Universidad de Valparaíso
CHILE

Facultad de Ciencias
Instituto de Estadística
Ingeniería en Estadística

Deep Learning aplicado a la Segmentación Semántica de Imágenes aéreas

Luis Altamirano P.
10 de diciembre, 2021

Profesor Guía
Rodrigo Salas F. Dr.
Escuela de Ingeniería Biomédica, Universidad de Valparaíso

Profesor Co-Guía
Daira Velandia M. Dra.
Instituto de Estadística, Universidad de Valparaíso

Proyecto de titulación para optar al:

grado académico de: *Licenciado en Estadística*
título profesional de: *Ingeniero en Estadística*
minor en: *Minería de datos*

Resumen

En el presente trabajo se evalúa dos arquitecturas de segmentación que se implementan en imágenes aéreas obtenidas por drones y satélites, se utilizan características preentrenadas de una red convolucional para la detección de información como patrones, figuras, siluetas, etc. Estas arquitecturas se han popularizado entre desarrolladores por sus satisfactorios resultados, pero no se ha profundizado en la utilización de sus métodos de extracción y aplicación en imágenes aéreas.

La segmentación semántica es un problema de la visión por computadora que se desarrolla en la comprensión del contexto de una imagen, la dificultad de este método se encuentra en el desarrollo de máscaras que son las encargadas de que objetos debe de entender un modelo, generar un conjunto de imágenes para entrenar modelos es una tarea complicada, ya que se debe de etiquetar cada objeto presente en la imagen, esto provoca conjuntos pequeños para entrenar. La utilización de redes convolucionales preentrenadas permite mejorar la precisión de los modelos a utilizar, transferir esta información, permite desarrollar modelos adecuados a la implementación de conjunto de imágenes aéreas.

Este trabajo se implementan dos arquitecturas para la segmentación de imágenes aéreas que son la U-Net y DeepLab. Se definen ambas arquitecturas para conocer sus metodologías de segmentación y se aplican al conjunto de imágenes aéreas, estos se deben preprocesar para mejorar la precisión y evitar un sobreajuste, por ello se normalizan para evitar valores grandes, un mayor cálculo en funciones y reducir la potencia requerida para necesitar un menor tiempo en el entrenamiento.

El desempeño de las arquitecturas es satisfactorio en ambos conjuntos de imágenes, en drones se logra clasificar cada objeto presente con una media por clase de al menos 50 %, esto quiere decir que más de la mitad de píxeles en cada clase de las imágenes logra ser clasificado correctamente, la media es influenciada por valores pequeños y grandes, porque pueden existir clases cuya clasificación sea precisa mientras que existen otras que inferiores (23 clases) y mientras que las imágenes captadas por satélites el desempeño es bajo, porque se aplica un método de clasificación multiclase, como el conjunto posee 8 clases al comparar las máscaras se visualizan 1, 2 o hasta 3 clases que segmentar, esto provoca que los modelos clasifiquen considerando el total de etiquetas del conjunto, pero el resultado es aceptable, ya que al observar la imagen original posee tales clases que las máscaras las definen con clase que predomina en la imagen, la razon de esto es por el conjunto de datos que esta preparado para detectar las clases de interés.

Palabras clave: Red neuronal convolucional, *Deep Learning*, U-Net, DeepLab, Segmentación semántica, Imágenes aéreas, Codificación y Decodificación.

Algunas Palabras

La vida es complicada, difícil de entender, hace cinco años no sabía que hacer o estudiar, cinco años después he estado esforzándome por conseguir mi título como Ingeniero en Estadística, han sido cinco años fabulosos de conociendo personas y conociéndome a mi mismo, pero siempre hay altos y bajos, lamentablemente las cosas suceden por algo y claro que es así, porque aquí me encuentro finalizando una etapa que no creí que ocurriera.

Agradezco a mi familia por soportarme y quererme, sé que este logro lo compartiremos, porque a pesar de que es mío también es de ellos.

Agradezco a los buenos compañeros y amigos que conocí, me ayudaron a mantenerme y a ser más feliz de lo que era antes.

Agradezco a cada profesor que me ayudó en esta formación, me hicieron entender que todo esfuerzo vale la pena sin importar el resultado mientras sepa aprender de mis errores.

Agradezco a los funcionarios por la pandemia no los volví a ver, pero aún los recuerdo por su cálida simpatía entregada cuando llegaba a la facultad.

Who cares what these cretins believe? They don't decide who you are, you decide, you are who you decide to be. - The iron giant

Índice general

Capítulo 1

Introducción

Este trabajo trata la aplicación de dos arquitecturas de segmentación y clasificación que extraen características de el uso del *Deep Learning* previamente entrenadas. Se aplican en dos arquitecturas será en dos conjuntos de imágenes aéreas multiclases captadas por drones y satélites. Se evalúa el desempeño con métricas de evaluación que miden los resultados de extracción de características importantes que entrega la imagen y ponen a prueba la generalización a un conjunto de imágenes aéreas desconocidas multiclases. Dependiendo de los resultados obtenidos de estas arquitecturas su implementación serian prometedoras para la automatización de drones o para la clasificación de zonas agricultoras, carreteras, etc.

1.1. Motivación

Recrear las habilidades humanas en máquinas ha sido un desafío para la ciencia, pero se ha logrado no solo recrear sino mejorar estas habilidades, obteniendo avances que abren nuevas posibilidades. Actualmente se ha requerido mejorar estas máquinas no solamente en sus capacidades motoras pues también comprensivas; la inteligencia artificial ha alcanzado tal nivel que se busca refinar algoritmos autónomos, es decir desarrollar máquinas que posean comprensión visual y tomen decisiones propias en base a lo aprendido como lo hacen drones de entrega y vehículos de transporte. La ciencia que se especializa en esta área se le conoce como visión por computadora o visión artificial. Por medio de la recopilación de imágenes, procesamiento y análisis es posible desarrollar un algoritmo capaz de aprender del contexto que entrega una imagen (Contaval, 2016), no es solo clasificar si la imagen es un perro o un gato pues estos algoritmos deben diferenciar e identificar la cantidad de objetos que se le presenten.

El ser humano a lo largo de su historia ha tratado de desarrollar maquinarias que imitan las diferentes características de seres vivos como volar, fuerza o visión, por esta última podemos interpretar nuestro entorno y tomar decisiones en base a lo que podemos ver, el animal que se distingue con esta característica es el águila, su vista es capaz de superar hasta 4 veces a la de un humano y con una gran nitidez para poder detectar a su presa en una gran altitud. La naturaleza nos entrega soluciones a problemas de nuestra especie como detectar anomalías, personas perdidas o el transporte de objetos (Celdrán, 2012), en el siglo 21 poseemos las herramientas necesarias para imitar las características de un águila, ya que la evolución de diferentes herramientas como las cámaras y aviones no tripulados o también conocidos como *UAV*, hacen posible el desarrollo de aparatos tecnológicos para mejorar la toma de decisiones tanto en el uso militar o como el empresarial.

1.2. Definición del problema y objetivos

Con el avance de la tecnología, el amplio uso de dispositivos con sistema de posicionamiento global o *global positioning system* (GPS) en vehículos y teléfonos móviles, la popularidad de las aplicaciones móviles, el *crowdsourcing* y los sistemas de información geográfica, así como dispositivos de almacenamiento de datos más económicos, se están recopilando gran cantidad de datos georreferenciados (imagen localizada en un sistema coordenadas) en más disciplinas como los negocios, la ciencia e ingeniería (Jiang y Shekhar, 2017). Una herramienta útil de obtención de imágenes son los drones, estos han logrado aumentar significativamente sus funciones, su alta cobertura de vuelo, pueden mapear kilómetros cuadrados de imágenes de gran resolución, además estos drones son capaces de cargar hasta 20 kilogramos.

Las imágenes obtenidas por drones se utilizan en diversas áreas como la detección de anomalías en una zona, la supervisión de la ganadería o agricultura y para el control de calidad de constructoras, algunas de estas actividades se puede llevar a cabo automatizando un dron y algunas empresas de encomienda han avanzado el aspecto y así lograr que un dron realice un envío sin necesidad de tener un supervisor, para esto es necesario el desarrollo de un algoritmo capaz de entender las diferentes escenas a las que se pueda enfrentar un dron en su vuelo y aumentar la seguridad y la eficacia para lograr aterrizajes en lugares apropiados. La obtención de información para desarrollar y entrenar estos algoritmos hoy en día puede obtenerse en forma masiva utilizando drones para mapear ciudades y así recopilar información necesaria para automatizar un dron (Chicchón, 2018).

A causa del incremento de las aplicaciones utilizadas en robots móviles, automóviles autónomos, realidad virtual, por nombrar algunos, se ha generado un mayor interés en el área de reconocimiento de escena, lo que impulsa la necesidad de comprender imágenes complejas (Ordaz y Flores, 2019). La segmentación semántica es la herramienta más apta en lograr reconocer escenas complejas por sus métodos basados en *deep learning*, este método ha logrado la detección o clasificación de múltiples objetos que presenta una imagen, logrando resultados satisfactorios para la conducción autónoma de vehículos, ya que identifica el camino que es seguro transitar, también en la inspección industrial para detectar defectos en materiales, identificar montañas o ríos, para mejorar la comprensión y en imágenes médicas para analizar y detectar anomalías y por último la visión robótica que ayuda a explorar e identificar objetos.

1.3. Segmentación semántica

La expresión segmentación semántica proviene de dividir o separar (segmentar) un conjunto de cosas de diversos aspectos (semántico) y por ello se entiende como un método que clasifica objetos o clases que contenga una imagen. La cantidad de objetos deben de estar predefinidos en el conjunto de datos y se le conoce como mascara o *ground-truth* en inglés, pues cada imagen original debe contar con una mascara que identifique cada objeto presente con un color o cuadro delimitador representativo que corresponde como su etiqueta, por ello sus delimitaciones deben de estar correctamente definidas para un entrenamiento satisfactorio y obtener un algoritmo que comprenda el contexto de una imagen. Se han desarrollado diversos métodos de segmentación como: reconocimiento de imágenes, detección de objetos, segmentación semántica y segmentación de instancias (Palomino y Concha, 2009). En la Figura 1.1 se observan las diferencias entre reconocimiento de imágenes (*Classification*) donde solamente se asigna una etiqueta del objeto presente a toda la

imagen que este seria “Globo”, detección de objetos (*Object Detection*) que se encaja cada cuadro delimitador al objeto presente en la imagen que seria a cada “Globo”, segmentación semántica (*Semantic Segmentation*) capta los múltiples objetos de las diferentes clases como un solo valor que destaca a cada “Globo” del contorno y la segmentación de instancias (*Instance Segmentation*) la cual trata los múltiples objetos de las diferentes clases como individuales que destaca logrando diferenciar a cada “Globo” como un único individuo.

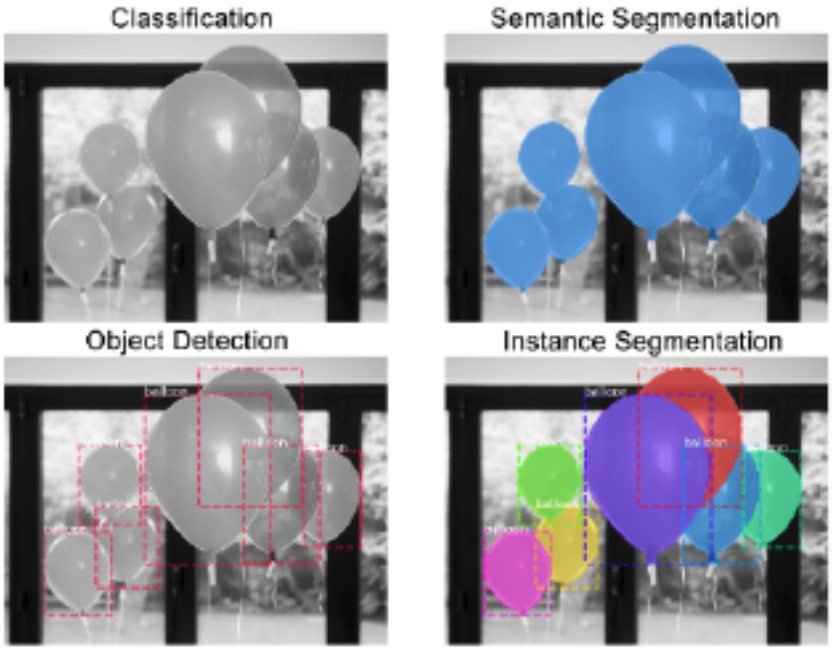


Figura 1.1: Diferencias entre métodos de segmentación de imágenes. Tomado de (Dwivedi, 2019).

1.3.1. Definición de Segmentación semántica

Los autores (Barrera, Guindel, García, y Martín, 2018), definen la segmentación semántica como el método de etiquetar cada píxel de una imagen a una clase ya predefinida, puede ser una persona, un perro, césped, etcétera, logrando una comprensión completa de la imagen. Por otro lado, (Zhang y cols., 2018), mencionan que la segmentación semántica es una predicción por píxel de los objetos de una imagen, entregando una definición completa del contexto de la imagen ya sea la ubicación, la forma del objeto o la categoría. A continuación, se puede visualizar en la Figura 1.2 los primeros resultados satisfactorios que tuvo la segmentación semántica (Shotton, Johnson, y Cipolla, 2008) las cuales utilizan redes multicapa como clasificador, aun que se observa un resultado ruidoso aun así logran definir las clases o objetos en este.

Figura 1.2: Primeros resultados satisfactorios del método de segmentación semántica de los autores. Tomado de (Shotton y cols., 2008).

1.4. Objetivo general

El objetivo de este trabajo es evaluar el desempeño de dos metodologías de *Deep Learning* para la segmentación semántica de imágenes aéreas.

1.4.1. Objetivos específicos

Como objetivos específicos de este trabajo se plantean los siguientes:

- Desarrollar y aplicar las metodologías de *Deep Learning* U-Net y DeepLab para la segmentación semántica de imágenes aéreas.
- Evaluar el desempeño de las metodologías de *Deep Learning* para la segmentación semántica de imágenes aéreas obtenidas mediante drones.
- Evaluar el desempeño de las metodologías de *Deep Learning* para la segmentación semántica de imágenes aéreas obtenidas mediante satélites.

1.5. Hipótesis

La hipótesis planteada es para este trabajo es: Mediante técnicas de *Deep Learning* se puede implementar un modelo apropiado para segmentar imágenes aéreas.

Con este trabajo se busca motivar la aplicación de arquitecturas de segmentación en la detección o clasificación de diversos objetos, ya que estos modelos permiten mejorar la observación en una imagen y mejorar la gestión de calidad en un gran margen de trabajo, es posible minimizar sus pérdidas y optimizar la búsqueda de diversos objetos por arquitecturas de segmentación.

Capítulo 2

Marco Teórico

Se definen las arquitecturas de segmentación que se utilizan en este trabajo, primero se realiza una breve explicación de las redes neuronales artificiales para comprender su estructura básica y posterior se detalla el proceso de una red neuronal convolucional para mejor entendimiento de las arquitecturas de segmentación de *deep learning*.

2.1. Introducción a las Redes Neuronales Artificiales (RNAs)

Las RNAs nacen de la inspiración biológica para convertir una máquina en inteligente su desarrollo surge de lo psicológico, los griegos Platón y Aristóteles entregan explicaciones teóricas del funcionamiento del cerebro, desde entonces diferentes científicos se han basado en estos estudios para entender como aprende el sistema neuronal desarrollando interesantes teorías del funcionamiento y de cómo aplicarlas en computadoras, actualmente se conoce como inteligencia artificial. En el año 1960 se desarrolla la primera RNA conocida como perceptrón, era capaz de reconocer diferentes patrones y detectar similares en un conjunto de datos diferente (Durán, 2017). Se han desarrollado diferentes tipos de RNA para resolver complejos problemas como por ejemplo el perceptrón multicapa, recurrente, convolucional, entre otros.

