos con los hiperparámetros de [5] (denominados como shubham) que si fueron documentados. Adicionalmente se tienen los resultados de probar el modelo con la mejor pérdida de validación (valloss) y mejor valor de precisión media de segmentación (segmAP) en validación durante el entrenamiento. La segunda sección mostrará una comparación del resultado base con los mejores resultados obtenidos, indicando el respectivo método de DA implementado. En la tercera se comentarán variaciones realizadas a estos mejores experimentos, buscando una metodología que entregue mejores resultados. En la cuarta sección se mencionan otros experimentos realizados con diferentes tipos de DA. Las últimas secciones muestran a mayor detalle los resultados del mejor experimento.

5.1. Caso base

A continuación se tienen los resultados obtenidos del caso base, comparándolo con un modelo entrenado con los datos de [5] y sus versiones con mejor precisión media de segmentación en validación y pérdida de validación. Debido a que la base de datos de [5] no se encuentra disponible, fue necesario realizar un experimento utilizando sus hiperparámetros disponibles, pero con la base de datos utilizada en el presente trabajo. De esta manera se pueden comparar los resultados obtenidos con los de un trabajo previo. Las tablas 5.1 y 5.2 contienen los resultados de las métricas COCO evaluadas. El experimento denominado **Sin DA** corresponde al caso base, al cual no se le aplica ningún método de DA, ni durante el entrenamiento ni a la base de datos, exceptuando el reajuste de tamaño mencionado con anterioridad. El experimento denominado "shubham" corresponde al realizado con los hiperparámetros de [5]. Los experimentos que tienen (segmAP) son los que se prueban con el modelo que obtuvo mejor precisión media de segmentación en validación durante el entrenamiento. En cambio, los que tienen (valloss) corresponden a los que se prueban con el modelo que obtuvo mejor pérdida de validación durante el entrenamiento.

Tabla 5.1: Resultados de "bbox" en formato COCO para el caso base, los hiperparámetros de [5] y sus versiones con mejor segmAP y pérdida de validación.

Resultado	AP	AP50	AP75	APs	APm	APl
Sin DA	44.6406 ±0.1001	71.2282 ±0.3181	49.1748 ±0.4142	34.8062 ±0.1379	62.2882 ±0.2707	76.6650 ±1.5528
Sin DA (segmAP)	43.7084 ±0.2759	64.7008 ±0.6146	49.7898 ±0.6914	31.8022 ±0.4478	66.4628 ±0.4094	81.6942 ±0.4871
Sin DA (valloss)	44.6236 ±1.0971	66.6746 ±2.3034	50.6678 ±1.2255	33.0406 ±1.4748	66.7696 ±0.6861	81.0090 ±1.2151
shubham	44.3840 ±0.3034	72.0772 ±0.5108	47.2500 ±0.6695	35.2056 ±0.3497	60.9042 ±0.4176	80.9582 ±1.2508
shubham (segmAP)	42.5760 ±0.3780	63.6584 ±0.6476	49.3566 ±0.5909	31.4200 ±0.5372	63.5002 ±0.2475	83.3614 ±0.6681
shubham (valloss)	42.5760 ±0.3780	63.6584 ±0.6476	49.3566 ±0.5909	31.4200 ±0.5372	63.5002 ±0.2475	83.3614 ±0.6681

Tabla 5.2: Resultados de segmentación en formato COCO para el caso base, los hiperparámetros de [5] y sus versiones con mejor segmAP y pérdida de validación.

Resultado	AP	AP50	AP75	APs	APm	APl
Sin DA	43.6206 ±0.1043	70.7066 ±0.3586	47.2988 ±0.5425	29.6324 ±0.2118	64.5204 ±0.2121	86.0506 ±0.3677
Sin DA (segmAP)	42.6182 ±0.3401	64.5910 ±0.6053	49.6064 ±0.2985	27.7668 ±0.5078	67.2214 ±0.3265	86.6922 ±0.7232
Sin DA (valloss)	43.3888 ±1.0749	66.3644 ±2.5807	49.7548 ±0.6674	28.6432 ±1.200	67.4014 ±0.7016	86.5314 ±0.3098
shubham	43.8904 ±0.1286	72.3170 ±0.4736	48.2804 ±0.2201	31.4182 ±0.2168	64.5206 ±0.2619	88.8994 ±0.3271
shubham (segmAP)	42.2902 ±0.3143	63.7062 ±0.4815	48.3146 ±0.4723	28.2400 ±0.4614	65.9916 ±0.2571	87.6686 ±0.4265
shubham (valloss)	42.2902 ±0.3143	63.7062 ±0.4815	48.3146 ±0.4723	28.2400 ±0.4614	65.9916 ±0.2571	87.6686 ±0.4265

La figura 5.1 y la tabla A.1 (en el anexo A) representan, respectivamente, los datos de segmAP de forma gráfica y la prueba de Tukey realizada sobre la misma métrica.

Los resultados presentados en las tablas y la figura muestran que es preferible utilizar el modelo una vez se encuentra completamente entrenado. Lo cual se vio reforzado durante la realización de los demás experimentos, donde la diferencia fue más notoria. Los resultados del caso base y los de "shubham" son bastante similares. Debido a su similitud, los métodos que mejoren el caso base serán métodos que mejoran los resultados obtenidos en [5].

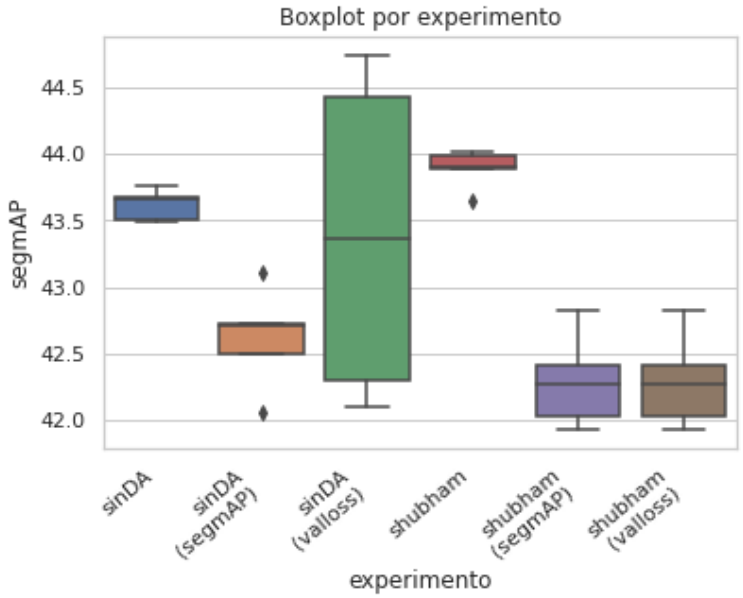


Figura 5.1: Comparación de segmAP entre el caso base, los hiperparámetros de [5] y sus versiones con mejor segmAP y pérdida de validación.

5.2. Mejores resultados

En esta sección, como se dijo previamente, se muestran los resultados obtenidos del caso base, comparándolo con los 3 mejores resultados obtenidos. Las tablas 5.3 y 5.4 contienen los resultados de las métricas COCO evaluadas. Los experimentos que no corresponden a **Sin DA**, son los que utilizan el método de DA de recortes aleatorios (**RC**). Este DA se aplica durante el entrenamiento a cada imagen de entrada sin excepción. Los números que acompañan a al nombre corresponden a la proporción con la que se corta la imagen original. Por ejemplo, en el caso de **RC.8.6** se corta la imagen original (339x512) de forma relativa según los valores indicados. El alto se reduce entre un valor de 0.8 - 1 del original, mientras que el ancho se reduce entre valores de 0.6 - 1 del original. Es decir, el tamaño de la imagen resultante puede quedar de tamaño (271-339)x(307-512) píxeles. Esta metodología logra crear una variedad de imágenes con diferentes tamaños durante el entrenamiento, para que la red pueda observar diferentes ejemplos mientras entrena. Es importante mencionar que se realizaron más experimentos que los mostrados en las tablas, utilizando rangos entre 0.6 - 0.8 para el alto, mientras que para el ancho, rangos de 0.4 - 0.7. Sin embargo, debido a la cantidad de resultados, sólo se muestran los 3 mejores obtenidos.

En las tablas presentadas se puede ver una clara mejora al caso base, exceptuando los casos de AP50 en "bounding box" (bbox) y uno en el AP50 de segmentación, en dónde existe una pequeña diferencia que vuelve mejor el caso base. Debido a que la naturaleza del trabajo consiste mayoritariamente en la correcta segmentación de rocas, es preferible centrarse más en los resultados de la tabla 5.4, los resultados de segmentación. De ellos se puede concluir que aplicar **RC**, utilizando los valores presentados, durante el entrenamiento de la red resulta conveniente, pues genera mejores resultados, especialmente de AP.

Tabla 5.3: Resultados de "bbox" en formato COCO para el caso base y los mejores resultados.

Resultado	AP	AP50	AP75	APs	APm	APl
Sin DA	44.6406 ±0.1001	71.2282 ±0.3181	49.1748 ±0.4142	34.8062 ±0.1379	62.2882 ±0.2707	76.6650 ±1.5528
RC.7.7	46.6884 ±0.2451	70.4424 ±0.6025	51.5358 ±0.3760	36.5838 ±0.5589	66.6502 ±0.5703	84.2480 ±0.5837
RC.8.6	46.9280 ±0.2009	70.7624 ±0.3980	51.9230 ±0.3586	36.8402 ±0.3887	66.7748 ±0.2749	83.3792 ±0.5511
RC.8.7	46.7990 ±0.1819	70.6986 ±0.4446	51.8678 ±0.3569	36.7174 ±0.2904	66.5096 ±0.0876	83.7056 ±1.4807

Tabla 5.4: Resultados de segmentación en formato COCO para el caso base y los mejores resultados.

Resultado	AP	AP50	AP75	APs	APm	APl
Sin DA	43.6206 ±0.1043	70.7066 ±0.3586	47.2988 ±0.5425	29.6324 ±0.2118	64.5204 ±0.2121	86.0506 ±0.3677
RC.7.7	46.7256 ±0.2465	70.3144 ±0.6796	52.3566 ±0.4398	32.6232 ±0.4236	68.6582 ±0.4201	89.3748 ±0.3364
RC.8.6	46.6848 ±0.1539	70.7810 ±0.4972	52.1028 ±0.3954	32.7740 ±0.3350	68.5340 ±0.1238	88.7802 ±0.4900
RC.8.7	46.8848 ±0.1508	70.9728 ±0.4147	52.4134 ±0.5398	32.8062 ±0.1266	68.7676 ±0.3858	88.9774 ±0.3468

En la figura 5.2 se muestra un "boxplot" de la métrica AP de segmentación (segmAP) de la tabla 5.4. En ella se puede observar la diferencia entre los resultados al no aplicar ningún DA y al aplicar RC con los valores indicados. Esa diferencia queda más detallada al realizar una prueba de Tukey, cuyos resultados se encuentran en la tabla 5.5.

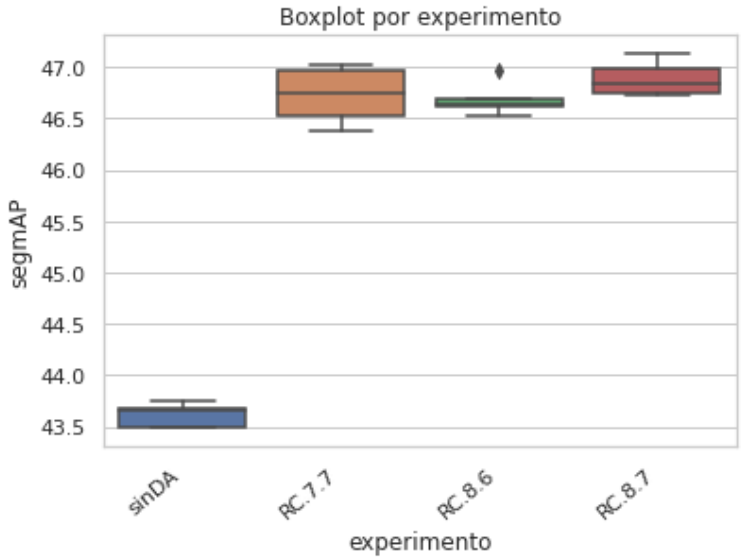


Figura 5.2: Comparación de segmAP entre los mejores métodos y el caso base.

Tabla 5.5: Prueba de Tukey entre mejores resultados y el caso base.

Grupo 1	Grupo 2	p-adj	Rechaza
RC.7.7	RC.8.6	0.9000	Falso
RC.7.7	RC.8.7	0.5642	Falso
RC.7.7	Sin DA	0.0010	Verdad
RC.8.6	RC.8.7	0.3833	Falso
RC.8.6	Sin DA	0.0010	Verdad
RC.8.7	Sin DA	0.0010	Verdad

La prueba de Tukey se realiza sobre la métrica de segmAP, la cual se considera la que mejor representa los resultados deseados. Sus resultados demuestran que no existe gran diferencia entre los 3 experimentos mostrados de RC. Pero todos ellos resultan claramente mejores que el caso base, logrando segmentar mejor las rocas presentes dentro de la base de prueba.

5.3. Variaciones de resultados

Se procedió a realizar algunas modificaciones a los mejores experimentos realizados, pues se deseaba averiguar si la configuración utilizada era la más conveniente. Es por ello que se probaron 2 variaciones. La primera consiste en modificar la aplicación del RC durante el entrenamiento de la red. Es decir, en vez de aplicar siempre el RC, indicar un cierto porcentaje de aplicación. De esta manera, durante el entrenamiento, la red sería entrenada con más de las imágenes originales. La segunda variación consiste en modificar previamente la base de datos de entrenamiento. Debido al claro desbalanceo que existe en la cantidad de rocas de diferentes tamaños, observado en la figura 4.5, se realiza manualmente un balanceo de los tamaños, generado mediante leves variaciones a las imágenes originales.

5.3.1. Modificando porcentaje de aplicación de RC

Las tablas 5.6, 5.7, 5.8, 5.9, 5.10 y 5.11 muestran los resultados obtenidos al implementar un RC durante el entrenamiento que no se aplica a todas las imágenes de entrada. Se realizan diferentes experimentos para cada uno de los mejores experimentos obtenidos para poder tener una idea general del efecto de modificar el porcentaje de aplicación. La primera parte del nombre de cada experimento es la misma que en la sección anterior, mientras que la segunda parte del nombre indica la probabilidad de aplicación del RC. Por ejemplo, **RC.8.7_prob.3** indica que, durante el entrenamiento, una imagen tiene una probabilidad de 0.3 de ser cortada de forma aleatoria, con proporción de (0.8, 0.7) al alto y ancho respectivamente. Es importante mencionar que se realizaron menos experimentos para un par de casos debido a que se observó un comportamiento similar.

Tabla 5.6: Resultados de "bbox" en formato COCO para variaciones de porcentaje de aplicación de mejores resultados de RC.7.7.

Resultado	AP	AP50	AP75	APs	APm	APl
RC.7.7	46.6884 ±0.2451	70.4424 ±0.6025	51.5358 ±0.3760	36.5838 ±0.5589	66.6502 ±0.5703	84.2480 ±0.5837
RC.7.7_prob.3	47.0588 ±0.2314	72.2404 ±0.4441	52.1800 ±0.5625	37.2114 ±0.2097	65.7970 ±0.1607	81.5726 ±1.4162
RC.7.7_prob.5	47.2970 ±0.1203	71.9560 ±0.3239	52.2744 ±0.5359	37.2234 ±0.1059	66.4978 ±0.3806	83.2746 ±0.6408

Tabla 5.7: Resultados de segmentación en formato COCO para variaciones de porcentaje de aplicación de mejores resultados de RC.7.7.

Resultado	AP	AP50	AP75	APs	APm	APl
RC.7.7	46.7256 ±0.2465	70.3144 ±0.6796	52.3566 ±0.4398	32.6232 ±0.4236	68.6582 ±0.4201	89.3748 ±0.3364
RC.7.7_prob.3	46.1570 ±0.2178	71.7652 ±0.1260	51.1362 ±0.5669	32.3234 ±0.1366	67.3702 ±0.3292	87.9358 ±0.7178
RC.7.7_prob.5	46.4440 ±0.1649	71.0504 ±0.0927	51.7290 ±0.5120	32.6546 ±0.3095	67.9392 ±0.3248	88.5978 ±0.1277

Tabla 5.8: Resultados de "bbox" en formato COCO para variaciones de porcentaje de aplicación de mejores resultados de RC.8.6.

Resultado	AP	AP50	AP75	APs	APm	APl
RC.8.6	46.9280 ±0.2009	70.7624 ±0.3980	51.9230 ±0.3586	36.8402 ±0.3887	66.7748 ±0.2749	83.3792 ±0.5511
RC.8.6_prob.3	46.9606 ±0.1335	71.9954 ±0.4131	52.1694 ±0.4199	37.0030 ±0.3008	65.8084 ±0.2699	81.2994 ±1.0898
RC.8.6_prob.5	47.1826 ±0.1258	71.8766 ±0.3217	52.6410 ±0.1051	37.2618 ±0.2811	66.1766 ±0.3183	82.4808 ±1.1357
RC.8.6_prob.7	47.2456 ±0.3814	71.2244 ±0.6053	52.9910 ±0.4567	37.2514 ±0.5957	66.4964 ±0.2699	84.0734 ±0.5602
RC.8.6_prob.8	47.2704 ±0.2124	71.2902 ±0.5940	52.6772 ±0.5160	37.3786 ±0.4288	66.7856 ±0.2293	83.8178 ±0.8259
RC.8.6_prob.9	47.0578 ±0.1803	70.9654 ±0.3638	52.6562 ±0.3680	37.2706 ±0.2887	66.3864 ±0.2502	83.1678 ±0.4249

Tabla 5.9: Resultados de segmentación en formato COCO para variaciones de porcentaje de aplicación de mejores resultados de RC.8.6.

Resultado	AP	AP50	AP75	APs	APm	APl
RC.8.6	46.6848 ±0.1539	70.7810 ±0.4972	52.1028 ±0.3954	32.7740 ±0.3350	68.5340 ±0.1238	88.7802 ±0.4900
RC.8.6_prob.3	45.9016 ±0.1324	71.6726 ±0.0848	50.8644 ±0.3676	31.9558 ±0.1887	67.1786 ±0.2326	87.5842 ±0.6706
RC.8.6_prob.5	46.3102 ±0.1102	71.0780 ±0.3586	51.5730 ±0.3761	32.5308 ±0.1581	67.7130 ±0.3488	88.1958 ±0.3738
RC.8.6_prob.7	46.6296 ±0.2649	71.0328 ±0.4067	52.0432 ±0.5129	32.9384 ±0.5322	68.1252 ±0.2615	88.5886 ±0.3330
RC.8.6_prob.8	46.6952 ±0.2334	70.8620 ±0.4926	52.1146 ±0.5148	32.9728 ±0.1425	68.0654 ±0.4042	88.9026 ±0.4299
RC.8.6_prob.9	46.8118 ±0.2157	71.2278 ±0.5611	52.1022 ±0.4060	32.8958 ±0.2703	68.4172 ±0.1585	88.6722 ±0.2915

Tabla 5.10: Resultados de "bbox" en formato COCO para variaciones de porcentaje de aplicación de mejores resultados de RC.8.7.

Resultado	AP	AP50	AP75	APs	APm	APl
RC.8.7	46.7990 ±0.1819	70.6986 ±0.4446	51.8678 ±0.3569	36.7174 ±0.2904	66.5096 ±0.0876	83.7056 ±1.4807
RC.8.7_prob.3	46.7424 ±0.1898	72.2512 ±0.3681	51.9638 ±0.4416	36.9288 ±0.2923	65.4432 ±0.2521	80.9046 ±1.1206
RC.8.7_prob.5	47.2838 ±0.3680	71.9342 ±0.3791	52.8402 ±0.1218	37.3422 ±0.4070	66.4610 ±0.2308	82.1066 ±1.6640

Tabla 5.11: Resultados de segmentación en formato COCO para variaciones de porcentaje de aplicación de mejores resultados de RC.8.7.

Resultado	AP	AP50	AP75	APs	APm	APl
RC.8.7	46.8848 ±0.1508	70.9728 ±0.4147	52.4134 ±0.5398	32.8062 ±0.1266	68.7676 ±0.3858	88.9774 ±0.3468
RC.8.7_prob.3	45.7578 ±0.1121	71.4124 ±0.6774	50.6078 ±0.2877	31.8650 ±0.2976	67.2964 ±0.4196	87.6576 ±0.8457
RC.8.7_prob.5	46.2642 ±0.2895	71.5528 ±0.4604	51.1600 ±0.3376	32.6368 ±0.3381	67.7292 ±0.1488	88.0958 ±0.5662

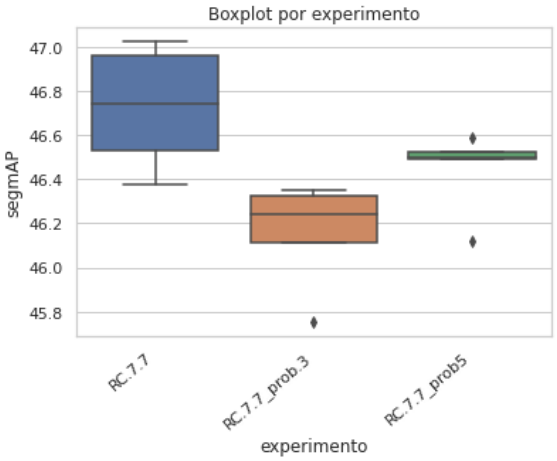
Estos resultados, si bien son mejores que el caso base y generan mejores resultados para el caso "bounding box", resultan bastante similares para las métricas de segmentación. Logran mejorar algunos casos, mientras que en otros resulta mejor el caso original. En la figura 5.3 se pueden observar los diferentes resultados obtenidos para segmAP en un gráfico de cajas. Para poder definir si alguno es objetivamente mejor que otro, sería necesario realizar más experimentos y tener en cuenta más detalles. Adicionalmente, una prueba de Tukey para cada caso indica que no existe una mayor diferencia entre los resultados de segmAP que son más similares entre si. Los casos más interesantes son los de **RC.8.6** con **RC.8.6_prob.7**, **RC.8.6_prob.8** y **RC.8.6_prob.9**, los cuales tienen un valor p de 0.9 entre si, indicando una diferencia casi nula. Los resultados de la prueba de Tukey de los casos más interesantes se encuentran en la tabla 5.12, los otros se encuentran en el anexo B, las tablas B.1 y B.2 respectivamente.

Tabla 5.12: Prueba de Tukey entre el mejor resultado relacionado y variaciones de la probabilidad de aplicar RC.8.6.