Para entender el comportamiento de una red neuronal artificial es importante conocer el funcionamiento biológico de una red neuronal. La Figura 2.1 se observa la estructura típica de las neuronas que se conforma por un axón, dendritas y el cuerpo (soma), las dendritas cumplen el trabajo de transportar los impulsos eléctricos al cuerpo de la neurona (entrada), el cuerpo de la neurona reconocerá y procesará las señales entrantes y finalmente esta información será enviada por un único axón que comunicará al cuerpo de la neurona con las demás, esto se conoce como sinapsis. El cerebro humano contiene un gran número de neurona (alrededor de 10^{11}) que se conectan entre si formando una red neuronal (Durán, 2017).

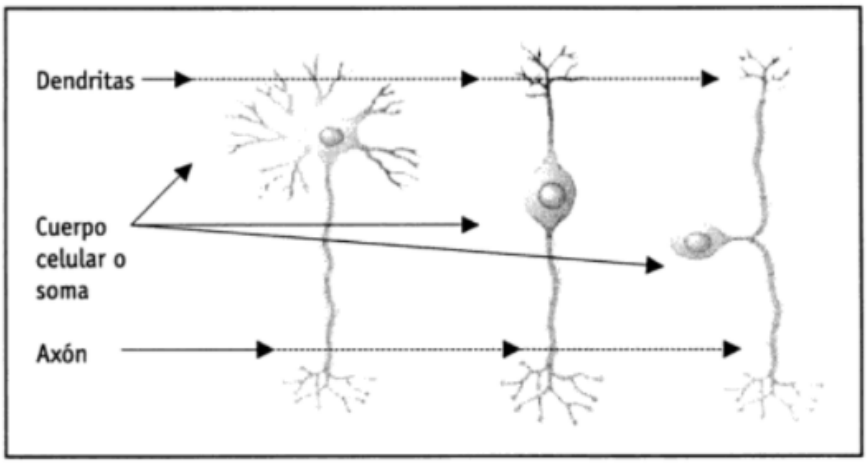


Figura 2.1: Estructura básica de una red neuronal biológica. Tomado de (López y Fernandez, 2008).

En 1949, el neuropsicólogo (Hebb, 1949) mostró un ejemplo simple, basado en la fisiología, para imitar la plasticidad sináptica entre neuronas y calcular el peso de la conexión neuronal dentro de la red. Explicando de manera biológica "Las señales que llegan a la sinapsis son las entradas a la neurona: estas son ponderadas (atenuadas o simplificadas) a través de un parámetro, denominado peso, asociado a la sinapsis correspondiente. Estas señales de entrada pueden excitar a la neurona o inhibirla (peso negativo). El efecto es la suma de las entradas ponderadas. Si la suma es igual o mayor que el umbral de la neurona, entonces se activa" (Systems, 2020). De esta manera es como se aplica una red neuronal artificial, en la Figura 2.2 se observa la arquitectura simple que describe (Hebb, 1949), donde x_m son las señales que son ponderadas por los pesos w_m y son procesadas por la unión sumadora y finalmente son activadas por una función que dependerá del valor de la neurona para estar activada, puede estar entre $[0, 1]$ o $[-1, 1]$, la neurona puede estar inactiva si el valor es $(-1, 0)$ o activa cuando toma el valor 1, pues de esta manera se transmitirá la información a una nueva neurona.

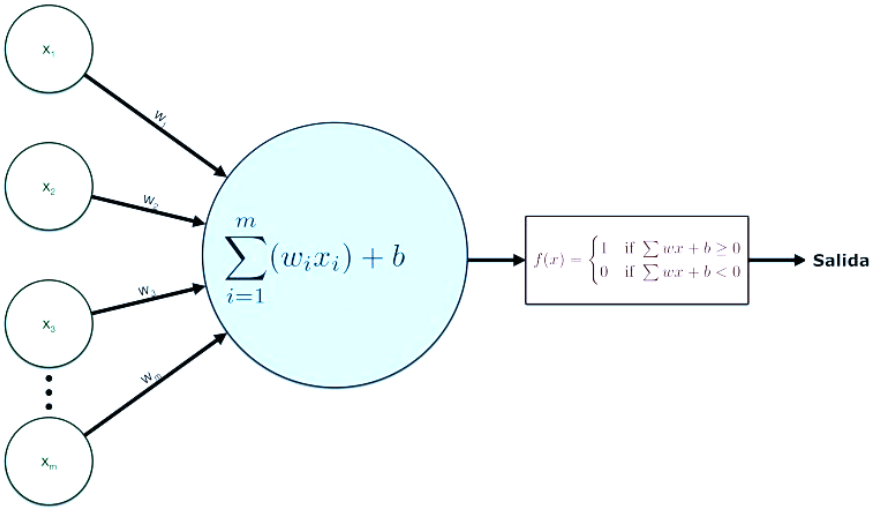


Figura 2.2: Arquitectura básica de una red neuronal artificial compuesta por neuronas, pesos, unión sumadora y función de activación. Tomado de (Systems, 2020).

La interpretación de cómo trabaja una neurona biológica y artificial, tomó décadas en entender y desarrollar lo que actualmente se utiliza sin darnos cuenta en computadoras, celulares, la recomendación de un vídeo, la publicidad, etc. Su implementación va tanto en la ingeniera como en el marketing.

2.2. Red Neuronal Convolucional (RNC)

El desarrollo de la red neuronal convolucional (*Convolutional neural network*, CNN) en el 2012 logro batir el récord del 2011 reduciendo el error del conjunto de datos ImageNet con más de 15 millones de etiquetas de alta resolución. Imágenes que pertenecen a aproximadamente 22.000 clases. Las imagenes fueron recopilados de la web y etiquetados por personas utilizando la herramienta de *crowdfunding* de *Amazon Mechanical Turk*. La RNC logró reducir anteriores resultados de error de un 25.8 % a 16.4 % (Krizhevsky, Sutskever, y Hinton, 2012), resulto ser un método eficaz y mejoró no tan solo la velocidad de procesar imágenes, también el aprendizaje, se popularizó por su innovadora arquitectura captando la atención de diferentes desarrolladores, además, actualmente el desarrollo de algoritmos en el reconocimiento de imágenes está basado en la red neuronal convolucional mejorando o integrando nuevos atributos que mostraremos más adelante.

2.2.1. Definición de una RNC

Este tipo de red fue diseñada por (Krizhevsky y cols., 2012) imitando la corteza visual y consiguiendo manipular matrices bidimensionales, su desempeño con imágenes resultó ser satisfactorio y al agregar *deep learning* toma como nombre "red neuronal convolucional profunda", de esta manera procesará grandes cantidades de imágenes encontrando características importantes como patrones, formas, entre otros y así distinguir de los diferentes objetos que se encuentren para finalmente conseguir resultados satisfactorios en el aprendizaje y logre distinguir patrones similares en un conjunto de datos desconocido, en la Figura 2.3 se visualiza la arquitectura básica de una RNC.

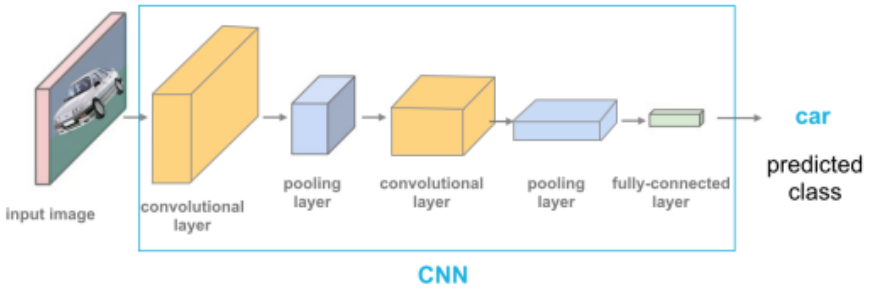


Figura 2.3: Ejemplo de una red neuronal convolucional, donde se alternan capas de convolución y de *pooling*. Al final de la red se encuentra la capa totalmente conectada que se ocupa de clasificación donde este caso es auto. Tomada de (Berrondo Urruzola, 2020).

2.2.2. Preproceso

La aplicación de la red puede ser en cualquier calidad de imagen, pero a mayores dimensiones requerirá una mayor potencia computacional requiriendo más tiempo. Actualmente las cámaras han mejorado su calidad en la capturá de imágenes logrando alcanzar la calidad de 4k (3840×2160 píxeles), procesar esta calidad de imagen en la red provoca una carga mayor en tiempo, pero la ventaja está en que las características importantes se destacan mejor que en las calidades inferiores de capturá por su cantidad de píxeles y se esperaría resultados satisfactorios mayores que en calidades inferiores, pero dicho antes este proceso provocaría un largo periodo de desarrollo, al trabajar con imágenes se debe considerar tanto la velocidad de proceso como la calidad de imagen. La cantidad de imágenes también será influyente, porque la red aprenderá de la diversidad de patrones que se encuentren en los objetos para poder reconocerlos en una aplicación real, esta cantidad se duplica por el conjunto de máscaras que son la representación de etiquetas en la imagen, los objetos que aprenderá la red poseerán estas etiquetas como colores representativos, es decir si poseemos en una imagen diversos objetos, pero queremos que la red identifique a personas, pavimento y el entorno, el investigador debe desarrollar a cada imagen del conjunto una máscara que se le asigne a estos un color como: rojo para persona, gris a pavimento y azul al entorno, así la red comprenderá que debe detectar en la imagen. Por ello el conjunto de la máscara es sumamente importante para desarrollar un modelo de segmentación.

Las imágenes a color contienen 3 canales que corresponden a rojo, verde y azul (*RGB*, abreviado en inglés) los píxeles tienen valores entre 0 y 255, normalizar las imágenes es importante antes de adentrarlas a una red (Bagnato, 2020), porque la escala es bastante grande y la red poseerá problemas de valores dominantes afectando a la clasificación, además al trabajar con imágenes los pesos calculados para clasificar los objetos serían altos provocando un mayor uso de memoria y más tiempo de proceso. Existe una técnica comúnmente utilizada para normalizar imágenes que es la estandarización esta se define como:

$$\text{Estandarización} = \frac{x - \mu}{\sigma} \,. \tag{2.1}$$

donde x corresponde al valor del píxel, μ a la media, σ la desviación típica y b a la cantidad de píxeles en la imagen, estos se definen como.

$$b = n * m \,, \quad \text{donde } n \text{ es el ancho y } m \text{ la altura de la imagen} \tag{2.2}$$

$$\mu = \frac{\sum_{i=1}^{n} x_i}{b} \,. \tag{2.3}$$

$$\sigma = \frac{\sum_{i=1}^{n} (x_i - \mu)}{b} \,. \tag{2.4}$$

Además, la estandarización no solamente ayudará a evitar los problemas antes mencionados también facilitará la generalización en el entrenamiento provocando mejores resultados al aplicar él modelo a imágenes desconocidas (Programador clic, 2018).

2.2.3. Entrada de la RNC

Luego de preprocesar las imágenes de dimensiones $n \times m$ estas podrán ingresar a la red como un vector de tamaño $n \times m \times 1$, la cantidad de neuronas de entrada dependerá de la cantidad de píxeles que contenga la imagen, ya que cada neurona se hará responsable de cada píxel por ejemplo, si la imagen posee dimensiones $300 \times 200 \times 1$ (en tamaño de vector se considera como $60{,}000 \times 1$, pero para mantener la idea de dimensiones y la sencillez se interpreta como $300 \times 200 \times 1$) entonces la red poseerá 60.000 neuronas como capa de entrada. Sin embargo, esto se aplica si la imagen está en escala de grises que es conveniente para un proceso menor, si la imagen es a color se tienen que contar los 3 canales de entrada que esta cuenta (rojo, verde y azul) por lo que esta cantidad se triplicaría dando una entrada de 180.000 neuronas, por esto trabajar con imágenes es complejo, pues al diseñar una arquitectura o modelo se debe ser capaz no solo analizar las imágenes sino también de ser eficaz con la memoria y el tiempo. Para distinguir de la información que contienen estas neuronas pasan por un filtro para obtener las características importantes de las imágenes y reducir la imagen al punto de entender sus características, esto se logra con las convoluciones (Bagnato, 2020).

2.2.4. Convoluciones

Una convolución es el proceso por el que se distingue la red neuronal convolucional (por ello su nombre), por la utilización de diferentes técnicas la red aplicará distintos filtros para distinguir características y luego activar las neuronas que contienen esta información, para transferirlas a las capas ocultas (se le llama oculta, porque desconocemos tanto los valores de entrada como de salida), para desarrollar esta solución se restringe el número de conexiones posibles entre las neuronas de la capa oculta y las clases de objetos que contenga la imagen (Durán, 2017).

Kernel

El kernel es el filtro que se aplica a la imagen de entrada para extraer las características importantes de una imagen, estos pueden ser bordes, enfoque, desenfoque, entre otros y de igual manera para encontrar diferentes patrones que pueda reconocer la red. En la Figura 2.6 se visualiza una imagen de entrada con un kernel 3×3, este tamaño es el comúnmente utilizado, pero puede poseer un tamaño $p \times p$ (el valor de p dependerá del diseñador y debe pertenecer a los enteros), además, no puede superar el tamaño de una imagen de entrada ($p \times p < n \times m$), sus valores son tomados al azar, pero se reconocen algunos kernels por sus resultados en la aplicación en una imagen, por ejemplo, en la Figura 2.4 se tiene el kernel identidad que entrega los mismos valores de la imagen, pero en la Figura 2.5 se observa un kernel que mejora la nitidez destacando los bordes que posee la imagen, de esta manera el modelo detecta patrones para distinguir los objetos.

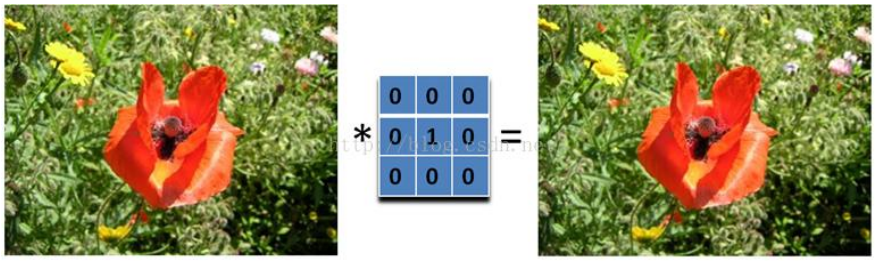


Figura 2.4: Kernel identidad tamaño 3×3 al aplicar a una imagen conservaremos los mismos valores que esta posea. Tomado de (Xiang, 2019).

Figura 2.5: Kernel de nitidez tamaño 3×3 al aplicar resaltara los bordes que contenga la imagen transformándola más nítida. Tomado de (Xiang, 2019).

Los valores del kernel pueden ser fijos dependiendo del resultado en la clasificación, se debe aplicar un método llamado *backpropagation*, la red en la capa de salida decidirá si una neurona pertenece a tal clase, si es clasificado con una clase errónea el método calculara este error por el valor del píxel y de los kernels utilizados, ajustará los pesos en los kernels utilizados al clasíficar erróneamente el píxel, ya que este pudo suavizarlo o desenfocarlo reduciendo importancia, en cada entrenamiento los valores del kernel serán fijos, obteniendo mejor precisión de clasificación (Calvo, 2018).