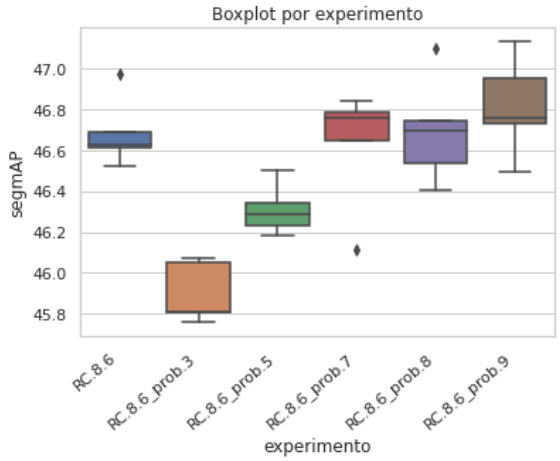
Grupo 1	Grupo 2	p-adj	Rechaza
RC.8.6	RC.8.6_prob.3	0.0010	Verdad
RC.8.6	RC.8.6_prob.5	0.1040	Falso
RC.8.6	RC.8.6_prob.7	0.9000	Falso
RC.8.6	RC.8.6_prob.8	0.9000	Falso
RC.8.6	RC.8.6_prob.9	0.9000	Falso
RC.8.6_prob.3	RC.8.6_prob.5	0.4086	Falso
RC.8.6_prob.3	RC.8.6_prob.7	0.0010	Verdad
RC.8.6_prob.3	RC.8.6_prob.8	0.0010	Verdad
RC.8.6_prob.3	RC.8.6_prob.9	0.0010	Verdad
RC.8.6_prob.5	RC.8.6_prob.7	0.2193	Falso
RC.8.6_prob.5	RC.8.6_prob.8	0.0891	Falso
RC.8.6_prob.5	RC.8.6_prob.9	0.0137	Verdad
RC.8.6_prob.7	RC.8.6_prob.8	0.9000	Falso
RC.8.6_prob.7	RC.8.6_prob.9	0.7397	Falso
RC.8.6_prob.8	RC.8.6_prob.9	0.9000	Falso

De todas maneras, los resultados presentados indican que una probabilidad alta de aplicar

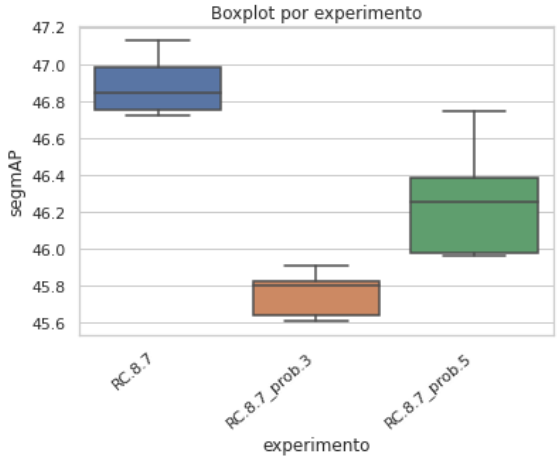
RC es mejor que una baja. No sólo parecen tener mejor resultado de segmentación, sino que también mejora los resultados de "bounding box".



(a) Caso RC.7.7



(b) Caso RC.8.6

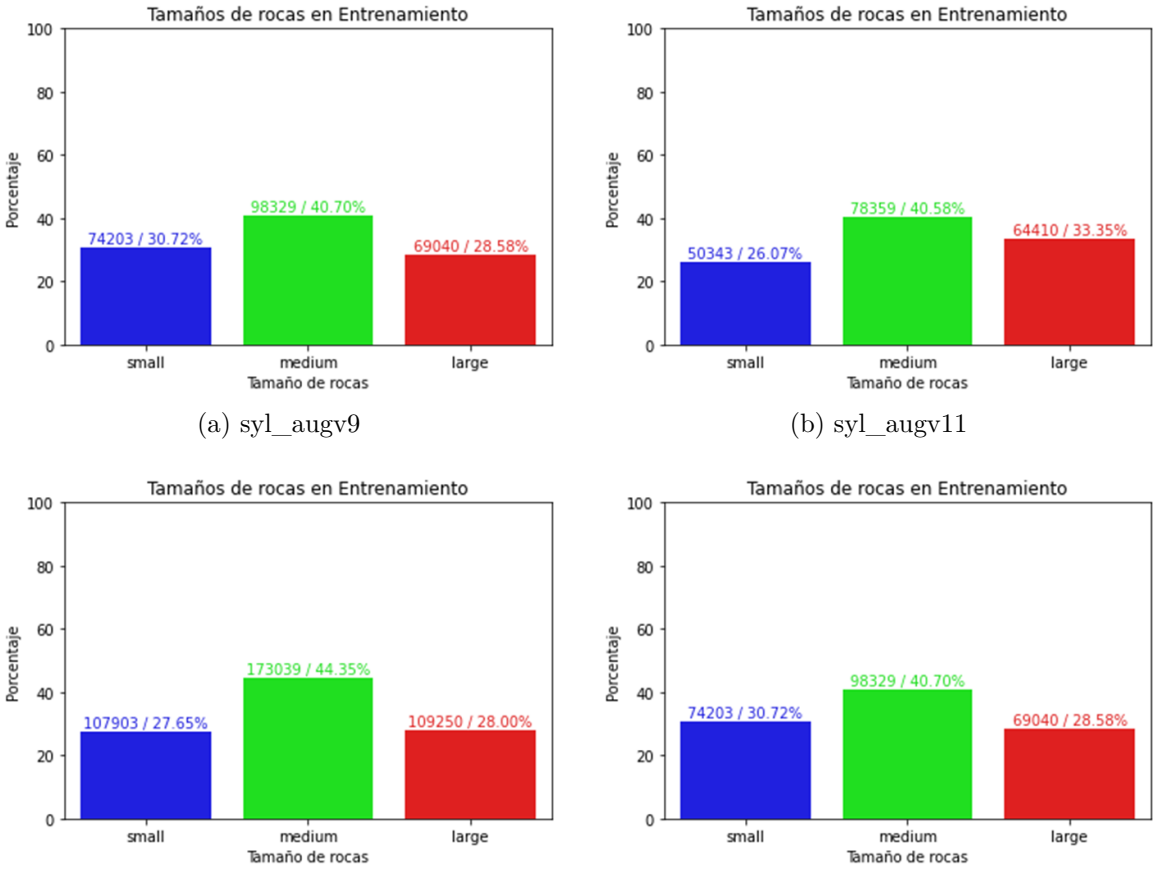


(c) Caso RC.8.7

Figura 5.3: Comparación de segmAP entre original y sus variaciones de porcentaje de aplicación.

5.3.2. Modificando Base de Datos de entrenamiento

La segunda modificación realizada es sobre la base de datos de entrenamiento. Se aplican diferentes métodos para aumentar y balancear la cantidad de rocas dentro del entrenamiento, buscando una mejora en los resultados. Se espera que, al balancear las rocas dentro de la base de datos de entrenamiento, el modelo sea capaz de aprender mejor a reconocer todos los tamaños existentes, mejorando los resultados tanto generales como específicos a cada tamaño. En la figura 5.4 se pueden ver las cantidades de rocas de tamaños diferentes que hay en cada una de las versiones diferentes de experimentos realizados. A continuación se explica mejor como se generó cada una de estas bases de entrenamiento editadas.



(a) syl_augv9

(b) syl_augv11

(c) syl_augv12

(d) syl_augv13

Figura 5.4: Gráficos que indican, según porcentaje, los distintos tamaños de rocas dentro de cada conjunto de entrenamiento editado. Sobre cada barra se encuentra el valor total de rocas, acompañado del porcentaje exacto.

- (a) syl_augv9: Se eligieron 12 imágenes que tuvieran más rocas grandes y una menor cantidad de rocas medianas. A todas esas imágenes se les aplicaron rotaciones con reflejo (visto en la figura 2.12.(b)) entre $\pm 15^{o}$ una cantidad de 10 veces, por lo que hay varias imágenes repetidas.
- (b) syl_augv11: Se hace lo mismo que en el punto anterior (a), sin embargo, las rotaciones implementadas cortan los bordes una vez inicia el reflejo (visto en la figura 2.12.(c)). Debido a lo anterior, las imágenes modifican su tamaño.
- (c) syl_augv12: En este caso se implementa lo mismo que en el punto anterior (b), pero se agregan 7 imágenes más a la lista.
- (d) syl_augv13: Este caso es igual al del punto (a), pero se añade el método de aumentación de datos del contraste aleatorio. De esta manera se prueba la edición previa al entrenamiento de las imágenes.

Observando los datos en las tablas 5.13 y 5.14 se puede ver claramente que no existe una mejora en comparación a utilizar la base de entrenamiento original, incluso empeoran en comparación al caso base.

Tabla 5.13: Resultados de "bbox" en formato COCO para variaciones de la base de entrenamiento de mejores resultados de RC.8.6.

Resultado	AP	AP50	AP75	APs	APm	APl
RC.8.6	46.9280 ±0.2009	70.7624 ±0.3980	51.9230 ±0.3586	36.8402 ±0.3887	66.7748 ±0.2749	83.3792 ±0.5511
RC.8.6_syl augv9	42.9844 ±0.1998	68.0024 ±0.4071	46.9348 ±0.6049	31.2548 ±0.1970	63.7258 ±0.3943	81.5660 ±0.7340
RC.8.6_syl augv11	41.4056 ±0.4001	66.5104 ±0.8567	44.8900 ±0.6772	28.9914 ±0.6503	63.2178 ±0.5006	82.0712 ±0.6808
RC.8.6_syl augv12	43.2130 ±0.1853	69.0708 ±0.4061	46.6708 ±0.4894	32.8752 ±0.2705	62.2712 ±0.2465	82.7962 ±0.8386
RC.8.6_syl augv13	43.3270 ±0.4135	68.0854 ±0.7348	47.3018 ±0.7389	31.7592 ±0.4530	64.0752 ±0.7903	82.5178 ±0.9724

Tabla 5.14: Resultados de segmentación en formato COCO para variaciones de la base de entrenamiento de mejores resultados de RC.8.6.

Resultado	AP	AP50	AP75	APs	APm	APl
RC.8.6	46.6848 ±0.1539	70.7810 ±0.4972	52.1028 ±0.3954	32.7740 ±0.3350	68.5340 ±0.1238	88.7802 ±0.4900
RC.8.6_syl augv9	41.3640 ±0.2799	67.5948 ±0.4263	43.6242 ±0.6714	25.7964 ±0.2842	64.0458 ±0.3045	87.2692 ±0.6464
RC.8.6_syl aug11	39.6498 ±0.3026	66.2294 ±0.7527	41.3914 ±0.4455	24.0982 ±0.5905	62.7336 ±0.4429	86.6464 ±0.4507
RC.8.6_syl augv12	41.3504 ±0.1931	68.7768 ±0.3629	43.6618 ±0.1848	26.9732 ±0.1542	63.4184 ±0.4479	86.8762 ±0.5620
RC.8.6_syl augv13	41.6672 ±0.3604	67.7124 ±0.7299	44.5054 ±0.7908	26.2306 ±0.4331	64.4836 ±0.4795	87.3978 ±0.2092

La figura 5.5 presenta de forma gráfica los resultados de segmAP obtenidos de cada experimento, en donde se ve claramente que los resultados son peor que su caso original con la base de entrenamiento sin edición. De todas maneras, en el anexo C se encuentra la tabla C.1 con los resultados de la realización de la prueba de Tukey para tener un mayor detalle de la comparación de los resultados.

A pesar de estar entrenando con más ejemplos de cada tamaño de roca, los resultados empeoran más de lo esperado, contradiciendo lo que se esperaba. Podría ser necesario que, debido a que se añade una mayor cantidad de datos para el entrenamiento, sea necesario utilizar más iteraciones durante el mismo, en comparación a las 4000 iteraciones utilizadas. También es probable que, debido a que muchas imágenes eran similares, el modelo terminara sobre ajustándose a las imágenes de entrenamiento, generando el comportamiento indeseado en la red. Sería necesario probar un método que balancee las rocas de forma que las imágenes generadas sean diferentes entre si. Así se podría confirmar si tener más ejemplos de rocas ayudaría a mejorar los resultados finales con la base de datos actual.