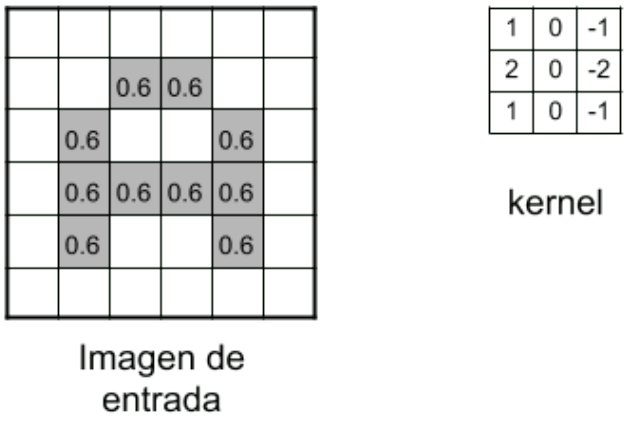


Figura 2.6: Ejemplo de una imagen en escala de grises normalizada y un kernel 3×3 con valores definidos, este kernel destaca. Tomado de (Bagnato, 2020).

El kernel se irá desplazando por toda la imagen de entrada a un píxel, operando contra un grupo de píxeles del mismo tamaño por el cálculo "producto escalar", generando una nueva matriz con valores pertenecientes a los reales pues estos serán una nueva capa de neuronas ocultas, el kernel también recibe como nombre "campo receptivo" dado que es la región de entrada que afecta a la región de un determinado píxel de salida (ver Figura 2.7) (Ku, 2020a). En la Figura 2.8 se observa un pequeño ejemplo de lo descrito anteriormente, en caso de ser una imagen a color se aplicarán 3 kernel de tamaño 3×3 que operarán contra los 3 canales para que los 3 diferentes mapas resultantes se sumen y tener una única salida.

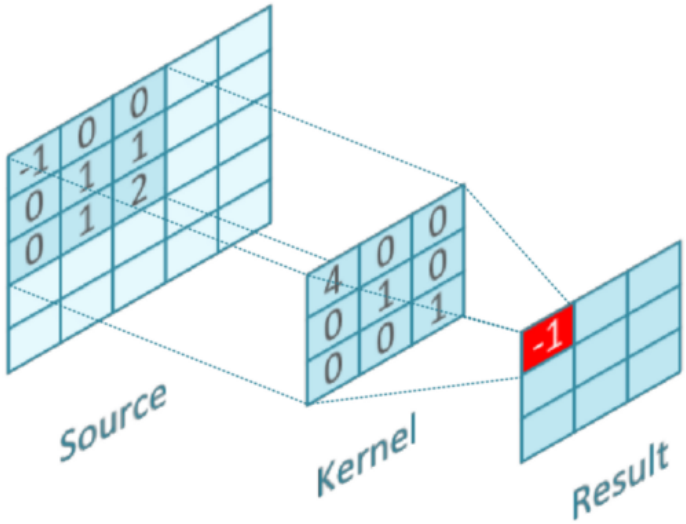


Figura 2.7: Ejemplo del campo receptivo 3×3 aplicado a la imagen 5×5 donde solo se destaca la primera operación del kernel para obtener una matriz de 3×3. Tomado de (Ku, 2020a).

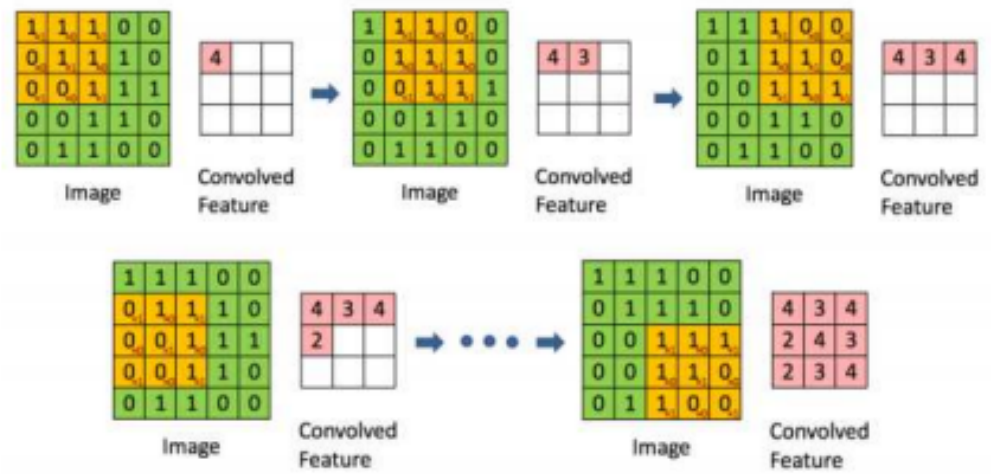
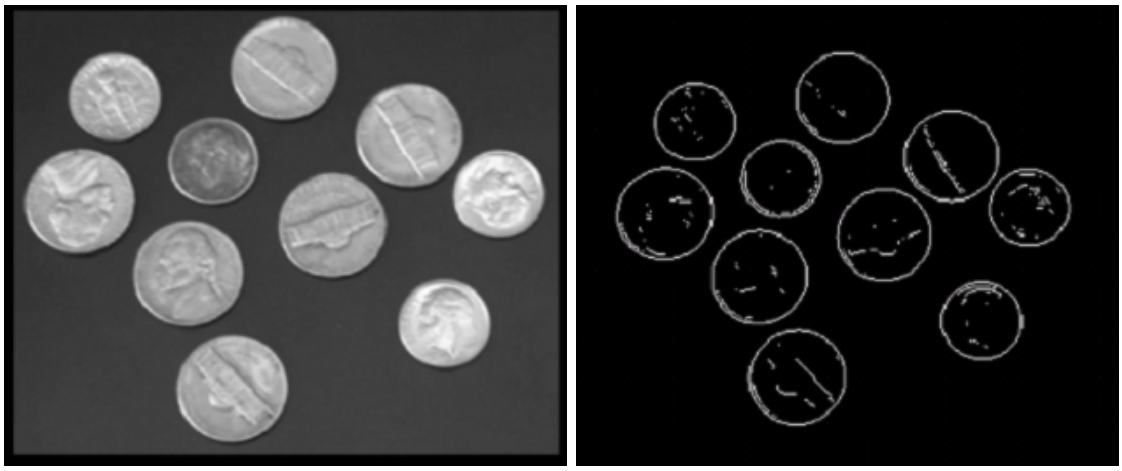


Figura 2.8: Ejemplo de desplazamiento del kernel 3×3 por una imagen de dimensiones 5×5 el resultado de este es una nueva imagen conocida como mapa de características de tamaño 3×3. Tomado de (Durán, 2017).

Este proceso se puede aplicar varias veces (dependiendo del diseñador), en la primera convolución este tendrá valor k (un entero arbitrario), al aplicar otra convolución el valor de k deberá de ser el doble, si aplicamos 3 convoluciones el valor inicial sería k, luego $2 * k$ y finalmente $4 * k$. Supongamos que aplicamos 32 kernels pues se obtendrán 32 nuevas matrices filtradas, estas nuevas matrices se les conoce como mapas de características (*feature mapping*, en inglés). Retomando el ejemplo en la capa de entrada se tiene una imagen con dimensiones $300 \times 200 \times 1$ al aplicar el kernel se obtiene un mapa de características con dimensiones de $298 \times 198 \times 1$ luego se aplicará otro hasta obtener un mapa de $298 \times 198 \times 32$, la aplicación de los kernels en general nos entregará una matriz de dimensiones $(n - p + 1) \times (m - p + 1) \times k$, al tener esta cantidad de mapas de características podemos calcular un aproximado de cuantas neuronas haremos uso para procesar esta información, sería un estimado de 1.8 millones de neuronas que se adentraran a una nueva capa oculta. Por medio de este mapa de características la red aprenderá de los distintos objetos que la imagen presente para en la salida segmentarlos, en la Figura 2.9 se presenta otro ejemplo de la aplicación de un kernel, pero este destaca los bordes de las monedas a tonalidades claras y al resto en oscuras destacando nuevos patrones de características importantes que la red aprenderá.



(a) Imagen original. (b) Imagen con kernel aplicado.

Figura 2.9: Ejemplo del resultado de aplicación matriz kernel a una imagen de monedas. Tomado de (Ai, 2019).

En una convolución se pueden destacar diferentes características por ello para realizar más convoluciones se debe preparar esta nueva información para adentrar a nuevas neuronas, se aplican dos nuevos pasos conocidos como "función de activación" que es parte de la convolución y una técnica de "submuestreo".

Función de activación

Para transmitir la información que se obtuvo del mapa de características se debe de realizar una activación en las neuronas biológicas, lo que se conoce como impulso energético, ya que transporta la información a otra neurona, por ello este proceso se le conoce como "función de activación", pues esta se encargará de devolver un valor de salida a un valor de entrada transformando los valores de las neuronas, esta acción al mapa de características toma como nombre "mapa de activación", hay gran variedad de funciones de activación para lograr diferentes objetivos como: la Sigmoide que es muy usada para problemas de clasificación binaria, Tangente Hiperbólica es parecida a la Sigmoide, pero funciona bien para datos cuya media está cercana a cero y la ReLU que simplemente cambia valores. Esta función la convierte en la indicada para la clasificación de imágenes (Rubiales, 2020), su función esta denotada por:

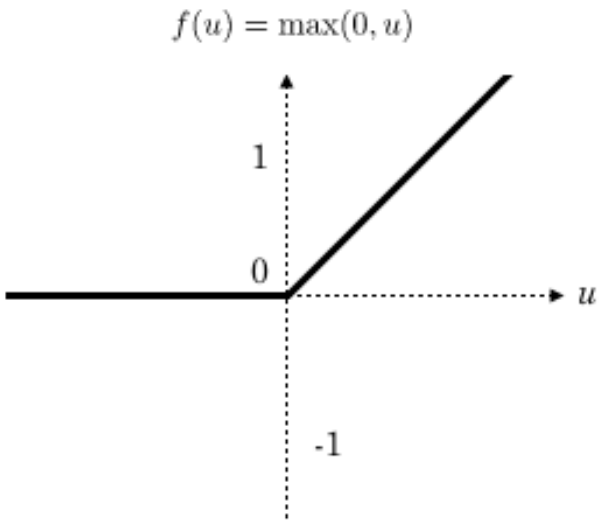


Figura 2.10: Descripción gráfica de la función de activación ReLU. Tomado de (Díaz Bou, 2021).

Esta sencilla función es la más utilizada en imágenes por no poseer un cálculo exponencial, ya que al contar con grandes cantidades de imágenes (como en el *deep learning*) el tiempo del proceso será mayor, pero al solo contar con una sustitución tendrá un cálculo menor por su simpleza, esta función modificará a cero los valores que sean menores a cero mientras los valores mayores o iguales se mantendrán, en este tipo de trabajos se debe de considerar el tiempo de procesamiento pues puede llegar a tomar horas, días o hasta semanas entrenar un modelo de segmentación. Sin embargo, al modificar valores a cero se anulan neuronas y no aportarán al modelo, se requeriría un importante poder computacional en futuras convoluciones si se mantienen estas valores nulos y seria un problema (Rubiales, 2020). Es necesario transmitir solamente la información importante dejando atrás la que no es relevante y así reducir el poder computacional necesario.

2.2.5. Submuestreo

Luego de realizar la convolución filtrando la imagen en 32 nuevos mapas de características y aplicar la función de activación para transferir la información a una nueva convolución se debe reducir las dimensiones, esto mejorará el mapa para la observación o análisis de características importantes, debido a que se tomara un grupo de píxeles que se puede aplicar una de estos métodos, *Mean-pooling* que promediará este grupo de píxeles o *Max-pooling* que extraerá el valor mayor que contenga este grupo, es decir que del grupo de píxeles del primer desplazamiento se elegirá el valor representativo. En la Figura 2.11 se observa la comparación entre ambos métodos, el contexto de la imagen influirá en el desempeño de los métodos, porque al trabajar con *Mean-pooling* suavizará el mapa y no se identificarán características nítidas, en cambio con *Max-pooling* seleccionará los píxeles de mayor valor (los más brillantes) y su ventaja esta que en zonas oscuras logra identificar los píxeles relevantes (Basavarajaiah, 2019), por ello este método es el más utilizado en las RNC.

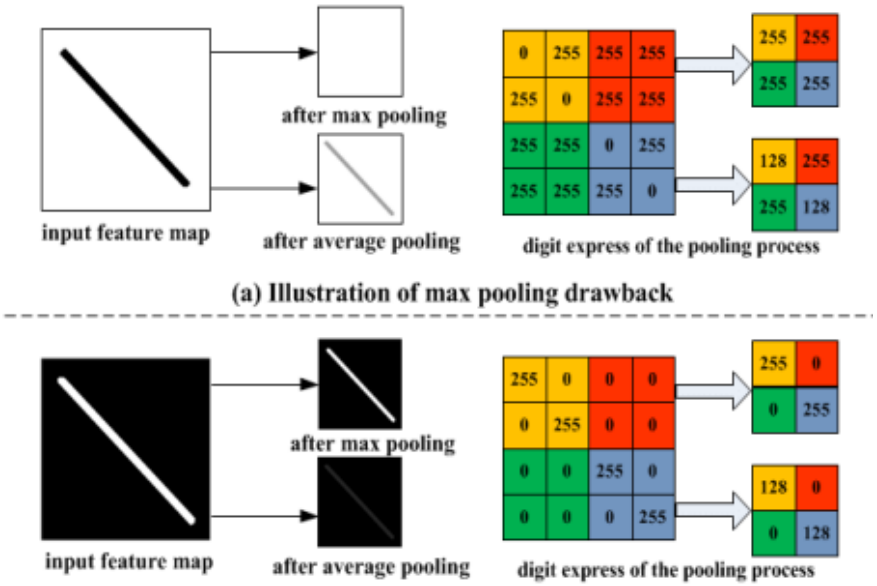


Figura 2.11: Comparación entre *Mean-pooling* y *Max-pooling* sobre una imagen en escala de grises (0 a 255). Tomado de (Durán, 2017).

El tamaño del cuadro de submuestreo $q \times q$ varía entre dos (para imágenes pequeñas) y cinco (para imágenes grandes), aplicando el submuestreo cada mapa de características será reducido a $(\frac{n-p+1}{q}) \times (\frac{m-p+1}{q}) \times k$, En las Figuras 2.11 y 2.12 se presentan un ejemplo donde se realiza *Max-pooling* de 2×2 a un mapa de características, a diferencia del kernel este no se desplazará de a un píxel pues lo hará con las proporciones del tamaño $q \times q$, es por este avance que reduce el mapa de características a la mitad, conservando la información relevante y de esta manera se obtendrá una nueva imagen de entrada (con la cantidad de mapas aplicados anteriormente), los canales entrantes originales aún se conservaran constantes y en teoría aún se debe conservar las características importantes de la imagen para detectar las distensiones en los objetos (Barrios, 2019).

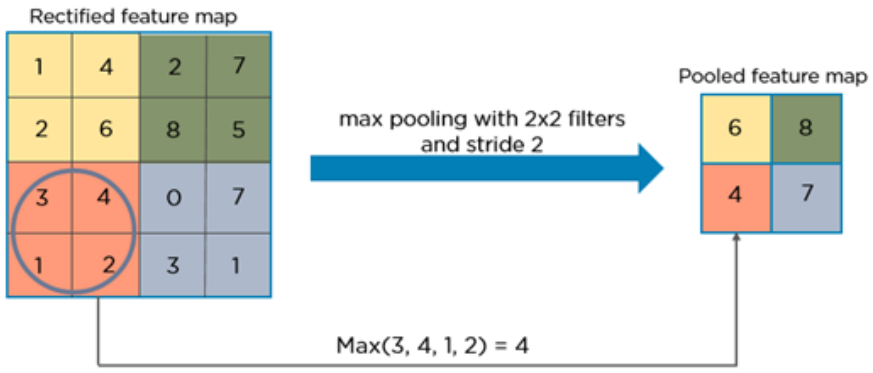


Figura 2.12: Ejemplo aplicación *Max-pooling* 2×2 en un mapa de características $2 \times 2 \times 4$. Tomado de (Lamba, 2019).