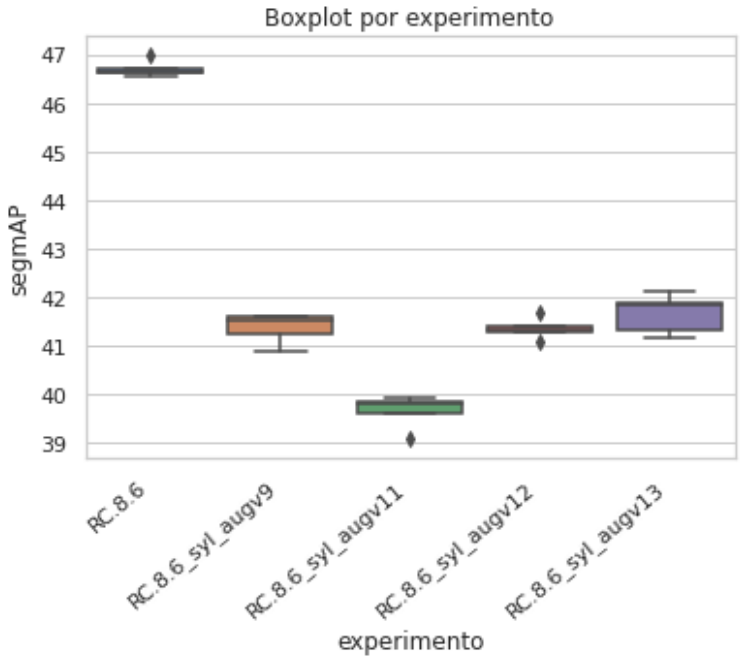


Figura 5.5: Comparación de segmAP entre original y sus variaciones de la base de entrenamiento.

5.4. Otros resultados

Finalmente se presentan algunos de los otros mejores resultados obtenidos aplicando diferentes tipos de DA. Estos resultados se pueden observar en las tablas 5.15 y 5.16. Los resultados corresponden, además del caso base y de uno de los mejores con RC, a la aplicación de brillo aleatorio (**RB**), saturación aleatoria (**RS**) y contraste aleatorio (**RCo**). Los números que acompañan a cada sigla son los límites en los que se puede modificar la característica correspondiente. Por ejemplo **RS.4,1.6** corresponde a un DA en el cual, a cada imagen durante el entrenamiento, se modifica la saturación entre un rango de 0.4 - 1.6 de la saturación original.

Un vistazo a los resultados obtenidos muestra que, para algunos parámetros, existe una leve mejora en comparación al caso base. Sin embargo, ninguna de estas mejoras es tan significativa como para asemejarse a uno de los mejores casos obtenidos con RC. De todas maneras se probó una combinación con los DA que mejor resultado daban, la cual resultó entregar valores similares a los mejores modelos.

Tabla 5.15: Resultados de "bbox" en formato COCO para resultados con otros tipos de DA.

Resultado	AP	AP50	AP75	APs	APm	APl
Sin DA	44.6406 ±0.1001	71.2282 ±0.3181	49.1748 ±0.4142	34.8062 ±0.1379	62.2882 ±0.2707	76.6650 ±1.5528
RC.8.6	46.9280 ±0.2009	70.7624 ±0.3980	51.9230 ±0.3586	36.8402 ±0.3887	66.7748 ±0.2749	83.3792 ±0.5511
RB.7,1.3	44.9018 ±0.2570	70.9934 ±0.4740	49.7234 ±0.4684	35.2320 ±0.2037	62.6386 ±0.2655	76.1666 ±0.7497
RB.6,1.3	44.9786 ±0.2975	70.9242 ±0.3657	50.0670 ±0.3606	35.2588 ±0.3927	62.8874 ±0.1776	76.4490 ±0.9658
RS.4,1.3	45.2058 ±0.1178	71.8556 ±0.4936	50.3978 ±0.4007	35.4468 ±0.2296	62.5404 ±0.2381	78.3980 ±0.6523
RS.7,1.6	45.3156 ±0.3050	71.8888 ±0.5128	50.0044 ±0.2753	35.6448 ±0.2476	62.5762 ±0.3378	76.8148 ±1.1402
RCo.4,1.3	44.7946 ±0.1648	70.7302 ±0.1937	49.6702 ±0.3553	34.9142 ±0.1349	63.2522 ±0.4492	75.6666 ±0.9355
RCo.7,1.6	44.7682 ±0.1708	70.7978 ±0.0974	49.4752 ±0.2740	35.3154 ±0.2265	62.9876 ±0.4153	75.2038 ±0.7465
RC.8.6_prob.7 RCo.7,1.6 RS.7,1.6	46.0188 ±0.2798	69.7570 ±0.4219	51.0562 ±0.4663	35.9960 ±0.1991	65.9886 ±0.2331	83.3172 ±0.9082

Tabla 5.16: Resultados de segmentación en formato COCO para resultados con otros tipos de DA.

Resultado	AP	AP50	AP75	APs	APm	APl
Sin DA	43.6206 ±0.1043	70.7066 ±0.3586	47.2988 ±0.5425	29.6324 ±0.2118	64.5204 ±0.2121	86.0506 ±0.3677
RC.8.6	46.6848 ±0.1539	70.7810 ±0.4972	52.1028 ±0.3954	32.7740 ±0.3350	68.5340 ±0.1238	88.7802 ±0.4900
RB.7,1.3	43.8342 ±0.1608	70.5900 ±0.3772	47.5076 ±0.1702	30.4014 ±0.2196	64.3766 ±0.2316	85.3050 ±0.3545
RB.6,1.3	43.9218 ±0.1824	70.6762 ±0.3698	47.6204 ±0.3658	30.3844 ±0.3364	64.6262 ±0.1904	85.7370 ±0.5905
RS.4,1.3	44.2308 ±0.0976	71.2582 ±0.1534	48.5542 ±0.2375	30.2988 ±0.1242	65.0112 ±0.2783	87.2062 ±0.5145
RS.7,1.6	44.3982 ±0.2559	71.3314 ±0.1209	48.4128 ±0.5849	30.5110 ±0.2764	65.4500 ±0.3121	85.2250 ±0.1897
RCo.4,1.3	43.7810 ±0.1210	70.1778 ±0.4496	47.1686 ±0.2445	30.0820 ±0.2057	64.4710 ±0.2572	85.3832 ±0.4215
RCo.7,1.6	43.9306 ±0.2379	70.5308 ±0.0435	47.6798 ±0.5682	30.4000 ±0.1283	64.2590 ±0.4610	86.4176 ±0.3726
RC.8.6_prob.7 RCo.7,1.6 RS.7,1.6	45.7508 ±0.3361	69.1838 ±0.4493	51.5718 ±0.4610	32.0130 ±0.3556	67.4474 ±0.1954	88.2102 ±0.4530

Esa diferencia se puede notar bastante al observar la figura 5.6, donde el caso RC.8.6 se encuentra alejado de las demás cajas. Por lo que se puede afirmar que el mejor método de DA, de los probados, que se puede aplicar durante el entrenamiento, corresponde al RC. Para reafirmar la conclusión anterior se tiene, en el anexo D, las tablas D.1 y D.2 con los resultados de la prueba de Tukey realizada.

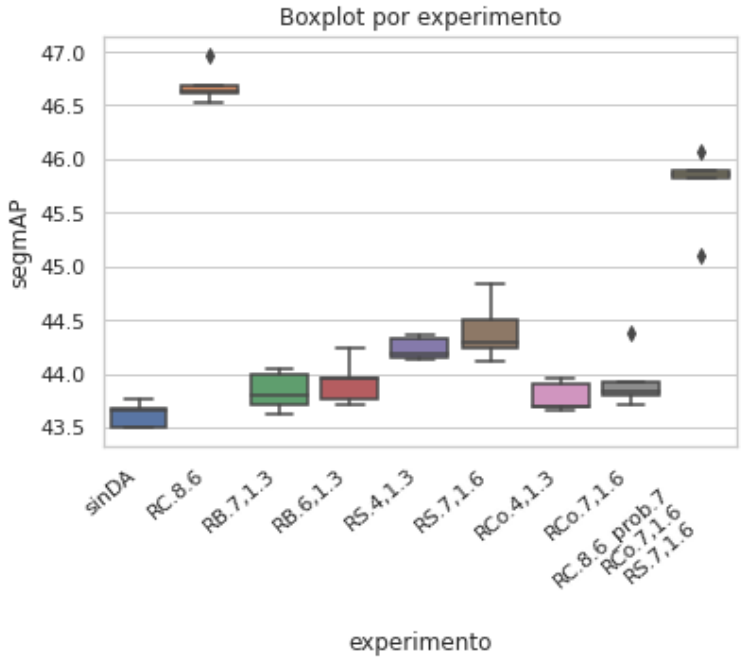


Figura 5.6: Comparación de segmAP entre el caso base, mejor resultado y otros métodos de DA.

5.5. Resultado General

La tabla 5.17 muestra los resultados de precisión para distintos tamaños calculados en todas las imágenes de prueba, utilizando el cálculo de precisión explicado al final de la sección 4.2.2. De esa tabla se puede extraer que las rocas que mejor son segmentadas corresponden a las medianas, que son las que más están representadas en la base de datos de entrenamiento (recordar figura 4.5). Sin embargo, a pesar que las rocas pequeñas son las segundas con mayor representación, son las que peor son segmentadas. Esto se puede deber a la dificultad de encontrar rocas pequeñas dentro de las imágenes. A pesar de ello, el porcentaje de segmentación general es bastante alto.

Tabla 5.17: Resultados de precisión del mejor experimento aplicando RC con una probabilidad de 0.9. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Resultado	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
RC.8.6_prob.9	74.4651 ±0.1178	91.2474 ±0.0639	85.9375 ±0.3303	83.8319 ±0.0636

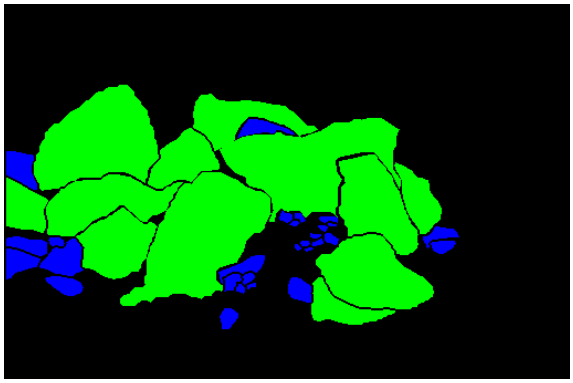
5.6. Resultados Detallados

En la presente sección se mostrarán en mayor detalle algunos de los resultados obtenidos para el experimento **RC.8.6_prob.9**. De esta manera se podrá verificar de forma cualitativa la funcionalidad del método. Se mostrarán algunas imágenes de prueba junto a su "Ground Truth" (GT) y su predicción. Las rocas pequeñas se verán representadas por un color azul, las

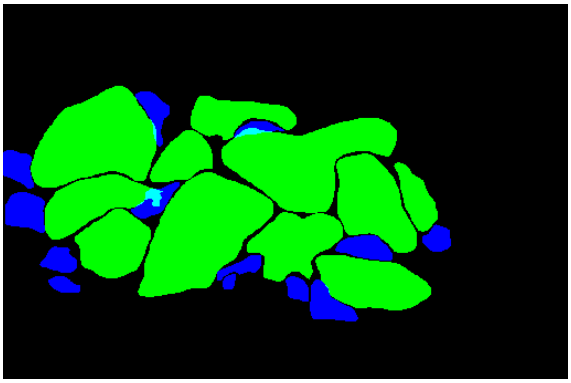
de tamaño medio serán verdes y las grandes serán de color rojo. Adicionalmente se indicarán los resultados COCO y el valor de precisión medido en cada imagen.

En las siguientes figuras se podrá observar de forma visual como segmenta el modelo implementado a las imágenes de prueba. En su gran mayoría, las rocas tienen una correcta segmentación, incluso logra segmentar rocas que no se encontraban en el GT. Sin embargo hay veces en las que se equivoca en el tamaño, segmenta rocas donde no hay, separa una roca o une rocas diferentes.

(a) Imagen



(b) GT



(c) Predicción

Figura 5.7: Imagen 65 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

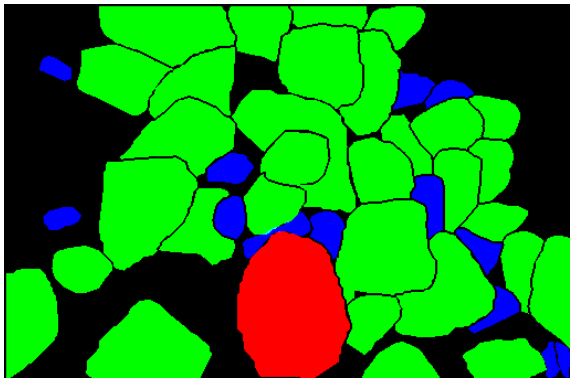
Tabla 5.18: Resultados COCO de la imagen 65. El - indica que no hay resultados relacionados a esa métrica.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	29.960	48.182	27.213	16.551	66.819	-
Segm	31.074	45.298	38.397	11.870	73.478	-

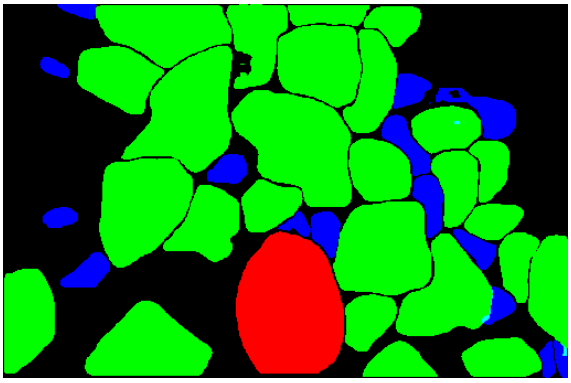
Tabla 5.19: Resultados de precisión calculados de la imagen 65. Resultados calculados con la precisión mencionada al final de la sección 4.2.2. El - indica que no hay resultados relacionados a esa métrica.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
65	75.000	91.667	-	80.556

(a) Imagen



(b) GT



(c) Predicción

Figura 5.8: Imagen 69 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

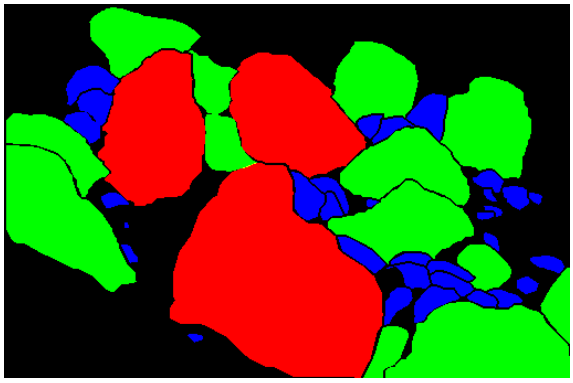
Tabla 5.20: Resultados COCO de la imagen 69.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	60.104	85.240	71.498	50.602	66.491	90.000
Segm	60.848	84.278	66.702	46.483	65.269	100.000

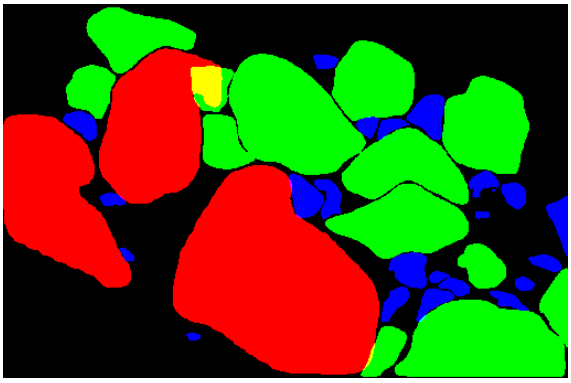
Tabla 5.21: Resultados de precisión calculados de la imagen 69. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
69	76.923	92.857	100.000	88.095

(a) Imagen



(b) GT



(c) Predicción

Figura 5.9: Imagen 72 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

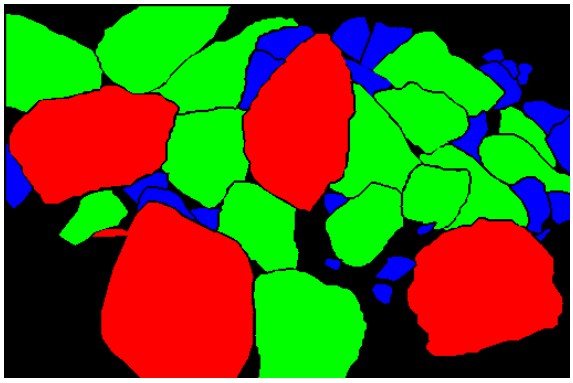
Tabla 5.22: Resultados COCO de la imagen 72.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	43.540	69.549	49.808	30.526	68.661	78.449
Segm	41.532	66.601	40.615	25.853	71.575	80.099

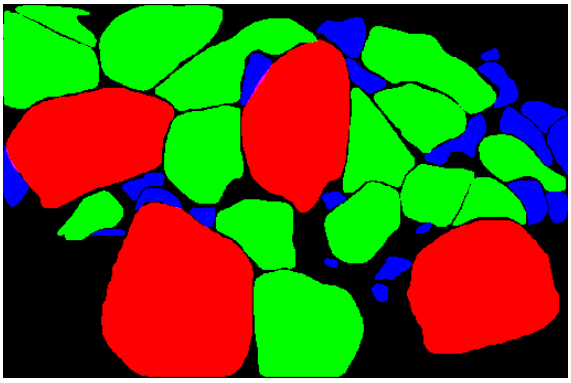
Tabla 5.23: Resultados de precisión calculados de la imagen 72. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
72	68.182	92.857	100.000	79.487

(a) Imagen



(b) GT



(c) Predicción

Figura 5.10: Imagen 76 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

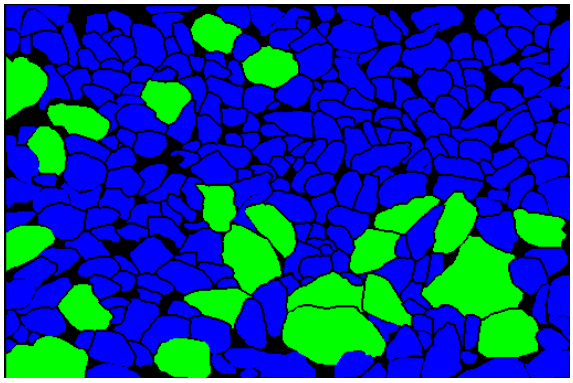
Tabla 5.24: Resultados COCO de la imagen 76.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	57.411	85.066	66.809	48.142	68.253	90.858
Segm	59.938	87.818	60.716	46.233	74.618	93.366

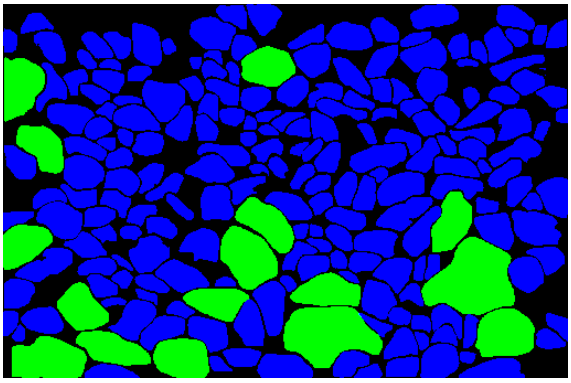
Tabla 5.25: Resultados de precisión calculados de la imagen 76. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
76	90.000	82.353	100.000	87.805

(a) Imagen



(b) GT



(c) Predicción

Figura 5.11: Imagen 78 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

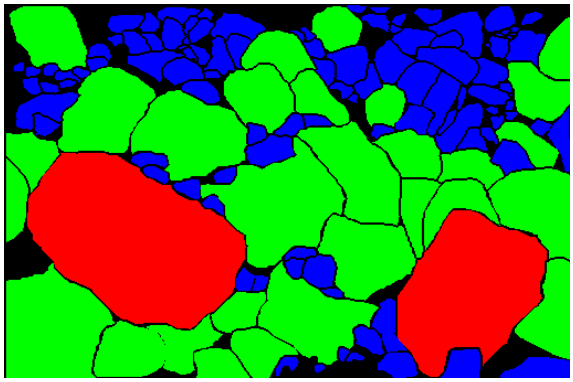
Tabla 5.26: Resultados COCO de la imagen 78. El - indica que no hay resultados relacionados a esa métrica.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	44.393	68.205	51.867	43.810	64.500	-
Segm	42.237	68.382	46.792	39.822	68.854	-

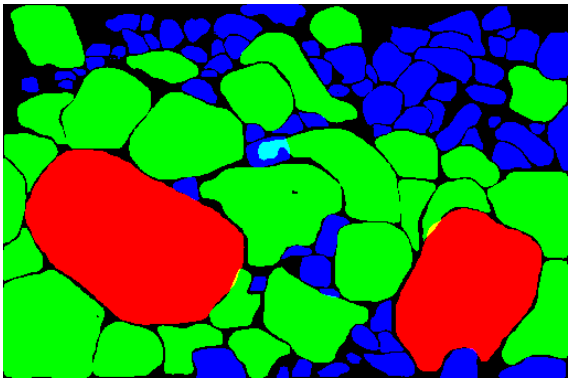
Tabla 5.27: Resultados de precisión calculados de la imagen 78. Resultados calculados con la precisión mencionada al final de la sección 4.2.2. El - indica que no hay resultados relacionados a esa métrica.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
78	88.832	95.238	-	89.450

(a) Imagen



(b) GT



(c) Predicción

Figura 5.12: Imagen 79 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

Tabla 5.28: Resultados COCO de la imagen 79.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	37.733	60.175	36.889	30.957	60.246	95.050
Segm	36.457	60.043	42.946	26.687	65.869	95.050

Tabla 5.29: Resultados de precisión calculados de la imagen 79. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
79	70.000	90.000	100.000	76.471

Entre estos ejemplos se pueden observar casos en los que una roca mediana fue partida en dos, casos en los cuales dos rocas distintas son segmentadas como una sola, como es el caso de la figura 5.8, otro caso en el cual una roca se encuentra sobre otra en el centro. Entre estos ejemplos se pueden observar la falta de detecciones a rocas pequeñas, entre otros errores. Un error que no se considera tan grave corresponde al cambio de tamaño de una roca segmentada. Una roca correctamente segmentada tiene un color diferente al que tiene asignado en el GT, indicando una diferencia en el tamaño. Esto se debe a que, probablemente, la roca se encuentre justo en el límite entre un tamaño y el siguiente/anterior. A pesar de ello, este problema se encuentra considerado en el cálculo de la precisión, pues existe una robustez a este límite de tamaño.

En los resultados también se observan buenas segmentaciones, por ejemplo en la figura 5.7, en la parte superior izquierda, la red fue capaz de segmentar una roca pequeña que no se encontraba en el GT. Lo mismo se aprecia en la figura 5.8, en la esquina superior izquierda, dónde se segmentó una roca pequeña que tampoco estaba en el GT.