En el ejemplo planteado se tenia un mapa de características de $298 \times 198 \times 32$ al reducirlo

por *Max-pooling* de 2×2 se obtiene un nuevo mapa de entrada con dimensiones $149 \times 99 \times 32$, antes se estimo la cantidad de neuronas de entrada que es un aproximado de 1.8 millones píxeles y ahora se posee alrededor de 470 mil píxeles, es decir que los mapas se redujeron un 74 % y a la vez se reduce la potencia computacional y tiempo de proceso.

La cantidad de submuestreos que se puede aplicar a la imagen esta entre 2 y 5 (esta cifra se asume por la capacidad que permiten las librerías de modelos de segmentación en *Python*) para terminar el ejemplo planteado se realiza otra convolución, se tienen mapas de características con dimensiones $149 \times 99 \times 32$ aplicamos los 64 kernels de dimensiones $3 \times 3 \times 32$ obteniendo un mapa de activación de dimensiones $147 \times 97 \times 64$, se aplica la función de activación ReLU para finalmente reducir las dimensiones con *Max-pooling* como resultado se obtiene un nuevo mapa de tamaño $74 \times 49 \times 64$, cuando se tiene un número impar de dimensiones se agregan valores ceros como fila o columna para extraer la información de los bordes, teniendo alrededor de 232 mil neuronas en esta última convolución. Finalmente, la red aprendió de las diferentes características de la imagen para así realizar la segmentación.

2.2.6. Clasificación

Al final de la red RNC se encuentra la capa totalmente conectada (perceptrón multicapa), entonces cada neurona de la última convolución estará conectada con cada neurona de la capa de salida, el tamaño de las neuronas que se tendrá en la salida dependerá de la cantidad de clases que contiene el conjunto de imágenes, pues si este posee clases como perro, persona y entorno entonces se tiene 3 neuronas de salida para transmitir esta información, esta capa ya no usará la función de activación ReLU, se utilizará la función Softmax, esta es la más utilizada en la clasificación multiclase, ya que es una generalización de una regresión logística y así se puede transformar esta información en probabilidad (Bagnato, 2020), se define como:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{para } j = 1, ...,K. , \tag{2.5}$$

Donde K corresponde a la cantidad total de neuronas que se obtienen en la última convolución y z a los valores de cada neurona. La estructura que tienen las neuronas en la capa de salida es la de un vector, por ejemplo tomando en cuenta 3 clases los vectores como: perro $[1, 0, 0]$, persona $[0, 1, 0]$ y entorno $[0, 0, 1]$, la capa oculta que conecta con la capa de salida de cada neurona es un vector y al aplicar Softmax las neuronas tienen una probabilidad que se distribuirán por las 3 clases para luego ser clasificados (es decir que la probabilidad que sea 1 es igual a $a + b + c$, donde a, b y c son clases o objetos del conjunto de imágenes), por ejemplo en la figura 2.13 observamos que un modelo se entrenó para clasificar 4 clases (*dog*, *cat*, *horse* y *cheetah*) al adentrarse en la última capa oculta este toma la forma de un vector que distingue cada clase y al aplicar la función Softmax estas toman una probabilidad de pertenecer a dicha clase, en este ejemplo la clase *dog* posee un 77,5 % de probabilidad de pertenecer a esta clase.

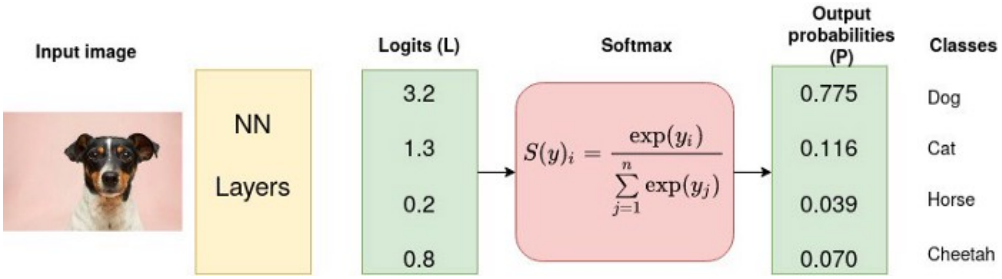


Figura 2.13: Ejemplo de simple de la aplicación Softmax a un modelo de clasificación entrenado con 4 clases. Tomado de (Koech, 2021).

Mencionado anteriormente en caso de equivocarse la red en cada entrenamiento se ira ajustando por *backpropagation*, este mejorará el valor de las probabilidades asignadas hasta hacerlos óptimos, calculará que tan equivocado estuvo en su clasificación, en el caso descrito, si esa neurona se clasificará como clase "*cheetah*", es decir el error de clasificación sería el mayor entre las otras clases, por ende, el peso en ajustar la probabilidad será mayor. En la siguiente Figura 2.14 se pueden observar los resultados que presentaron los desarrolladores de la red neuronal convolucional, pues estos realizaron un modelo para reconocer imágenes, es decir una clasificación de etiqueta única.

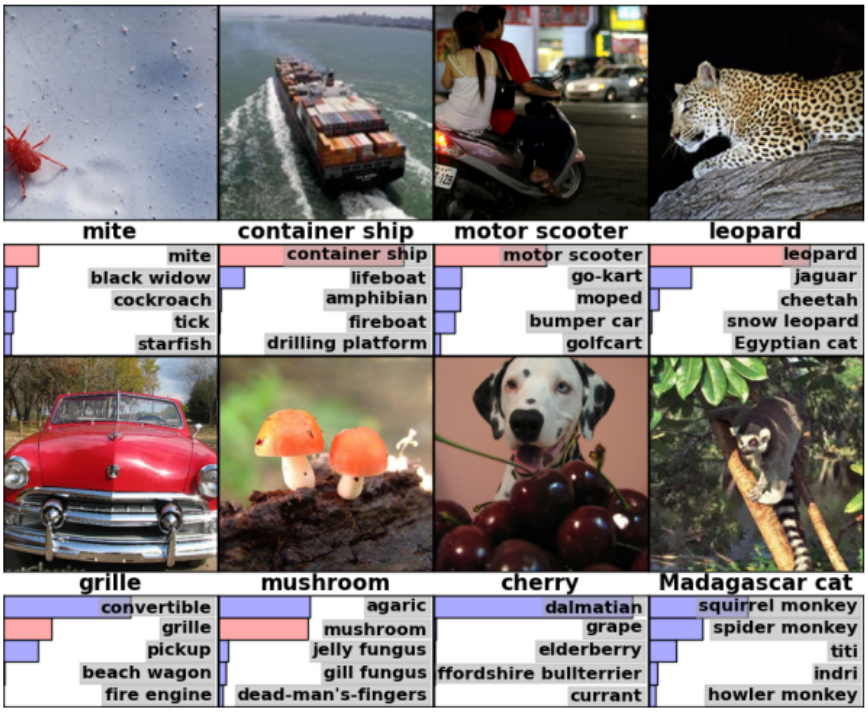


Figura 2.14: Resultados de segmentación por reconocimiento de imagen, ocho imágenes de prueba y las cinco clases consideradas más probables por el modelo RNC. La clase correcta está escrita debajo de cada imagen y la probabilidad asignada a la clase correcta también se muestra con una barra roja (si está entre las 5 primeras). Tomado de (Krizhevsky y cols., 2012).

Existen diversos modelos o arquitecturas basadas en redes RNC para segmentar imágenes

que incluso han agregado diferentes mejoras y esto es algo fascinante en el *deep learning* para los diseñadores que moldean y crean diferentes modelos para determinados conjunto de datos que luego otros los apliquen a nuevos conjuntos y conocer si se adaptan con clasificaciones satisfactorias, en este proyecto se aplicaran dos arquitecturas conocidas en la ciencia de la Visión Artificial definidas a continuación.

2.3. Arquitectura U-Net

La U-Net es una arquitectura de segmentación de imágenes que se utilizó en datos de biomédica para mejorar la imagen y obtener mayor comprensión de esta, logro ganar concursos como "Desafío de ISBI para la segmentación de estructuras neuronales en pilas de microscopio electrónico", el "Gran Desafío para la Detección Automatizada por Computadora de Caries en Radiografía de Mordida en ISBI 2015" y el "Desafío de Seguimiento Celular en ISBI 2015".

2.3.1. Definición de la arquitectura U-Net

Según (ArcGIS Developers, s.f.) su arquitectura se puede considerar en términos generales como una red de codificadores seguida de una red de decodificadores. U-Net es un modelo innovador para la segmentación y no solo en imágenes médicas, también para la aplicación en imágenes satelitales, agricultura, construcción, etc. Su arquitectura se centra en dos fases: Codificador y decodificador obteniendo una red simétrica que forma una U (ver Figura 2.15), el codificador realiza el trabajo de una red neuronal convolucional. La fase decodificación es lo novedoso e innovador de esta arquitectura, pues se realiza lo inverso, en lugar de contraer o reducir la imagen esta la expandirá proyectando la imagen segmentada y se aplicará la misma cantidad que en las convoluciones (Siddique, Sidike, Elkin, y Devabhaktuni, 2020).

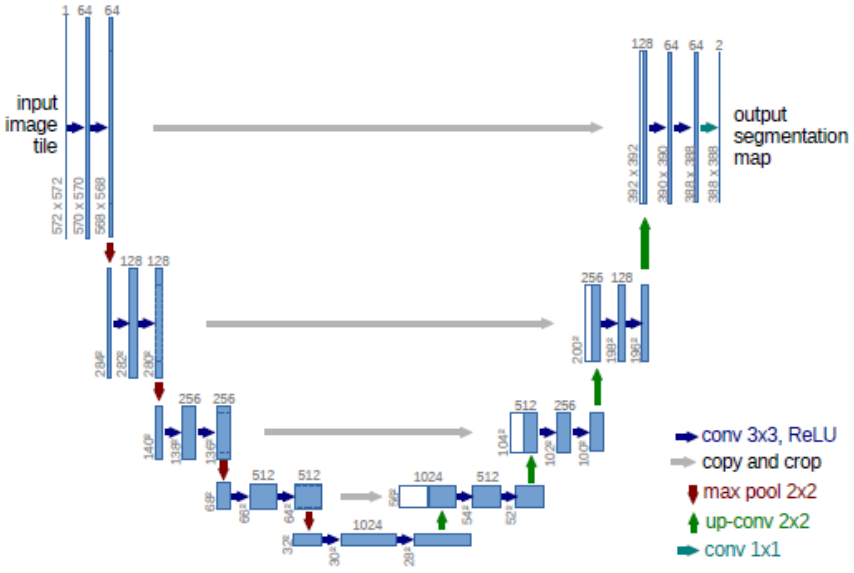


Figura 2.15: Arquitectura U-Net compuesta por codificadores (mitad izquierda) y decodificadores (mitad derecha). Tomado de (Ronneberger y cols., 2015).

2.3.2. Codificación

La fase de codificación se nombra de esta manera, porque almacenaremos y obtendremos información de las cantidades de imágenes que se procesará, pues haremos solamente uso de una red neuronal convolucional (Ronneberger y cols., 2015) utilizando lo descrito en la sesión RNC, pero se aplicarán dos pasos adicionales que ayudarán a obtener una mayor precisión de clasificación, el primero sería un preproceso (además del que se hace antes de ingresar las imágenes) que se realiza antes de utilizar la función de activación ReLU, este preproceso se conoce como "Normalización por lotes", (Durán, 2019) explica que esta técnica es utilizada para normalizar con la media y desviación típica de estos pequeños lotes de matrices, ya que lo habitual es utilizar la media y varianza de todo el conjunto, pero gracias a esta técnica se evita que en el entrenamiento la red no se sobreajuste, esto sucede cuando un modelo se ajusta a los datos de entrenamiento y como consecuencia no generaliza bien en los datos del conjunto de prueba obteniendo resultados satisfactorios en el entrenamiento (*train*) y un bajo desempeño en la prueba *test* (Alvaro, 2020).

El segundo paso que se implementa se visualiza en la Figura 2.15 como una recta gris horizontal, se conoce como "concatenación (*concatenating*, en inglés)" los mapas de características que se obtiene al final de cada proceso de convolución (al aplicar los kernel y la función de activación ReLU) serán recortados para ajustarlos y pegarlos a los mapas de características en la fase de decodificación de cada convolución aumentando el doble los mapas, realizar esto será como darle "pistas" a la red para obtener una menor perdida en la clasificación (Ronneberger y cols., 2015).

2.3.3. Decodificación

Esta fase tiene como objetivo entregar la imagen procesada en una segmentada, con la cantidad de clases definidas que aprendió por los patrones encontrados como dimensiones aumentadas. Esta imagen procesada está bastante comprimida, la RNC con la función de activación Softmax es capaz de clasificar con solo la fase de codificación, pero como resultado se obtendrá una imagen bastante reducida que dificulta el entendimiento de esta, se conoce como cuello de botella pues este procedimiento es limitado. Por ello (Ronneberger y cols., 2015) diseñó esta fase para expandirla y obtener una imagen segmentada con dimensiones 16 veces más grande que entrega una por RNC, en la Figura 2.15 U-Net submuestrea 4 veces, por lo que reduce dimensiones 16 a $\frac{n}{16} \times \frac{m}{16}$ (además de la reducción al aplicar un kernel) y se observa una imagen de entrada con dimensiones 572×572 a cambio poseemos una de 388×388 segmentada, es decir que en todo el procedimiento de segmentar la imagen reduce sus dimensiones un 32 % y si observamos la parte de RNC se logra reducir la red a dimensiones 28×28, pues entrega una imagen reducida un 95 %. Es por esto por lo que U-Net fue una arquitectura innovadora, ya que no solo logro aumentar las dimensiones de la red sino también mantener el aprendizaje.

Los autores (Ronneberger y cols., 2015) no definen la técnica empleada en detalle para este procedimiento, ya que se centran en comparar sus resultados con otras arquitecturas y demostrar su eficacia, no solamente en métricas de desempeño también en velocidad de procesamiento de imágenes. Cada paso en la fase de decodificación consiste en una expansión con "sobremuestreo" del mapa de características seguido de una "convolución de 2×2" que reduce a la mitad el número de mapas (los k mapas utilizados en cada convolución), se aplica la concatenación (mencionada antes) correspondiente en la parte codificación y "convoluciones de 3×3", cada una seguida de la función de activación ReLU. Finalmente, en la capa final se utiliza una "convolución 1×1" para mapear cada vector de características, es decir los k mapas, al número deseado de clases.

Existen diferentes artículos que tratan de definir esta técnica y en estos se nombra de maneras distintas como: desconvolución, convolución transpuesta, convolución separable, entre otros. La más apropiada es convolución transpuesta por el objetivo de ir en inversa, ya que desconvolución se entiende que se esta deshaciendo lo aprendido. La convolución separable entrega una técnica apta para este proceso, pero los autores que definen la arquitectura no utilizan este nombre, este es un principal problema para entender esta red, porque el nombre del método puede representar otros nombres o puede poseer otro método. (García Higuera y cols., 2020), define la convolución transpuesta como el sentido contrario de una convolución, pero resulta emplear una técnica diferente, porque hace ver la expansión con el uso del kernel, pues para reducir dimensiones se utiliza *Max-pooling* que es una técnica de submuestreo y este define el uso de una técnica de sobremuestreo para expandir los mapas.