El modelo incluso fue capaz de arreglar un error de segmentación del GT en la figura 5.10. En la roca grande de abajo a la izquierda, se observa una "protuberancia" en el lado superior izquierdo, correspondiente a una roca pequeña. El modelo fue capaz de separar este error de segmentación realizado en el GT. El resto de imágenes y resultados del conjunto de prueba se encuentra en el anexo E, en las que se pueden observar más segmentaciones.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Conclusión

La necesidad de detectar y segmentar los fragmentos de rocas para conocer su distribución de tamaño surge de la minería, donde es imprescindible conocer el tamaño de los fragmentos extraídos para optimizar los distintos procesos y evitar un costo mayor. Existen diferentes métodos para lograr conocer los tamaños de los fragmentos, sin embargo pocos corresponden a métodos con inteligencia computacional. Debido a ello se busca utilizar un método con CNNs que pueda segmentar cada instancia de roca dentro de imágenes digitales. Una vez que los fragmentos se encuentran segmentados, si se conoce la proporción de píxel a centímetros, se puede conocer el tamaño real de cada roca de manera rápida y sencilla, lo que lograría ayudar a la optimización de procesos de fragmentación dentro de la minería.

En los últimos años se han desarrollado algunos métodos que son capaces de segmentar las rocas. Sin embargo, varios de los trabajos revisados no cumplían con lo que se buscaba realizar. Algunos trabajos son poco replicables, otros realizaban su estudio en un ambiente controlado con rocas pequeñas, contrario a lo esperado en la minería. Es por ello que se desarrolló un método capaz de segmentar múltiples rocas de diferentes tamaños en un ambiente no controlado.

De todos los métodos probados, el mejor resultó ser el que utiliza una probabilidad de 0.9 de aplicar Random Crop durante el entrenamiento del modelo, entregando un resultado de segmAP cercano al 50 % y de precisión total mayor al 80 %. Con los resultados que se mostraron en las tablas 5.4 y 5.9, además de sus figuras y prueba de Tukey respectivas, y junto al resultado general y los resultados detallados mostrados, se demuestra que el método desarrollado es capaz de segmentar correctamente una gran cantidad de rocas de diferentes tamaños en imágenes digitales de un ambiente minero.

Debido a lo anterior, se considera que el primer objetivo específico mencionado fue cumplido, pues las imágenes utilizadas para probar el modelo consisten en imágenes de rocas dentro de un ambiente minero. El segundo y tercer objetivo mencionado también se cumplieron, pues se probaron diferentes hiperparámetros y múltiples métodos de DA diferentes durante la búsqueda del mejor modelo.

Por lo tanto, se concluye que el método desarrollado podría resultar útil para las mineras,

ya que serían capaces de obtener, en cortos lapsos de tiempo, la segmentación de rocas de diferentes tamaños en múltiples imágenes. Utilizando la transformación correcta para pasar la información de píxeles a centímetros, habrían sido capaces de obtener la información del tamaño de las rocas extraídas rápidamente, logrando evitar un mayor gasto de tiempo y un aumento en los costos.

6.2. Trabajo Futuro

Si bien el método implementado funciona correctamente, puesto que logra segmentar múltiples tamaños de rocas en un ambiente minero, aún se percibe potencial de mejora. En un futuro se podrían probar más variaciones de parámetros y de "backbones", además de otras metodologías de DA, o uniones de los que entregan mejores resultados. Durante el presente trabajo los experimentos se vieron limitados debido al tipo de hardware utilizado, el cual ponía limitaciones al tiempo de uso.

Adicionalmente, se considera necesario tener una base de datos de mayor tamaño, que contenga más ejemplos de rocas y se encuentren bien niveladas según su distribución de tamaño. Un detalle importante a tener en cuenta en esas imágenes es que el ambiente no sea sólo de exteriores, si no que también incluya ambientes subterráneos, para tener otros ejemplos de ambientes mineros diferentes a los utilizados en el presente trabajo, los cuales eran con iluminación exterior.

Finalmente, se podrían evaluar diferentes alternativas de uso de múltiples arquitecturas para mejorar la segmentación. Es decir, utilizar diferentes modelos entrenados para tamaños específicos de rocas. Así, al unir los resultados, se podrían obtener mejores segmentaciones por cada tipo de roca dentro de las imágenes, pues cada modelo se centrará en un tamaño específico. Asimismo, se podría evaluar el uso de neuroevolución, para así obtener una red que se ajuste mejor al problema.

Capítulo 7

Bibliografía

[1] Morin, M. A. y Ficarazzo, F., "Monte carlo simulation as a tool to predict blasting fragmentation based on the kuz-ram model," Computers and Geosciences, vol. 32, pp. 352–359, 2006.

[2] Monjezi, M., Rezaei, M., y Varjani, A. Y., "Prediction of rock fragmentation due to blasting in gol-e-gohar iron mine using fuzzy logic," International Journal of Rock Mechanics and Mining Sciences, vol. 46, pp. 1273–1280, 2009.

[3] Michaux, S. y Djordjevic, N., "Influence of explosive energy on the strength of the rock fragments and sag mill throughput," Minerals Engineering, vol. 18, pp. 439–448, 2005.

[4] Bamford, T., Esmaeili, K., y Schoellig, A. P., "A deep learning approach for rock fragmentation analysis," International Journal of Rock Mechanics and Mining Sciences, vol. 145, pp. 1–13, 2021.

[5] Shrivastava, S., Deb, D., y Bhattacharjee, S., "Prediction of particle size distribution curves of dump materials using convolutional neural networks," Rock Mechanics and Rock Engineering, vol. 55, pp. 471–479, 2022.

[6] Géron, A., Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. O'Reilly Media, Inc., 2019.

[7] O'Shea, K. y Nash, R., "An introduction to convolutional neural networks," CoRR, vol. abs/1511.08458, 2015, http://arxiv.org/abs/1511.08458.

[8] Goodfellow, I., Bengio, Y., y Courville, A., Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[9] LeCun, Y., Haffner, P., Bottou, L., y Bengio, Y., "Object recognition with gradient-based learning," Shape, Contour and Grouping in Computer Vision. Lecture Notes in Computer Science, vol. 1681, pp. 319–345, 1999.

[10] Ying, X., "An overview of overfitting and its solutions," Journal of Physics: Conference Series, 2019.

[11] Shorten, C. y Khoshgoftaar, T. M., "A survey on image data augmentation for deep learning," Journal of Big Data, 2019.

[12] Si, C., Zhu, P., Di, K., y Wang, R., "Data for: Rock fragment image segmentation combining cnn and watershed algorithm." Mendeley Data, 2019.

[13] Hafiz, A. M. y Bhat, G. M., "A survey on instance segmentation: state of the art,"

International Journal of Multimedia Information Retrieval, vol. 9, pp. 171–189, 2020.

[14] Galdames, F. J., Perez, C. A., Estévez, P. A., y Adams, M., "Rock lithological instance classification by hyperspectral images using dimensionality reduction and deep learning," Chemometrics and Intelligent Laboratory Systems, vol. 224, 2022.

[15] Li, H., Asbjörnsson, G., y Lindqvist, M., "Image process of rock size distribution using dexined-based neural network," Minerals, 2021.

[16] Qiao, W., Zhao, Y., Xu, Y., Lei, Y., Wang, Y., Yu, S., y Li, H., "Deep learning-based pixel-level rock fragment recognition during tunnel excavation using instance segmentation model," Tunneling and Unferground Space Technology incorporating Trenchless Technology Research, vol. 115, 2021.

[17] Tuan, N. M., Kim, Y., Lee, J.-Y., y Chin, S., "Automatic stereo vision-based inspection system for particle shape analysis of coarse aggregates," Journal of Computing in Civil Engineering, vol. 36, 2022.

[18] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., y Lin, D., "MMDetection: Open mmlab detection toolbox and benchmark," arXiv preprint arXiv:1906.07155, 2019.

[19] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., y Girshick, R., "Detectron2." https://github.com/facebookresearch/detectron2, 2019.

[20] He, K., Gkioxari, G., Dollar, P., y Girshick, R., "Mask r-cnn," en Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017.

[21] Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., y Belongie, S., "Feature pyramid networks for object detection," en Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[22] He, K., Zhang, X., Ren, S., y Sun, J., "Deep residual learning for image recognition," en Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[23] Ren, S., He, K., Girshick, R., y Sun, J., "Faster r-cnn: Towards real-time object detection with region proposal networks," en Advances in Neural Information Processing Systems (Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., y Garnett, R., eds.), vol. 28, Curran Associates, Inc., 2015, https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

Anexo A

Resultados de prueba de Tukey para el caso base

La siguiente tabla representa la prueba de Tukey realizado para los resultados de segmAP del caso base con los experimentos de "shubham", junto a sus versiones con mejor segmAP y pérdida de validación.

Tabla A.1: Prueba de Tukey entre el caso base, los hiperparámetros de [5] y sus versiones con mejor segmAP y pérdida de validación.

Grupo 1	Grupo 2	p-adj	Rechaza
shubham	shubham(segmAP)	0.0017	Verdad
shubham	shubham(valloss)	0.0017	Verdad
shubham	Sin DA	0.9000	Falso
shubham	Sin DA(segmAP)	0.0159	Verdad
shubham	Sin DA(valloss)	0.6915	Falso
shubham(segmAP)	shubham(valloss)	0.9000	Falso
shubham(segmAP)	Sin DA	0.0108	Verdad
shubham(segmAP)	Sin DA(segmAP)	0.9000	Falso
shubham(segmAP)	Sin DA(valloss)	0.0480	Verdad
shubham(valloss)	Sin DA	0.0108	Verdad
shubham(valloss)	Sin DA(segmAP)	0.9000	Falso
shubham(valloss)	Sin DA(valloss)	0.0480	Verdad
Sin DA	Sin DA(segmAP)	0.0851	Falso
Sin DA	Sin DA(valloss)	0.9000	Falso
Sin DA(segmAP)	Sin DA(valloss)	0.2821	Falso

Anexo B

Resultados de prueba de Tukey para la modificación de porcentaje de RC

A continuación se adjuntan las tablas con los resultados obtenidos de las pruebas de Tukey, realizados sobre los datos obtenidos de la modificación del porcentaje de aplicación de RC.

Tabla B.1: Prueba de Tukey entre el mejor resultado relacionado y variaciones de la probabilidad de aplicar RC.7.7.

Grupo 1	Grupo 2	p-adj	Rechaza
RC.7.7	RC.7.7_prob.3	0.0067	Verdad
RC.7.7	RC.7.7_prob.5	0.1880	Falso
RC.7.7_prob.3	RC.7.7_prob.5	0.1777	Falso

Tabla B.2: Prueba de Tukey entre el mejor resultado relacionado y variaciones de la probabilidad de aplicar RC.8.7.

Grupo 1	Grupo 2	p-adj	Rechaza
RC.8.7	RC.8.7_prob.3	0.0010	Verdad
RC.8.7	RC.8.7_prob.5	0.0023	Verdad
RC.8.7_prob.3	RC.8.7_prob.5	0.0095	Verdad

Anexo C

Resultados de prueba de Tukey para la modificación de la base de entrenamiento

Los siguientes resultados corresponden a la tabla con los resultados obtenidos de la prueba de Tukey, realizada sobre los datos obtenidos de la modificación de la base de entrenamiento, usando RC al entrenar.

Tabla C.1: Prueba de Tukey entre el mejor resultado relacionado y variaciones de la base de entrenamiento al usar RC.8.6.