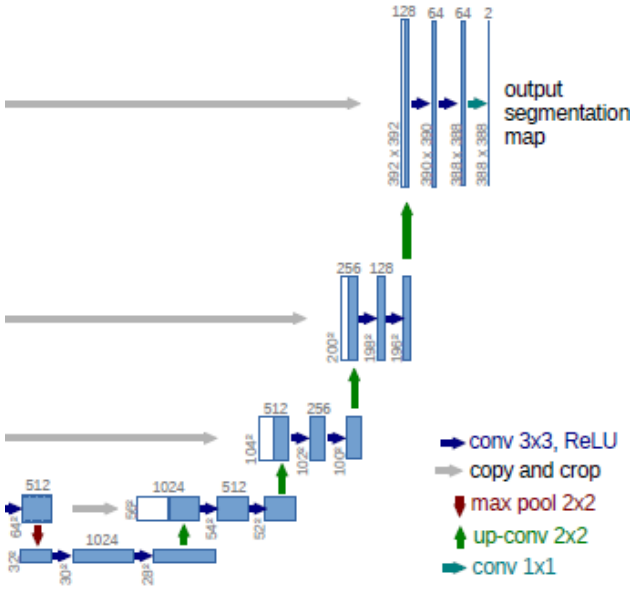


Figura 2.16: Fase de decodificación de la red, utiliza convoluciones 2x2 para expandir y reducir el mapa de características. Tomado de (Ronneberger y cols., 2015).

A continuación, se explicara el procedimiento adecuado de la fase decodificación que se observa en la Figura 2.16.

Sobremuestreo

Al igual que en la codificación se utiliza una técnica para reducir los mapas, en la parte de decodificación se utiliza la técnica *Max-unpooling*, que expandirá los mapas tomando un grupo de píxeles $d \times d \times t$ (donde t es decidido por el diseñador) tomando los valores representativos y expandiendo los mapas al doble (Temp, 2020). En la Figura 2.17 se observa el uso de ambas técnicas en un mapa 4×4 y 2×2, cabe destacar que la expansión mantendrá los valores en la misma coordenada, ya que estas las mantiene constantes en las convoluciones de la fase codificación.

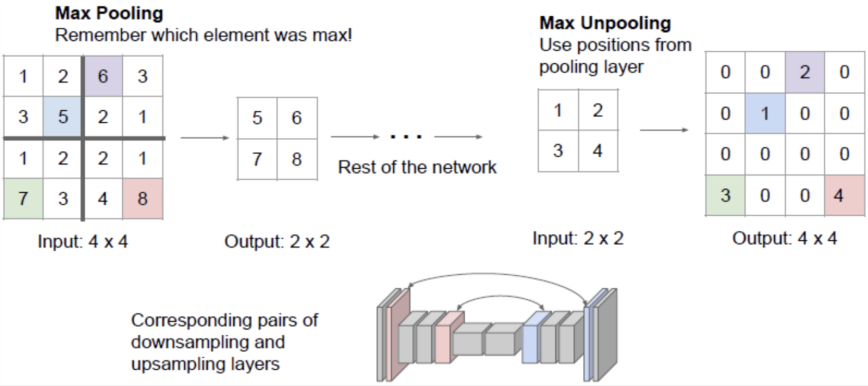


Figura 2.17: Comparación entre *Max-pooling* y *Max-unpooling* sin relleno con un grupo de píxeles de 2×2. Tomado de (Temp, 2020).

En la fase codificación se explica el uso de técnicas de manera bidimensional, es decir $m \times n \times k$ donde $m \times n$ es un vector, pues ahora se hace uso de la tridimensionalidad para explicar estas técnicas que se usan en la decodificación, porque combinan los mapas para destacar las características y también para reducirlos.

El *Max-unpooling* será entonces de dimensiones $2 \times 2 \times 2$, pues se toman 2 mapas y de estos tomaremos el valor máximo para agregar a la nueva matriz (el mapa de características), como observamos en la figura de ejemplo de esta técnica, recorrerá de la misma manera que en *Max-pooling* de cuadro en cuadro tamaño 2×2 abarcando todo el mapa. El resultado es una matriz $(2 * n) \times (2 * m) \times \frac{k}{2}$. Si continuamos con el procedimiento, decir las convoluciones y submuestreos terminamos con la mitad de los mapas de características y como consecuencia se pierde información, por ello se aplica la concatenación que iguala la cantidad de mapas como en la parte simétrica de la codificación y así se reduce la perdida.

Convoluciones

El procedimiento es parecido que en la fase codificación, la diferencia se encuentra en la cantidad de mapas que opera el kernel, la codificación el kernel operaba con todo el grupo de mapas k con un grupo de píxeles 3×3 y obtener un mapa único, esta vez k en las dimensiones del kernel sera igual a 2, porque de esta manera se reduce a la mitad los mapas de características y se desplaza de igual manera como se menciono en la Figura 2.7. En la Figura 2.18 se observa como un kernel de dimensiones $3 \times 3 \times 3$ se desplaza por un mapa de características, este reduce 3 mapas en solo uno.

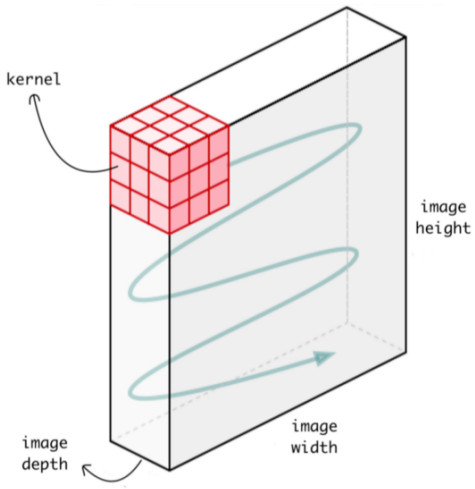


Figura 2.18: Ejemplo desplazamiento de un kernel $3 \times 3 \times 3$ de a un píxel por todo un mapa de características. Tomado de (Verma, 2021).

También en la Figura 2.19 se observa un ejemplo de la toma de píxeles de los k mapas para la obtención del valor único, esta obtención de mapas que operan con un kernel esta generalizado, ya que en la convolución de la codificación se hace uso de todo el mapa contra un kernel, pero se puede especificar la cantidad de mapas que se requiera utilizar para operar contra un kernel y reducir esta cantidad. Por este procedimiento la cantidad de mapas se reduce a la mitad y como resultado se tiene $(n - p + 1) \times (m - p + 1) \times \frac{k}{2*2}$ seguido de una activación ReLU.

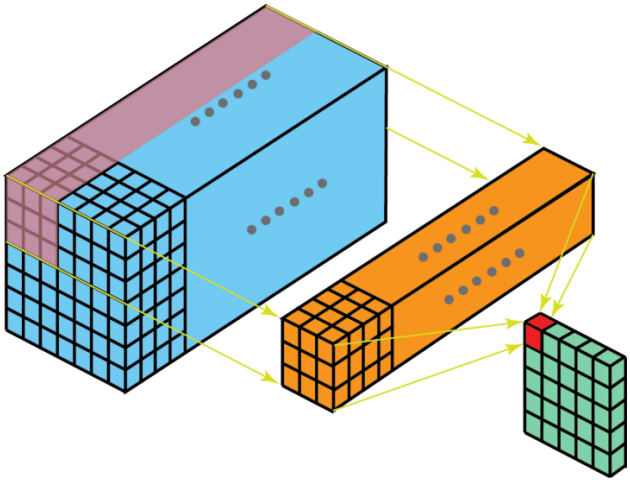


Figura 2.19: Ejemplo de operación de un kernel $3 \times 3 \times k$ contra un grupo de píxeles $3 \times 3 \times k$ de un mapa de características, donde k son los mapas. Tomado de (*Tipos de núcleos de convolución: simplificados*, 2020a).

La cantidad de convoluciones que se realicen en la decodificación depende de la cantidad que se realicen en la codificación, para mantener la simetría de la arquitectura y la información (concatenación). Este procedimiento cuenta con una convolución final de $1 \times 1 \times c$ que mapear los

mapas de características para reducir la cantidad al número de clases de la imagen, donde el valor de c dependerá de la cantidad de clases que se tenga definida, pues si se posee R clases entonces para calcular c hay que tener en cuenta la cantidad de mapas que se tiene en la última convolución de la decodificación y dividirla por la cantidad de clases y se obtiene c, utilizando lo anterior la convolución de $1 \times 1 \times (\frac{k}{R})$. Este tipo de convolución mantiene las dimensiones, pero disminuye la cantidad de mapas operando por "producto escalar".

Otra manera de ver esta convolución es operar los grupos de píxeles $3 \times 3 \times k$ (donde k son mapas obtenidos en la codificación) contra un kernel tamaño $3 \times 3 \times k$ para obtener el valor único que se desplazara para formar el nuevo mapa único y así aplicar el kernel la misma cantidad de veces que se especifico en la codificación, pues de esta manera será parecida, porque utilizaremos varios kernels para filtrar estos mapas de características y se utiliza la cantidad simétrica de kernels en la parte de codificación, es decir que si en la última convolución de la codificación fue de 264, la cantidad que se utilizara en la primera convolución en la fase decodificación será la misma e ira reduciendo a la mitad. Este procedimiento se le conoce como "convolución separable" el uso de estas lo hace favorable en el *deep learning* por disminuir la cantidad de parámetros (los canales, filtros, kernels, entre otros) reduciendo requerimiento de poder computacional (*Tipos de núcleos de convolución: simplificados*, 2020b). En la Figura 2.20 se observa este procedimiento, ya que un kernel de $3 \times 3 \times 3$ opera contra un grupo de píxeles 3×3 y con sus 3 mapas (este valor corresponde a k) la cantidad de veces que se aplique el kernel sera la misma en la ultima convolución de la codificación e ira reduciéndose a la mitad por cada sobremuestreo. Se aplica de igual manera en la "convolucion 1×1", pero la cantidad de kernels que se utiliza dependerá de la cantidad de clases que se tenga en el conjunto, entonces el kernel tendrá dimensiones $1 \times 1 \times k$ y se aplicara R veces contra los mapas de características para obtener las mismas dimensiones, pero la cantidad de mapas k sera igual a las de clases R.

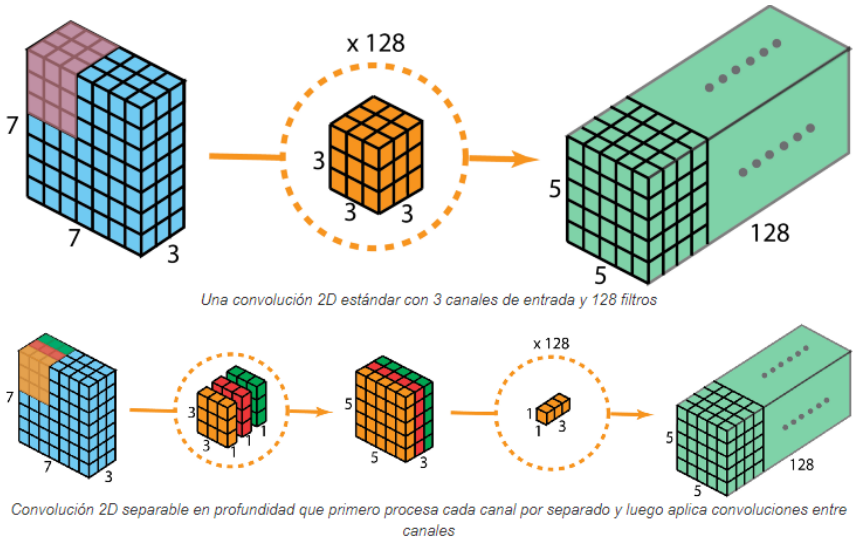


Figura 2.20: Ejemplo bidimensional del procedimiento convolución separable aplicando un kernel de $3 \times 3 \times 3$ 128 veces al mapa de características de tamaño $7 \times 7 \times 3$ para obtener un nuevo mapa de tamaño $5 \times 5 \times 128$ (*Tipos de núcleos de convolución: simplificados*, 2020b).

Esta manera de expandir las dimensiones ha sido adquirida por diversas arquitecturas

actuales, ya que el método de codificar y decodificar resultó ser innovador para la segmentación de imágenes por resolver el problema de cuello de botella.

2.4. DeepLab

La arquitectura U-Net comenzó una revolución en el desarrollo de arquitecturas de segmentación dando como base la codificación y decodificación en la segmentación de imágenes, una arquitectura popular actualmente es DeepLab desarrollada por empleados de Google, implementaron y mejoraron métodos de diferentes arquitecturas (Ku, 2020c).

2.4.1. Definición de DeepLab

Los autores y empleados de Google (Chen, Papandreou, Kokkinos, Murphy, y Yuille, 2017) desarrollaron esta arquitectura implementando una técnica llamada "Convolución Atroz (*atrous convolution*, en inglés)", una poderosa herramienta para ajustar el campo de visión del filtro, así como para controlar la resolución de los resultados de características calculadas por las redes neuronales convolucionales (RNC), en la aplicación de la segmentación semántica de imágenes. Además, emplean otro método llamado "Agrupación de pirámides espaciales Atroz (*Atrous Spatial Pyramid Pooling*, en inglés)", que analiza características convolucionales a múltiples escalas, con características a nivel de imagen que codifican el contexto general y mejoran aún más el rendimiento. En la Figura 2.21 podemos apreciar la arquitectura DeepLab basada en la codificación y decodificación, se ingresa una imagen para detectar patrones por la red convolucional, pero hace uso de la convolución atroz que también reduce la imagen para obtener 5 mapas que serán expuestos a la agrupación de pirámides espaciales atroz, como se tienen escalas diferentes la red aprende de los objetos pequeños y grandes para finalmente reducir estos 5 mapas a solo 1, así ingresar a la decodificación que por uso de otro mapa que mantuvo las dimensiones originales se expandirá para igualarlo y combinarse con este, finalmente, por un último submuestreo entregar la imagen segmentada.

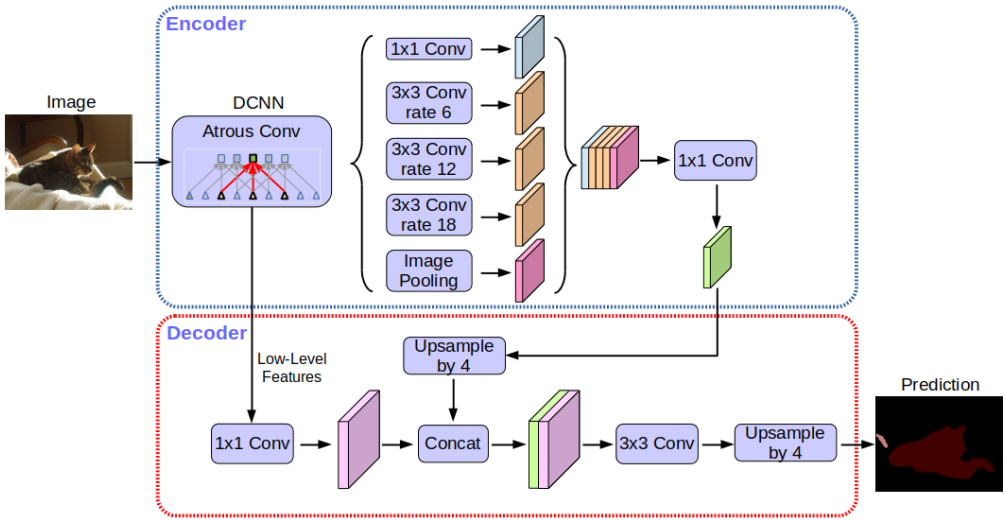


Figura 2.21: Arquitectura Deeplab compuesta por convoluciones atroz y *Atrous Spatial Pyramid Pooling*. Tomado de (Ku, 2020b).

A continuación, se definen los dos métodos que se utilizan en la codificación de la arquitectura DeepLab una herramienta potente en la arquitectura de segmentación.

2.4.2. Codificación

Convolución Atroz y Agrupación de pirámides espaciales Atroz

Esta técnica, también conocida como "convolución dilatada (*Dilated Convolution*, en inglés)" la emplea la arquitectura PSPNet, cuyo objetivo es expandir el campo receptivo para obtener más información y conservar la resolución (Ku, 2020c). DeepLab emplea esta como un reemplazo del submuestreo, porque al comprimir una imagen se está deshaciendo de información relevante para la segmentación, la convolución atroz reducirá la imagen, pero no la cantidad como se mostró en el submuestreo que reduce a la mitad las dimensiones, como consecuencia requiere mayor potencia, es decir que se sacrifica velocidad por precisión, pero por la utilización de convoluciones separables logra recuperar la velocidad de procesamiento perdida. La diferencia con la convolución que utiliza U-Net se encuentra en el kernel, se agrega un nuevo parámetro (además de las dimensiones $n \times m \times k$) llamado "*rate*" este aumentara el campo receptivo que posee el kernel agregando un espaciado entre los valores. En la Figura 2.22 se observa que un kernel de *rate*$= 1$ es el utilizado en RNC y U-Net, pero al ser de *rate*$= 2$ su campo receptivo aumenta tomando dimensiones 5×5, pero no deja de ser un kernel de 3×3.

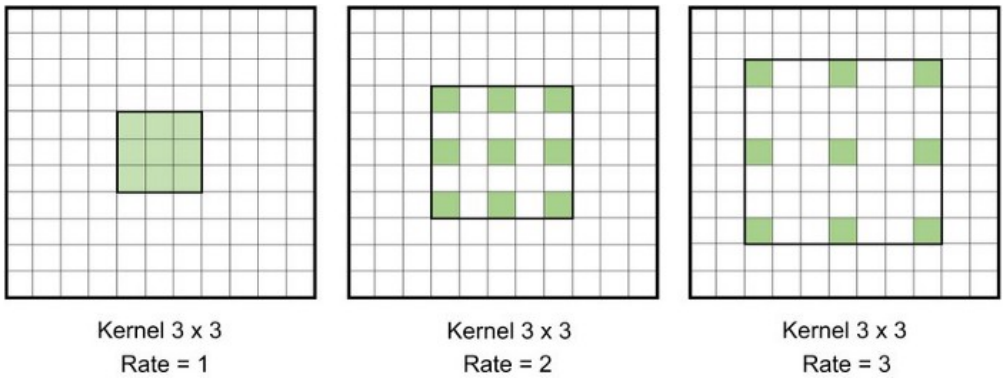


Figura 2.22: Ejemplo Convolución atroz o dilatada de 3 diferentes *rate*. Tomado de (Ku, 2020a).

Por cada convolución atroz (seguida de una activación ReLU) que se aplique este incrementa su *rate*, en la Figura 2.21 se observa que se utilizan 5 convoluciones al igual que la cantidad de U-Net (Figura 2.15), la diferencia que algunas poseen *rate*, solo dos convoluciones no utilizan el *rate*, estos mapas poseen dimensiones diferentes por el cálculo de $(n-p+1) \times (m-p+1)$, esto permite a la red poseer diferentes perspectivas de los objetos que se presenten en la imagen, porque en la convolución 1×1 permite una información global y en *image pooling* (corresponde al mapa de menor tamaño en el grupo) se aprecia la información local en un campo pequeño (Ku, 2020a). Para concatenar estos mapas se utiliza la técnica de sobremuestreo (ver Figura 2.17) para expandir los mapas en el grupo y que posean el tamaño adecuado para unirse, finalmente por una convolución 1×1 estos 5 mapas se hacen uno, este procedimiento es conocido como "Agrupación de pirámides espaciales Atroz", además se realiza una convolución 1×1 adicional que no se adentra en la agrupación que pasa directamente a la decodificación.

2.4.3. Decodificación

Como la red codificó la imagen solo queda expandir los mapas que utilizó para aprender (ver Figura 2.21), por el mapa obtenido en la agrupación debe de ser tratado con otro sobremuestreo y así ser unido con el mapa que no ingresó a la agrupación, porque es el guía para obtener dimensiones similares a la imagen de entrada, aplicar convolución 3×3 y un ultimo submuestreo 4×4, como la arquitectura no utiliza el submuestreo no hay necesidad de aplicar una cantidad sucesiva para clasificar mediante la función de activación Softmax, el ajuste de pesos se realiza por el *backpropagation* para reducir la pérdida (Pal, 2020). La sencillez en estas arquitecturas les entrega la facilidad de procesar las imágenes en un tiempo considerado y mantener la balanza entre precisión y velocidad.

Capítulo 3

Métricas de desempeño de los modelos

Para desarrollar un modelo eficiente se debe primero interpretar sus resultados, pero no bastará solamente con nuestro criterio, es por ello por lo que existen diversas métricas para evaluar el desempeño del modelo y dar valor a sus resultados, es decir que las métricas entregan confiabilidad al momento de predecir o clasificar un conjunto de datos.

Para el modelo de segmentación semántica existen dos criterios comunes para evaluar el desempeño de entrenamiento (*train*) y validación (*valid*), estos corresponden a la exactitud (también se le conoce como *accuracy*, en inglés) esta medición es una de las más ocupadas para evaluar modelos y la intersección sobre unión (*intersection over union (IoU)*, en inglés) es la principal en la medición de clasificación o detección en imágenes. Sin embargo, existe otra medida a utilizar que es la pérdida (*loss*, en inglés) este indicador le dirá al modelo que tan preciso esta siendo al clasificar el conjunto de datos, en cada entrenamiento los parámetros o pesos se ajustaran para reducir la pérdida, es decir si en la iteración t es mayor que $t-1$, el modelo deberá ajustar con nuevos valores los pesos para que $t+1$ sea menor y así sucesivamente hasta obtener un modelo con pérdida cercana a cero. Utilizaremos dos perdidas que serian la entropía cruzada (*cross entropy loss*, en inglés) y Coeficiente de perdida (*Dice Loss*, en inglés). Para evaluar el desempeño del conjunto de prueba (*test*) se utilizaran otras métricas además de las mencionadas que corresponden a la precisión (*precision*, inglés), Exhaustividad (*recall*, inglés) y Valor-F (*f1-score*, inglés) (Heras, 2020).

Para la compresión del funcionamiento de estas métricas se definen la matriz de confusión, ya que sus componentes son utilizadas para calcular las métricas mencionadas.

3.1. Matriz de confusión

Con la matriz de confusión se almacena la información comparando la predicción o clasificación del modelo con los datos reales, evaluara cada etiqueta de la clase como positivo o negativo. La matriz de confusión posee un tamaño $n \times n$, el valor de n dependerá de la cantidad de clases que se quiera clasificar (Chauhan, 2020). La estructura de la matriz corresponde a la siguiente.

		Predicción	
		Positivo	Negativo
Reales	Positivo	TP	FT
	Negativo	FN	TN

Cuadro 3.1: Matriz de Confusión de 2x2 para datos binarios.

Donde,

- TP: Los valores verdaderos positivos, la etiqueta de clase clasificada como clase.
- TN: Los valores verdaderos negativos, la etiqueta nula clasificada como nula.
- FP: Los valores falsos positivos, la etiqueta nula clasificada como clase.
- FN: Los valores falsos negativos, la etiqueta clase clasificada como nula.

3.1.1. Métricas de evaluación

- Exactitud: Mide la cantidad de aciertos que tuvo el modelo. Se define como:

$$\text{Exactitud} = \frac{TP + TN}{TP + FP + FN + TN} \,. \tag{3.1}$$

- Intersección sobre unión: Calcula la superposición de las clases, la real y la clasificada. Se define:

$$\text{IoU} = \frac{TP}{TP + FP + FN} \,. \tag{3.2}$$

También se interpreta como:

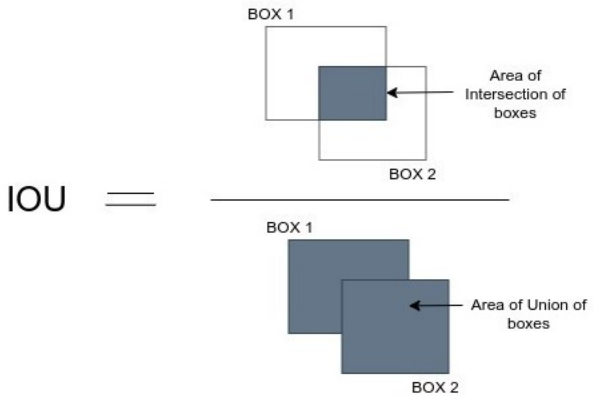


Figura 3.1: Representación gráfica del cálculo IoU, la intersección de un conjunto A y B es dividido por su unión. Tomado de (Subramanyam, 2021).

- Precisión: Mide la cantidad de aciertos, pero solamente positivos, se define como:

$$\text{Precisión} = \frac{TP}{TP + FP} \,. \tag{3.3}$$

- Exhaustividad: Nos indica la cantidad que es capaz de identificar el modelo, se define como:

$$\text{Exhaustividad} = \frac{TP}{TP + FN} \,. \tag{3.4}$$

- Valor-F: Es la combinación entre la precisión y exhaustividad por el cálculo de la media armónica, se define como:

$$\text{Valor-F} = 2 \cdot \frac{\text{Precisión} \cdot \text{Exhaustividad}}{\text{Precisión} + \text{Exhaustividad}} \,. \tag{3.5}$$

3.2. Función de pérdida

La función de pérdida, también conocida como "función objetivo" ayuda a medir el proceso del entrenamiento y validación, así se sabrá qué tan lejos se está del "objetivo", por ello su influencia al resto de métricas, es decir si la perdida es cercana a cero, las métricas de desempeño serán elevadas (satisfactorias), pero al ser mayor a uno, el desempeño será bajo y se verá reflejado en el resto de las métricas. El algoritmo busca cómo disminuir este valor actualizando los pesos por el *backpropagation*, por tanto no solo es un valor para interpretar su desempeño, también es un valor que la red deberá de minimizar (Gavilán, 2020). En este trabajo se utilizan dos funciones de pérdida:

- Entropía Cruzada (*Cross Entropy*, en inglés): Se utiliza en problemas de clasificación, en general, es una medida de la distancia entre distribuciones de probabilidad, es utilizada en modelo que en la capa de salida posean probabilidad (Gavilán, 2020), se define como:

$$\text{Entropía Cruzada} = -\sum_{i=1}^{n} t_i \log(p_i) \,, \tag{3.6}$$

donde n es el número de clases, t_i es el indicador binario (0 la clase que no corresponde y 1 para la clase correcta), p_i es la probabilidad obtenida de la función Softmax para la clase i-ésima

- Coeficiente de pérdida (*Dice Loss*, en inglés): es una métrica común para la segmentación de píxeles que también se puede modificar para que actúe como una función de perdida (*Loss Function Library - Keras PyTorch*, 2021), se define como:

$$\text{Coeficiente de pérdida} = \frac{2|X \cap Y|}{|X| + |Y|} \,, \tag{3.7}$$

donde X corresponde a la observación predicha e Y a la clase perteneciente.

Capítulo 4

Materiales y métodos

En esta sección se presenta el conjunto de imágenes obtenidas por dron y satélite, también se explica la construcción de las arquitecturas de segmentación.

4.1. Conjunto de datos

Kaggle es una plataforma de *Data Science* donde millones de usuarios que representan institutos, empresas o a sí mismos, adjuntan conjuntos de datos, desafíos con premios, problemas de programación. Para la obtención de un conjunto de imágenes que posean una cantidad considerable de imágenes y un conjunto de imágenes etiquetadas (mascara), Kaggle es el sitio principal para obtener lo necesario.

4.1.1. Imágenes aéreas captadas por dron

El Instituto de Gráficos y Visión por Computadora de Austria (*ICG - DroneDataset*, s.f.) facilita 400 imágenes (4 gigabytes) obtenidas por un dron en lugares urbanos del país, cuenta con 23 clases donde se visualizan árboles, césped, agua, rocas, personas, animales, entre otros. El propósito de esté conjunto de imágenes se centra en aumentar la seguridad de vuelo y aterrizaje de drones autónomos. Las imágenes fueron captadas por una cámara de alta resolución entregando imágenes de dimensiones 6000x4000, no cuenta con agrupaciones para el entrenamiento, validación y prueba. A continuación, en las Figuras se puede visualizar algunas imágenes del conjunto de datos.

(a) Imagen original captada por dron.

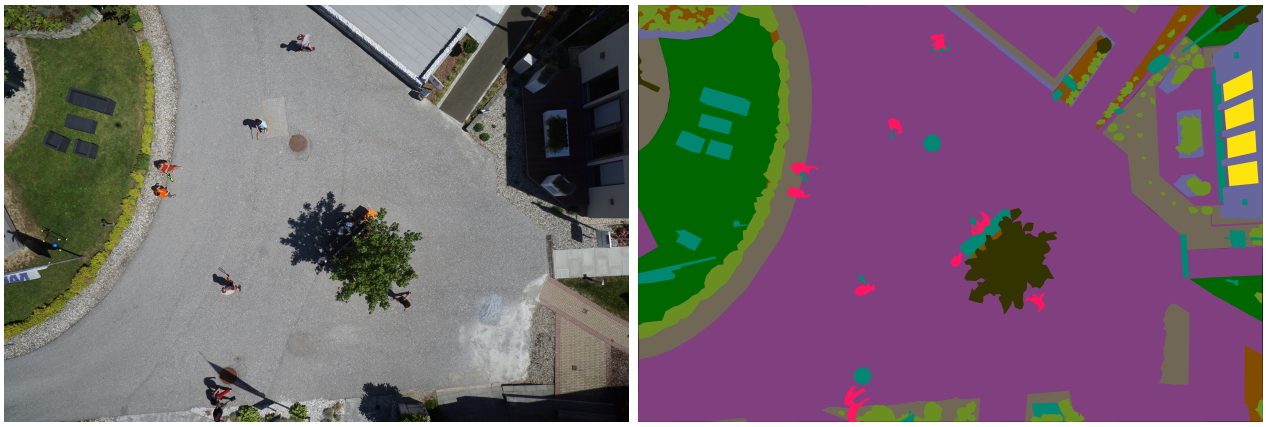
(b) Mascara con sus respectivas etiquetas.

Figura 4.1: Ejemplo de imagen captada por dron (con sus etiquetas y tamaño 6000x4000 píxeles). Tomade de (*ICG - DroneDataset*, s.f.).

(a) Imagen original captada por dron.

(b) Mascara con sus respectivas etiquetas.

Figura 4.2: Ejemplo de imagen captada por dron (con sus etiquetas y tamaño 6000x4000 píxeles). Tomade de (*ICG - DroneDataset*, s.f.).



(a) Imagen original captada por dron.

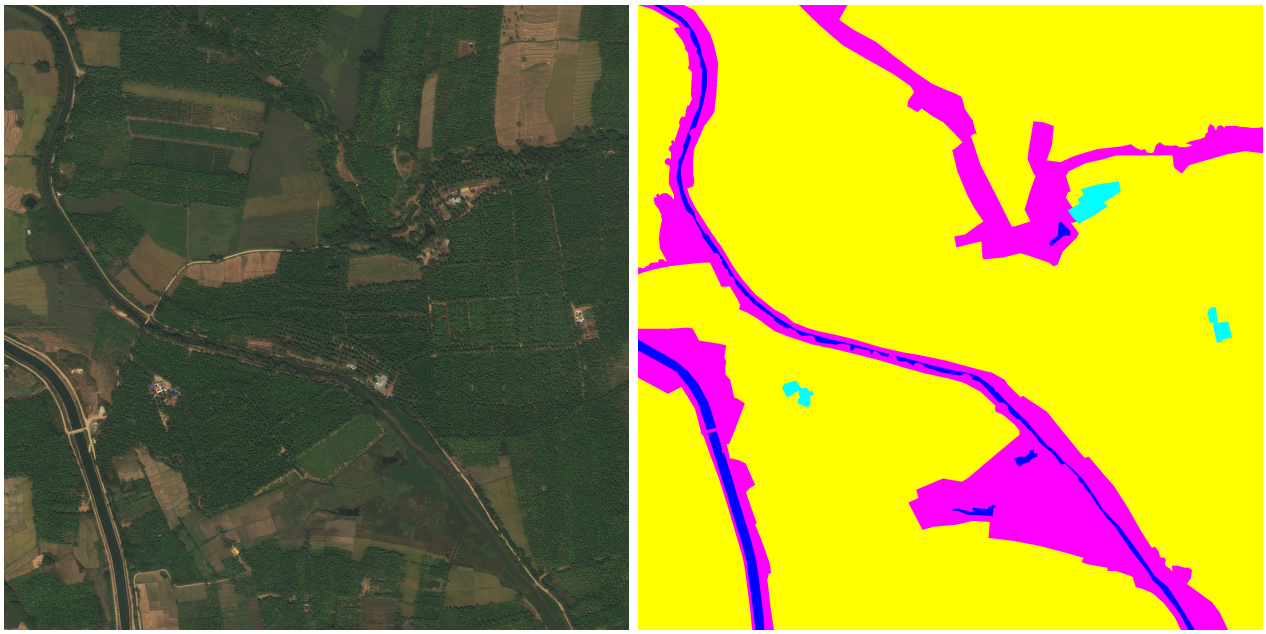
(b) Mascara con sus respectivas etiquetas.