Grupo 1	Grupo 2	p-adj	Rechaza
RC.8.6	RC.8.6_syl_augv11	0.0010	Verdad
RC.8.6	RC.8.6_syl_augv12	0.0010	Verdad
RC.8.6	RC.8.6_syl_augv13	0.0010	Verdad
RC.8.6	RC.8.6_syl_augv9	0.0010	Verdad
RC.8.6_syl_augv11	RC.8.6_syl_augv12	0.0010	Verdad
RC.8.6_syl_augv11	RC.8.6_syl_augv12	0.0010	Verdad
RC.8.6_syl_augv11	RC.8.6_syl_augv9	0.0010	Verdad
RC.8.6_syl_augv12	RC.8.6_syl_augv13	0.4754	Falso
RC.8.6_syl_augv12	RC.8.6_syl_augv9	0.9000	Falso
RC.8.6_syl_augv13	RC.8.6_syl_augv9	0.5147	Falso

Anexo D

Resultados de prueba de Tukey para otros métodos de DA

Las siguientes tablas representan los resultados obtenidos de la prueba de Tukey realizada sobre los datos obtenidos al entrenar con diferentes tipos de DA. Debido a la extensa cantidad de resultados, se tuvo que separar en dos.

Tabla D.1: Prueba de Tukey entre el caso base, mejor resultado y otros métodos de DA, parte 1.

Grupo 1	Grupo 2	p-adj	Rechaza
RB.6,1.3	RB.7,1.3	0.9000	Falso
RB.6,1.3	RC.8.6	0.0010	Verdad
RB.6,1.3	RC.8.6_prob.7 RCo.7,1.6_RS.7,1.6	0.0010	Verdad
RB.6,1.3	RCo.4,1.3	0.9000	Falso
RB.6,1.3	RCo.7,1.6	0.9000	Falso
RB.6,1.3	RS.4,1.3	0.4257	Falso
RB.6,1.3	RS.7,1.6	0.0388	Verdad
RB.6,1.3	Sin DA	0.4593	Falso
RB.7,1.3	RC.8.6	0.0010	Verdad
RB.7,1.3	RC.8.6_prob.7 RCo.7,1.6_RS.7,1.6	0.0010	Verdad
RB.7,1.3	RCo.4,1.3	0.9000	Falso
RB.7,1.3	RCo.7,1.6	0.9000	Falso
RB.7,1.3	RS.4,1.3	0.1408	Falso
RB.7,1.3	RS.7,1.6	0.0076	Verdad
RB.7,1.3	Sin DA	0.8158	Falso
RC.8.6	RC.8.6_prob.7 RCo.7,1.6_RS.7,1.6	0.0010	Verdad
RC.8.6	RCo.4,1.3	0.0010	Verdad
RC.8.6	RCo.7,1.6	0.0010	Verdad
RC.8.6	RS.4,1.3	0.0010	Verdad
RC.8.6	RS.7,1.6	0.0010	Verdad
RC.8.6	Sin DA	0.0010	Verdad

Tabla D.2: Prueba de Tukey entre el caso base, mejor resultado y otros métodos de DA, parte 2.

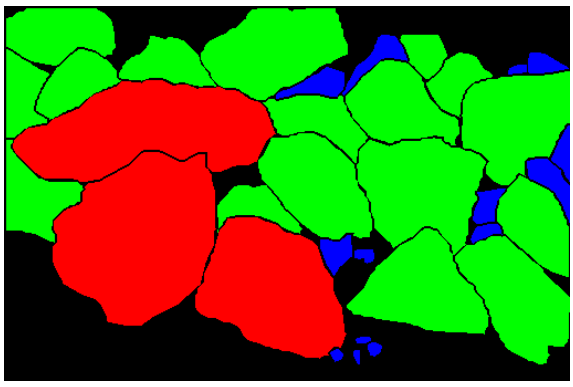
Grupo 1	Grupo 2	p-adj	Rechaza
RC.8.6_prob.7 RCo.7,1.6_RS.7,1.6	RCo.4,1.3	0.0010	Verdad
RC.8.6_prob.7 RCo.7,1.6_RS.7,1.6	RCo.7,1.6	0.0010	Verdad
RC.8.6_prob.7 RCo.7,1.6_RS.7,1.6	RS.4,1.3	0.0010	Verdad
RC.8.6_prob.7 RCo.7,1.6_RS.7,1.6	RS.7,1.6	0.0010	Verdad
RC.8.6_prob.7 RCo.7,1.6_RS.7,1.6	Sin DA	0.0010	Verdad
RCo.4,1.3	RCo.7,1.6	0.9000	Falso
RCo.4,1.3	RS.4,1.3	0.4498	Falso
RCo.4,1.3	RS.7,1.6	0.0026	Verdad
RCo.4,1.3	Sin DA	0.9000	Falso
RCo.7,1.6	RS.4,1.3	0.4636	Falso
RCo.7,1.6	RS.7,1.6	0.0452	Verdad
RCo.7,1.6	Sin DA	0.4213	Falso
RS.4,1.3	RS.7,1.6	0.9000	Falso
RS.4,1.3	Sin DA	0.0030	Verdad
RS.7,1.6	Sin DA	0.0010	Verdad

Anexo E

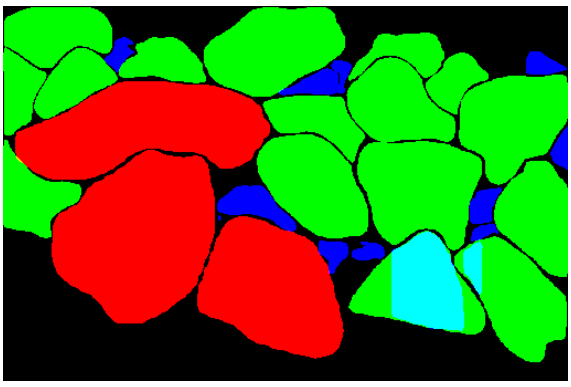
Resultados Detallados

En este anexo se pueden observar el resto de resultados del conjunto de prueba al ser ingresados al mejor modelo. Se observan las imágenes originales, su GT y su predicción. Adicionalmente se pueden ver los resultados de cada imagen con las métricas entregadas por COCO y la métrica de precisión implementada, explicada en al final de la sección 4.2.2.

(a) Imagen



(b) GT



(c) Predicción

Figura E.1: Imagen 66 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

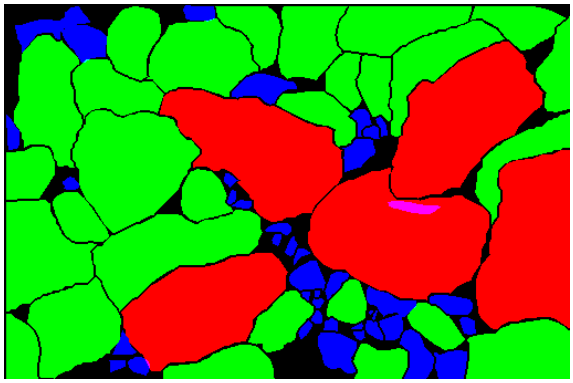
Tabla E.1: Resultados COCO de la imagen 66.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	62.059	75.930	69.310	29.347	84.574	92.653
Segm	60.579	75.930	72.501	25.517	82.686	94.422

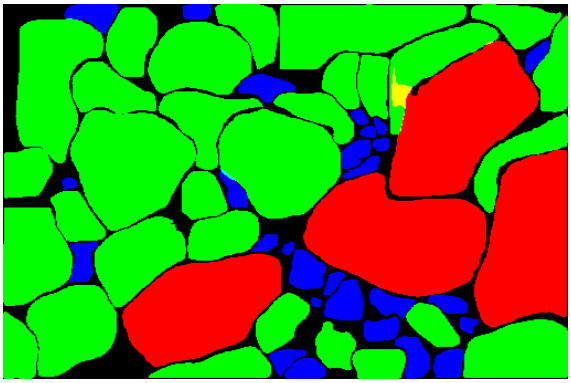
Tabla E.2: Resultados de precisión calculados de la imagen 66. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
66	60.000	94.118	100.000	83.333

(a) Imagen



(b) GT



(c) Predicción

Figura E.2: Imagen 67 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

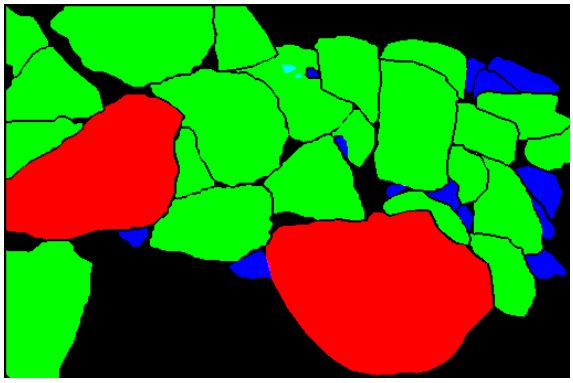
Tabla E.3: Resultados COCO de la imagen 67.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	40.012	65.167	41.018	24.136	58.415	80.668
Segm	41.991	65.167	42.592	22.225	63.639	80.099

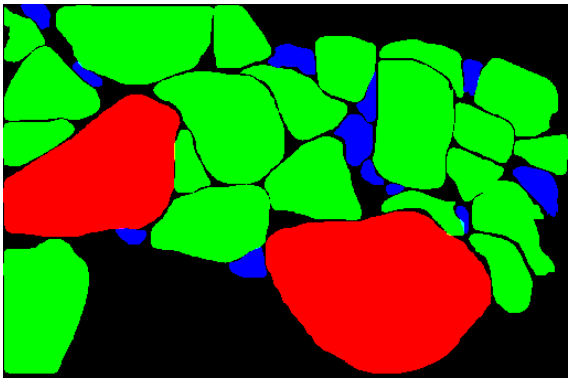
Tabla E.4: Resultados de precisión calculados de la imagen 67. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
67	55.172	83.333	100.000	71.429

(a) Imagen



(b) GT



(c) Predicción

Figura E.3: Imagen 68 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

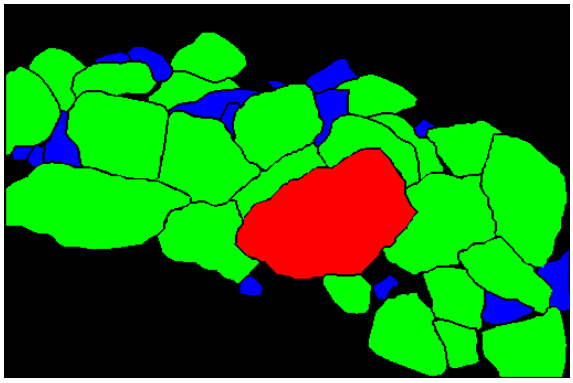
Tabla E.5: Resultados COCO de la imagen 68.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	48.475	68.640	53.225	13.878	66.818	100.000
Segm	51.104	68.373	54.792	13.894	71.703	100.000

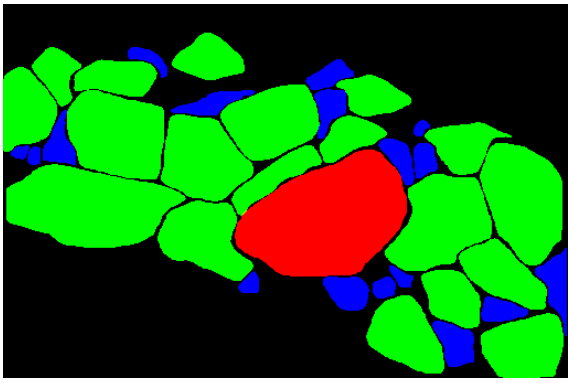
Tabla E.6: Resultados de precisión calculados de la imagen 68. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
68	50.000	95.455	100.000	80.556