Figura 4.3: Ejemplo de imagen captada por dron (con sus etiquetas y tamaño 6000x4000 píxeles). Tomade de (*ICG - DroneDataset*, s.f.).

En las Figuras de las imágenes captadas por drones, se aprecia que las máscaras son multiclase y la que domina corresponde al pavimento (etiqueta "morada") y al césped (etiqueta "verde") para las arquitecturas se les será sencillo conocer los patrones de estas etiquetas. En estas mascaras se aprecia el trabajo que conlleva etiquetar los objetos para obtener la compresión completa del contexto de la imagen, la dificultad de desarrollar estas mascaras es complicado.

4.1.2. Imágenes aéreas captadas por satélite

La clasificación y segmentación autónoma de la cobertura del suelo es de gran importancia para el desarrollo sostenible, la agricultura autónoma y la planificación urbana. *DeepGlobe Land Cover Classification Challenge* presenta el desafío de la clasificación autónoma de los tipos de cobertura del suelo. Este problema se define como una tarea de segmentación multiclase para detectar un total de 8 clases entre estas urbanas, agrícolas, de pastoreo, forestales, de agua, áridas y desconocidas (Demir y cols., 2018). Cuenta con 803 imágenes (3 gigabytes) de dimensiones 2448x2448, con la agrupación de entrenamiento, validación y prueba, pero en esta no se facilita la mascara. A continuación, observamos algunos ejemplos del conjunto de imágenes.

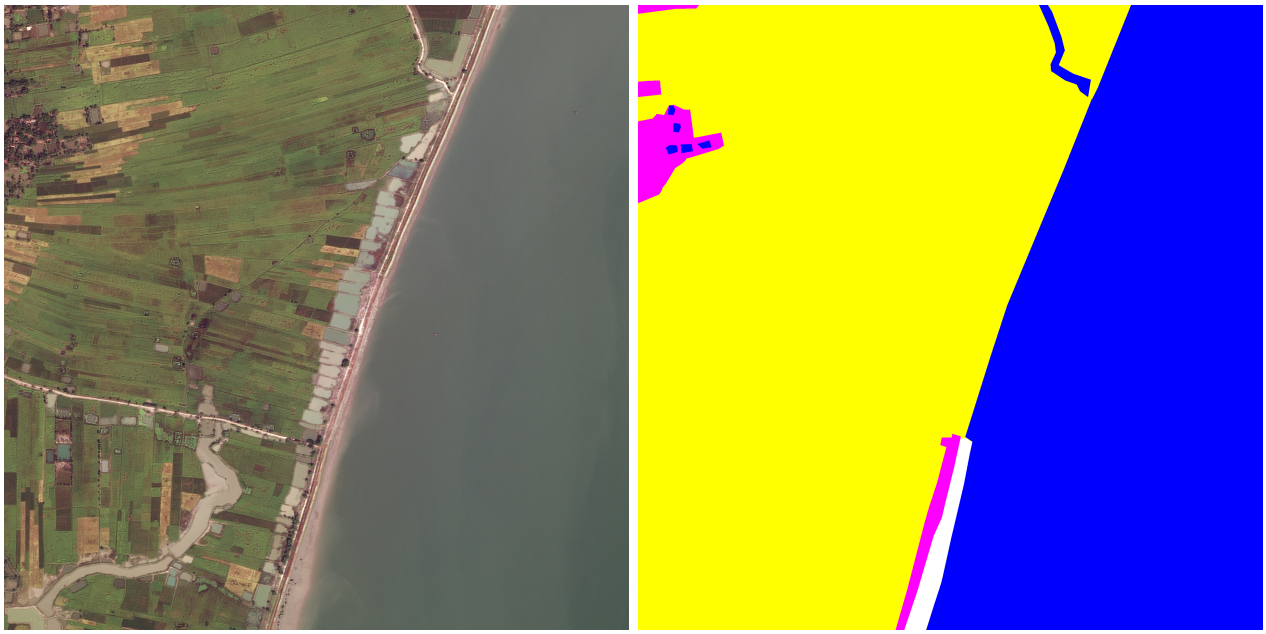


(a) Imagen original captada por satélite. (b) Mascara con sus respectivas etiquetas.

Figura 4.4: Ejemplo de imagen captada por satélite (con sus etiquetas y tamaño 2448x2448 píxeles). Tomada de (Demir y cols., 2018).

(a) Imagen original captada por satélite. (b) Mascara con sus respectivas etiquetas.

Figura 4.5: Ejemplo de imagen captada por satélite (con sus etiquetas y tamaño 2448x2448 píxeles). Tomada de (Demir y cols., 2018).



(a) Imagen original captada por satélite.

(b) Mascara con sus respectivas etiquetas.

Figura 4.6: Ejemplo de imagen captada por satélite (con sus etiquetas y tamaño 2448x2448 píxeles). Tomada de (Demir y cols., 2018).

Estas Figuras mostradas se aprecia que son multiclases, pero solo algunas, en la Figura 4.5b se observa solo 2 clases, esto puede complicar el entendimiento del modelo al clasificar, porque en la imagen original se analiza una complejidad de distintos elementos y esto se debe a que el conjunto es desarrollado para solamente identificar ciertas clases, es decir que no se estaría tratando con una segmentación de multiclase, pues corresponde a una binomial.

4.2. Metodología

Para este trabajo se utiliza el lenguaje de programación *Python* por medio de Kaggle, que ofrece un servicio para correr códigos, ya que al trabajar con imágenes se requiere un poder computacional considerado, la utilización de la GPU nos ayuda con el gran peso que requiere el proceso de los volúmenes de imágenes que se tiene, pues *Python* ofrece sencillez y potencia para utilizar y desarrollar poderosas herramientas, además, su amplia selección de librerías y *frameworks* facilita la construcción de estas arquitecturas de segmentación. *Pytorch* es una biblioteca de aprendizaje automático, cuenta con diversas librerías que ayudan a implementar las arquitecturas y también al preprocesado del conjunto de imágenes. Otra biblioteca que se utiliza es "*OpenCV*", diseñada para resolver problemas de visión por computadora entregando una diversidad de librerías para ajustar colores y destacar características. Por último, también se uso la biblioteca *Albumentations* que es una herramienta de la visión por computadora, está es dependiente de *OpenCV*, ya que admite muchos formatos de imagen. Por el uso de estas bibliotecas se puede implementar las arquitecturas de segmentación.

4.2.1. Preprocesado

Antes de ingresar los conjuntos de imágenes se debe de dividir el conjunto para entrenar, validar y la probar. Como el conjunto de prueba en la imagen satelital no cuenta con máscaras no se podrá evaluar el desempeño, se deberá mezclar el conjunto de entrenamiento y validación para poseer un conjunto de prueba para evaluar tal desempeño. La división de los conjuntos es:

- Para el conjunto de imágenes aéreas captadas por dron:
  - Entrenamiento: 306 imágenes.
  - Validación: 54 imágenes.
  - Prueba: 40 imágenes.

- Para el conjunto de imágenes aéreas captadas por satélite:
  - Entrenamiento: 643 imágenes.
  - Validación: 80 imágenes.
  - Prueba: 80 imágenes.

Para el ajuste final del conjunto de imágenes se hace uso de las bibliotecas *Pytorch*, *OpenCV* y *Albumentations*, la librería *Normalize* para normalizar el conjunto de datos, *COLOR_BGR2RGB* para ordenar los canales de colores "azul, verde y rojo" a "rojo, verde y azul" y *RandomCrop* para ajustar píxeles al azar al conjunto de entrenamiento y evitar un sobreajuste. De esta manera el conjunto de imágenes está preparado para ingresar a las arquitecturas de segmentación.

4.2.2. Implementación de las arquitecturas

Para la construcción tanto de U-Net como DeepLab se hace uso de la librería "*segmentation models*", los parámetros que utilizamos en la codificación corresponderían a "*mobilenet*" (Sandler, Howard, Zhu, Zhmoginov, y Chen, 2018) que utiliza la convolución separable (ver figura 2.19) con la función de activación ReLU6, esta función se diferencia con ReLU en que los valores que sean mayores a 6 devuelven su valor a 6. En cuanto a los pesos se utiliza unos preentrenados del *deep learning* para obtener un mejor desempeño en la segmentación por ello se uso "imagenet" (Fei-Fei, Deng, y Li, 2009) y por último la cantidad de clases a segmentar correspondiente a cada conjunto, 23 en dron y 8 para satélite. Para U-Net hacen falta dos parámetros más, que corresponden a la cantidad de codificaciones que se realizan (un total de 5) y el número de kernels que se aplican en la cada convolución 16 (en la primera), 32 (en la segunda), 64 (en la tercera), 128 (en la cuarta) y 256 (en la quinta), con esto concluye la construcción de las arquitecturas. El modelo de segmentación corresponde a la iteración que posea la menor perdida en la validación, porque es de interés que el modelo clasifique de manera satisfactoria con datos desconocidos, por lo tanto, si en el tiempo este llega aumentar se guardara el modelo con mayor pérdida, provocando que la eficacia en el conjunto de prueba no sea satisfactoria.

Kaggle permite la activación del acelerador por GPU, pero solo permite su uso a lo sumo 9 horas para guardar las salidas de un código, por esta limitación la cantidad de iteraciones para

entrenar ambas arquitecturas con el conjunto de imágenes por dron son de un total de 100 iteraciones (cubre 8 horas).

La cantidad de iteraciones para el conjunto de imágenes por satélite es de 30, dado a que poseemos el doble de imágenes a diferencia del conjunto del dron (cubre 8.5 horas).

Luego de entrenar ambas arquitecturas se implementan las métricas de desempeño para su evaluación y comparación.

Capítulo 5

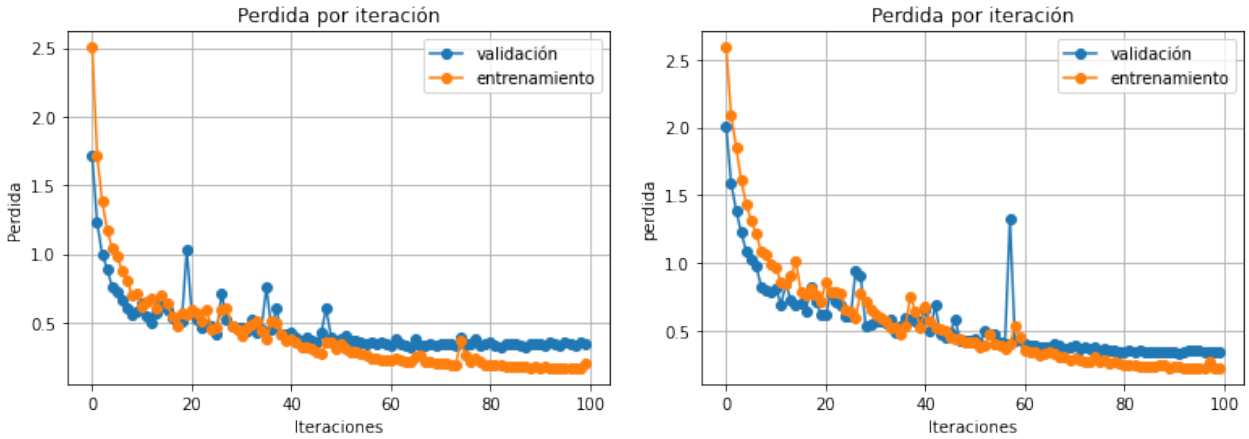
Resultados

En esta sección se evalúa y comparan los resultados obtenidos de las arquitecturas de segmentación sobre los conjuntos de imágenes captadas por drones y satélites.

5.1. Resultados: Imágenes aéreas captadas por dron

5.1.1. Función de Pérdida

DeepLab posee un mayor arranque que U-Net. DeepLab disminuye su pérdida a 0.5 antes de las 50 iteraciones y se mantiene estable en el resto del periodo. U-Net logra estabilizarse bajo 0.5 después de la iteración 60, la estabilidad indica que las arquitecturas ya no están aprendiendo nada relevante en las imágenes (ver Figura 5.1.



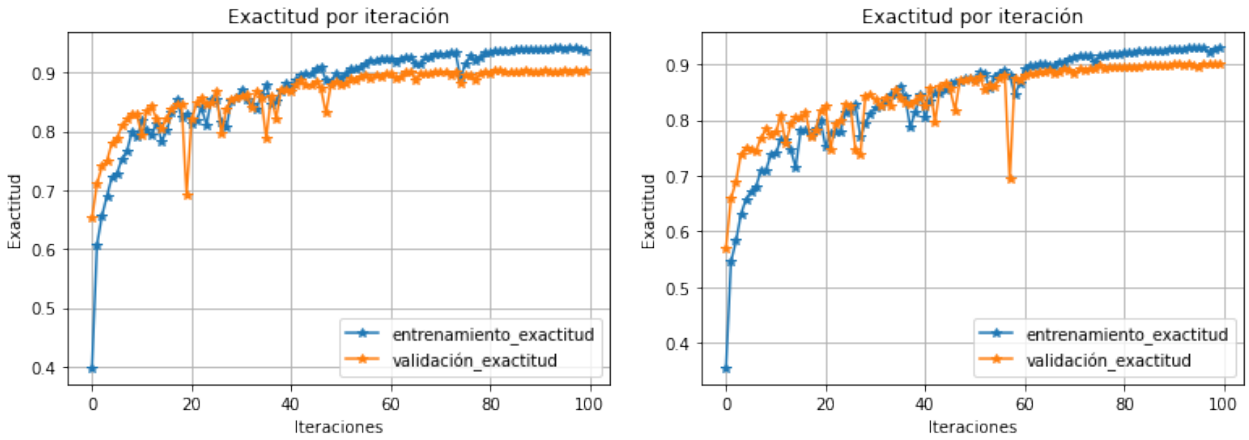
(a) Arquitectura de segmentación DeepLab.

(b) Arquitectura de segmentación U-Net.

Figura 5.1: Desempeño por la función de perdida de entropía cruzada de las arquitecturas de segmentación, 100 iteraciones. Elaboración propia.

5.1.2. Exactitud

Así como en la pérdida, la exactitud presenta el mismo patrón, DeepLab comienza a mantener estabilidad después de la iteración 50 y ocurre lo mismo con la U-Net. Un detalle que también se aprecia en la pérdida son las caídas en la validación, en U-Net se destaca antes de la iteración 60, esto se debe a un sobreajuste en los parámetros de las neuronas provocando que el entrenamiento sea satisfactorio, pero no logra generalizar de manera satisfactoria en la validación (ver Figura 5.2).