(a) Imagen



(b) GT



(c) Predicción

Figura E.4: Imagen 70 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

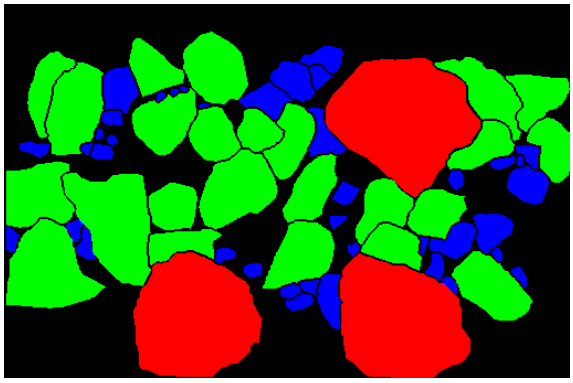
Tabla E.7: Resultados COCO de la imagen 70.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	59.958	81.683	71.374	49.060	67.133	90.000
Segm	63.178	86.132	71.223	42.624	72.155	100.000

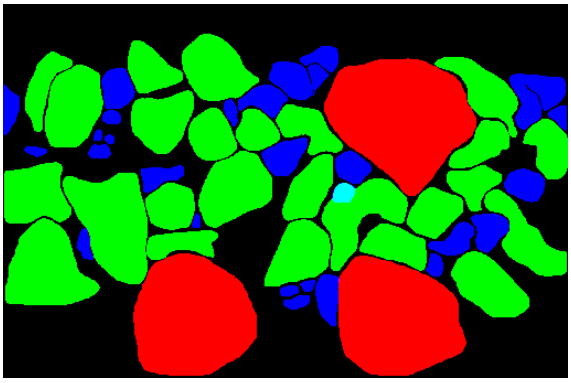
Tabla E.8: Resultados de precisión calculados de la imagen 70. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
70	80.000	95.455	100.000	89.474

(a) Imagen



(b) GT



(c) Predicción

Figura E.5: Imagen 71 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

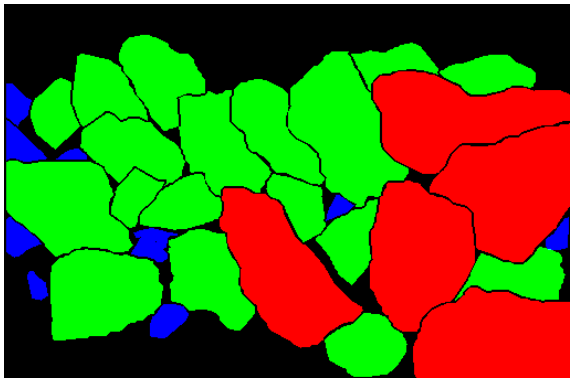
Tabla E.9: Resultados COCO de la imagen 71.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	46.244	66.695	54.888	32.013	65.295	100.000
Segm	46.840	64.927	55.104	29.890	69.419	91.122

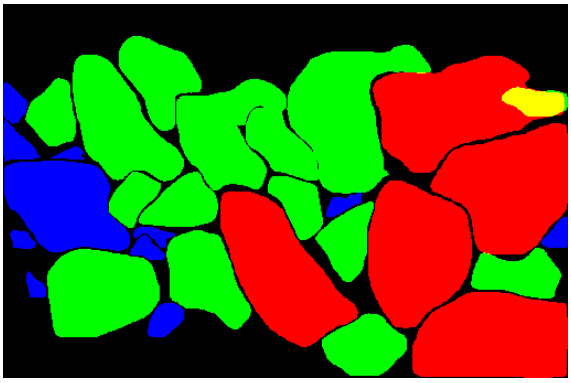
Tabla E.10: Resultados de precisión calculados de la imagen 71. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
71	64.516	84.000	100.000	74.576

(a) Imagen



(b) GT



(c) Predicción

Figura E.6: Imagen 73 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

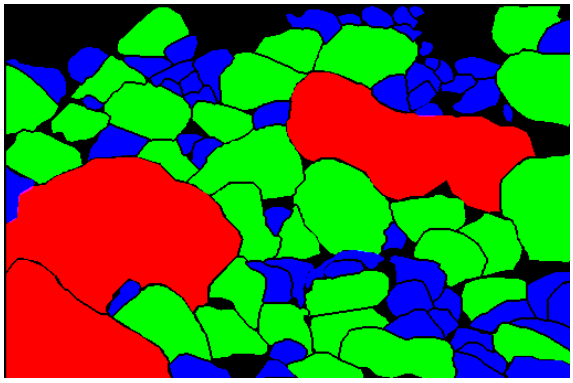
Tabla E.11: Resultados COCO de la imagen 73.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	56.440	83.226	61.700	52.550	61.894	68.918
Segm	55.590	86.114	58.713	53.647	55.195	79.785

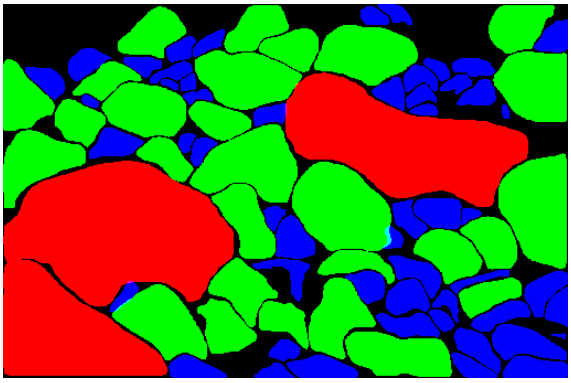
Tabla E.12: Resultados de precisión calculados de la imagen 73. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
73	80.000	76.471	100.000	81.250

(a) Imagen



(b) GT



(c) Predicción

Figura E.7: Imagen 74 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

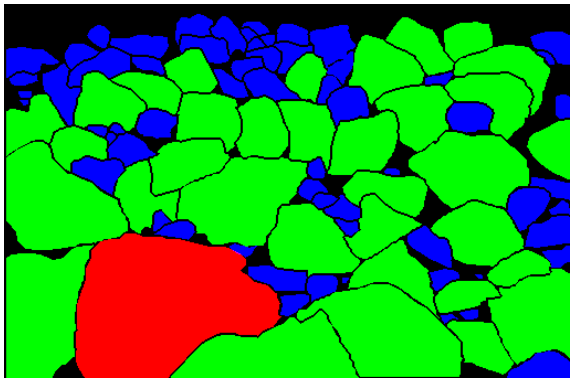
Tabla E.13: Resultados COCO de la imagen 74.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	48.406	79.977	53.485	43.753	58.150	88.911
Segm	48.637	77.594	54.823	38.352	60.598	90.000

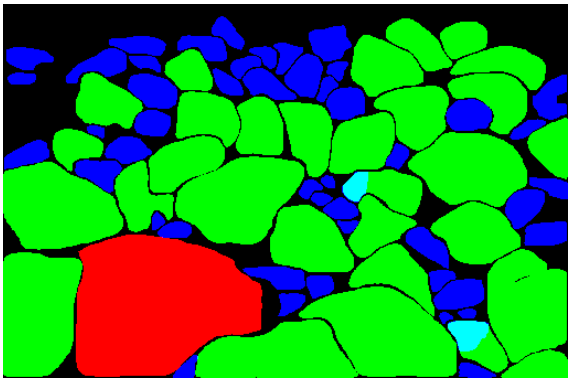
Tabla E.14: Resultados de precisión calculados de la imagen 74. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
74	78.431	90.323	100.000	83.529

(a) Imagen



(b) GT



(c) Predicción

Figura E.8: Imagen 75 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

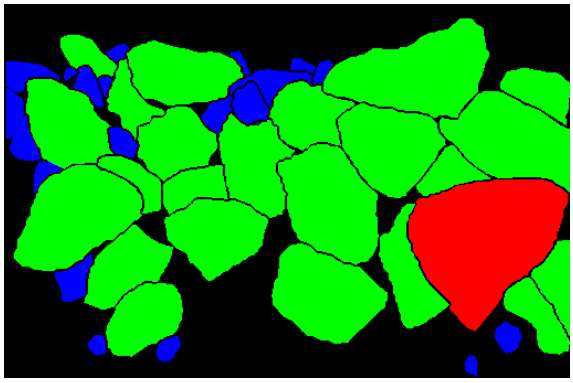
Tabla E.15: Resultados COCO de la imagen 75.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	52.093	74.231	58.361	40.492	71.963	90.000
Segm	52.935	74.231	65.113	37.070	78.985	90.000

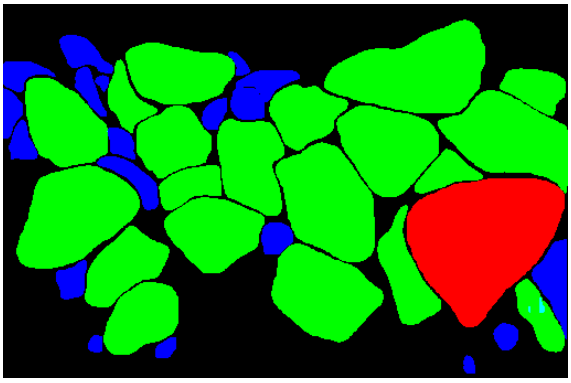
Tabla E.16: Resultados de precisión calculados de la imagen 75. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
75	80.435	100.000	100.000	88.750

(a) Imagen



(b) GT



(c) Predicción

Figura E.9: Imagen 77 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

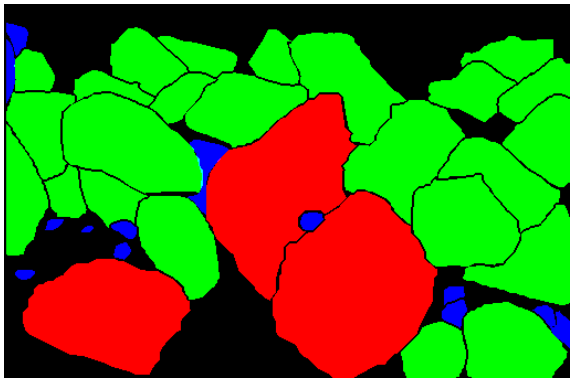
Tabla E.17: Resultados COCO de la imagen 77.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	63.697	87.805	74.929	43.372	80.279	93.333
Segm	62.811	87.805	73.651	40.579	80.373	100.000

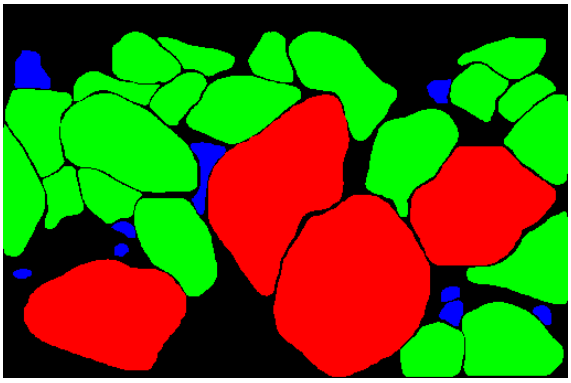
Tabla E.18: Resultados de precisión calculados de la imagen 77. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión "small"	Precisión "medium"	Precisión "large"	Precisión Total
77	88.235	100.000	100.000	95.122

(a) Imagen



(b) GT



(c) Predicción

Figura E.10: Imagen 80 dentro de la base de datos de prueba, junto a su GT y la predicción obtenida al procesarla con el mejor método.

Tabla E.19: Resultados COCO de la imagen 80.

Resultado	AP	AP50	AP75	APs	APm	APl
Bbox	55.187	80.198	59.660	30.561	65.722	96.634
Segm	55.903	80.198	63.956	29.267	66.864	95.545

Tabla E.20: Resultados de precisión calculados de la imagen 80. Resultados calculados con la precisión mencionada al final de la sección 4.2.2.

Imagen	Precisión “small”	Precisión “medium”	Precisión “large”	Precisión Total
80	85.714	95.833	75.000	91.429