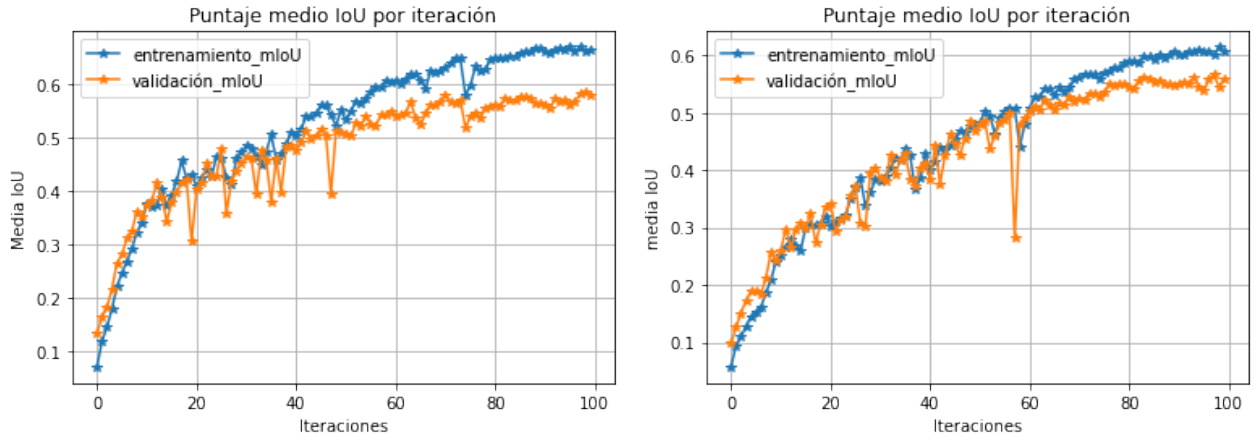
(a) Arquitectura de segmentación DeepLab.

(b) Arquitectura de segmentación U-Net.

Figura 5.2: Desempeño de la métrica exactitud entre $[0, 1]$ de las arquitecturas de segmentación, 100 iteraciones. Elaboración propia.

Intersección sobre Unión (IoU)

Se analiza una pequeña diferencia entre ambas arquitecturas en el desempeño de validación por la media IoU, DeepLab posee un margen mayor de diferencia en cuanto al entrenamiento y la validación, pero en U-Net se observa con un margen de diferencia bastante bajo en comparación de DeepLab, luego de las 80 iteraciones la diferencia es aún mayor, pero el desempeño de la validación es satisfactorio (ver Figura 5.3).



(a) Arquitectura de segmentación DeepLab.

(b) Arquitectura de segmentación U-Net.

Figura 5.3: Desempeño de la métrica media IoU entre $[0, 1]$ de las arquitecturas de segmentación, 100 iteraciones. Elaboración propia.

5.1.3. Segmentación semántica del conjunto de prueba

En el cuadro comparativo (ver Cuadro 5.1) se analizan las medidas de desempeño de las arquitecturas con el conjunto de prueba de las imágenes captadas por dron, ambos modelos logran resultados satisfactorios y la diferencia entre ambos es mínima, U-Net se destaca por solo milésimas.

Metrica	DeepLab	U-Net
Exactitud	0.913	**0.915**
IoU	**0.539**	0.533
Exhaustividad	0.878	**0.881**
Precisión	0.932	**0.934**
Valor-F	0.904	**0.906**

Cuadro 5.1: Resultados de la clasificación de arquitecturas de segmentación con el conjunto de prueba. Elaboración propia.

5.1.4. Ejemplos de segmentación semántica

A continuación, se presentan algunos ejemplos de los resultados de segmentación semántica con el conjunto de prueba de las imágenes captadas por dron. Se utiliza la metrica IoU para evaluar el desempeño en cada ejemplo.

Segmentación semántica por DeepLab

En la Figura 5.4 se observa como el modelo DeepLab detecta patrones, por ejemplo los accesorios que trae una persona consigo afectan en el patrón de segmentación del modelo, porque estos accesorios pasan a ser etiquetados como "persona (verde)" y los objetos los identifica con la etiqueta "amarilla", como la persona del centro posee un accesorio que es un sombrero que posee una estructura similar al paraguas, el modelo (o arquitectura) identifica este accesorio y a la vez a la persona, segmenta al sombrero como un objeto el cual no está mal, esto provoca "confusión", porque si este sombrero no fuese parte de la persona (fuese tratado como un objeto con etiqueta "amarilla"), corresponde a una clasificación correcta. Sucede algo parecido con la falda que lleva la mujer, por ser de una tonalidad parecida al pavimento el modelo segmenta la falda con esta etiqueta, el patrón que se encontró para segmentar el pavimento es por la tonalidad.

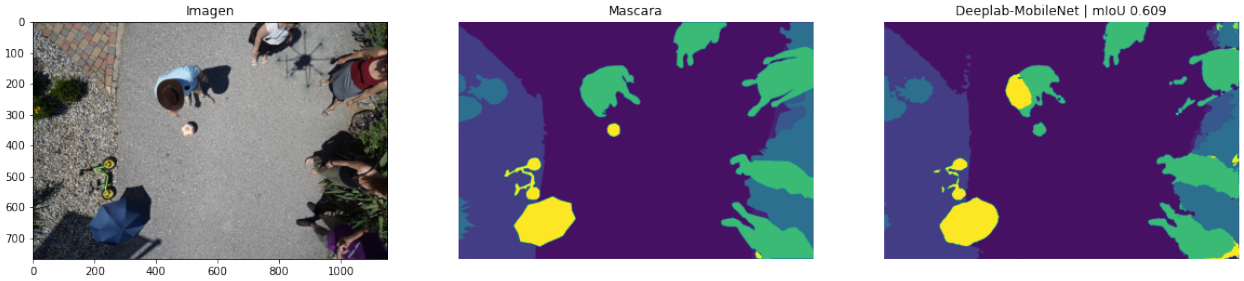


Figura 5.4: Resultado de segmentación semántica a una imagen del conjunto de prueba, se obtiene una media IoU 60.9 %, arquitectura DeepLab. Elaboración propia.

Figura 5.5: Resultado de segmentación semántica a una imagen del conjunto de prueba, se obtiene una media IoU 62 %, arquitectura DeepLab. Elaboración propia.

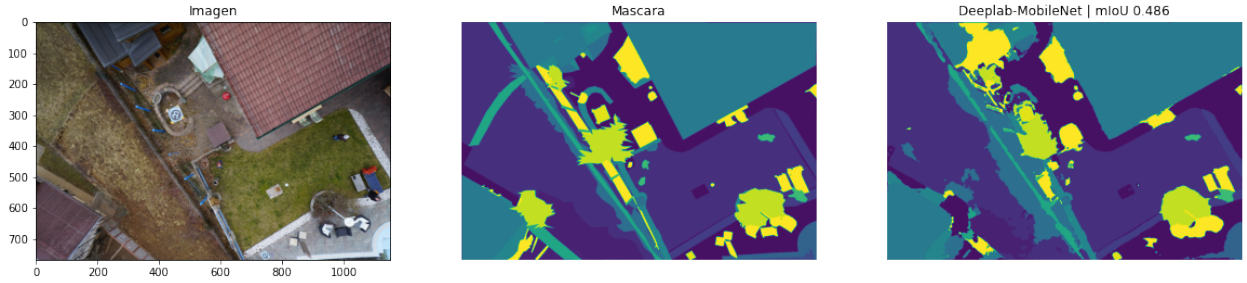


Figura 5.6: Resultado de segmentación semántica a una imagen del conjunto de prueba, se obtiene una media IoU 48.6 %, arquitectura DeepLab. Elaboración propia.

5.1.5. Segmentación semántica por U-Net

Lo descrito de la Figura 5.4 se aplica en la Figura 5.9, el modelo U-Net comprendió patrones similares a DeepLab.

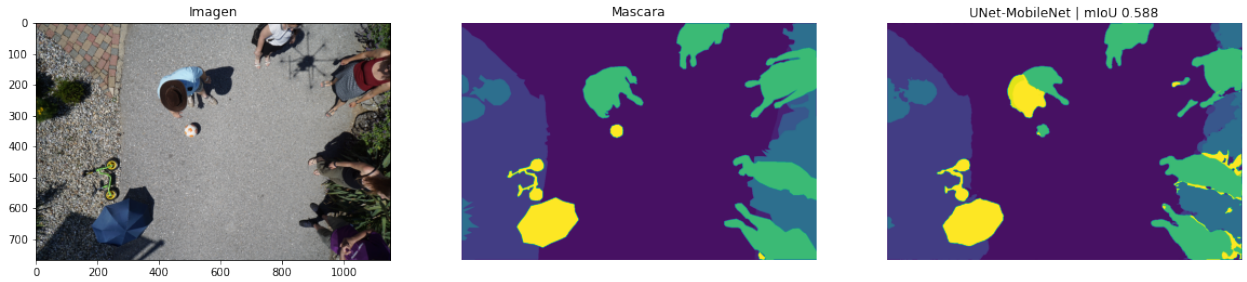


Figura 5.7: Resultado de segmentación semántica a una imagen del conjunto de prueba, se obtiene una media IoU 58.8 %, arquitectura U-Net. Elaboración propia.

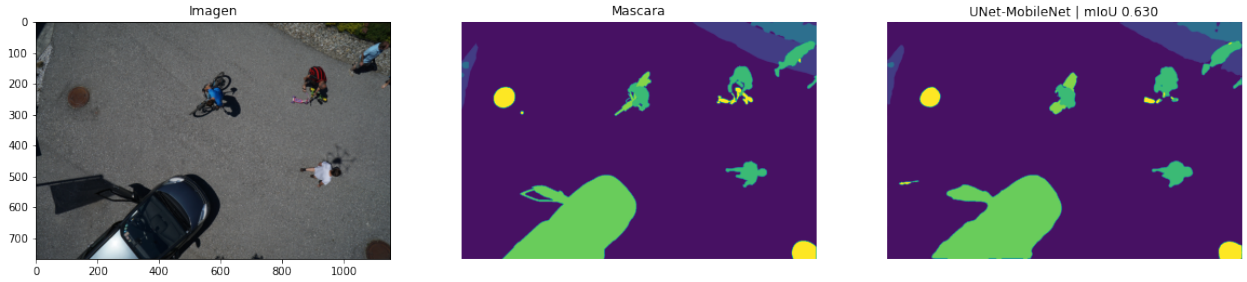


Figura 5.8: Resultado de segmentación semántica a una imagen del conjunto de prueba, se obtiene una media IoU 63 %, arquitectura U-Net. Elaboración propia.

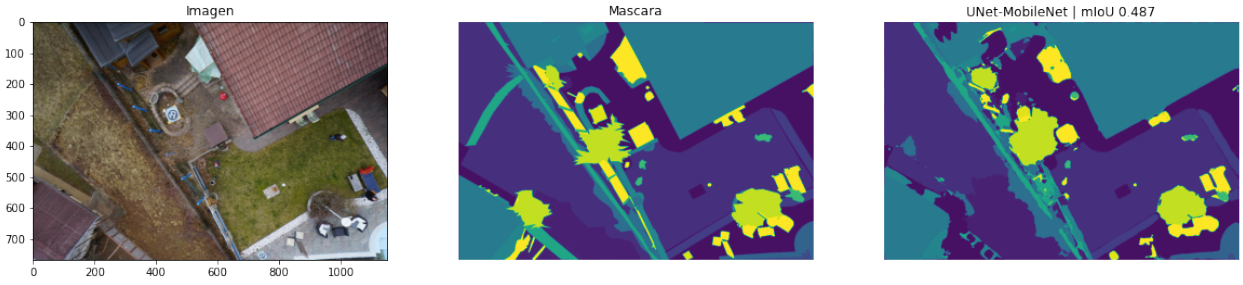
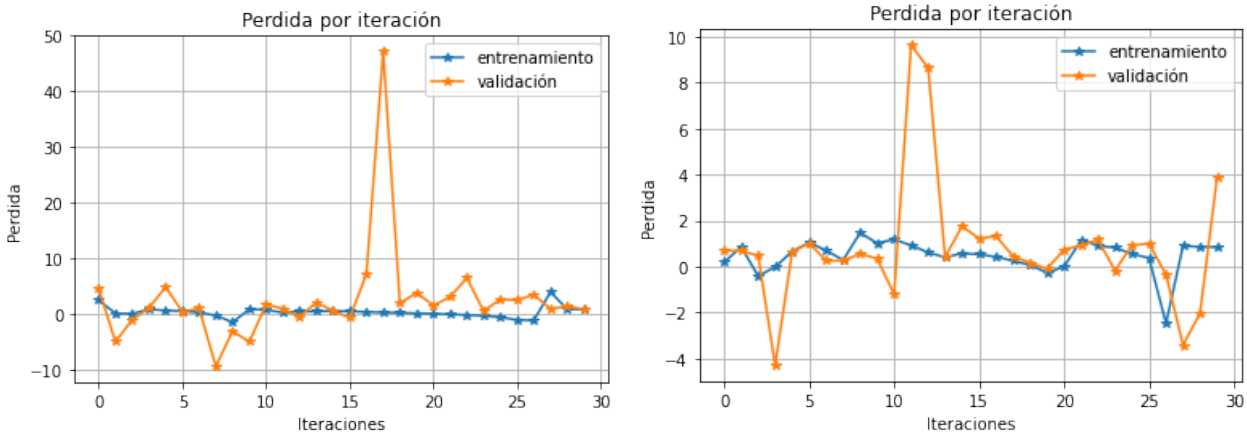


Figura 5.9: Resultado de segmentación semántica a una imagen del conjunto de prueba, se obtiene una media IoU 48.7 %, arquitectura U-Net. Elaboración propia.

5.2. Imágenes aéreas captadas por satélite

Función Perdida

Estas gráficas de pérdida resultan variantes por cada iteración en la validación. DeepLab provoca mayores pérdidas en ciertos puntos que U-Net, el máximo de pérdida que provoca DeepLab es de casi 50 y U-Net aproximadamente 10, lo extraño es que provocan pérdidas negativas. La probable razón de ocurrir esta pérdida negativa es por la cantidad de etiquetas en las máscaras, las arquitecturas aprendieron a clasificar las múltiples clases, como hay imágenes que posee clases múltiples también se encuentran máscaras que solo cuentan con una etiqueta, el modelo clasifica estas imágenes con las múltiples clases que contiene la imagen, pero resulta que la máscara solo contenía una clase, por lo que es mal provocando pérdidas negativas.



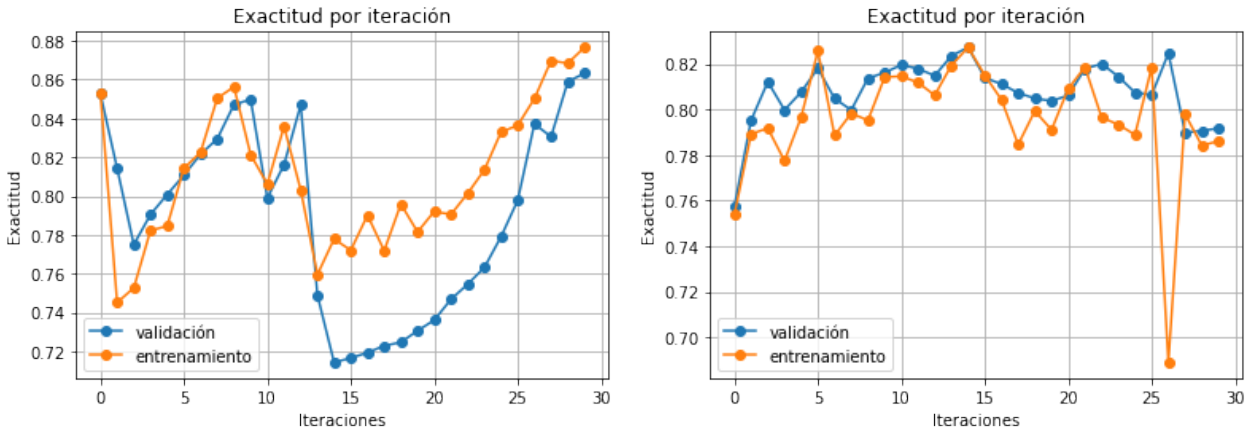
(a) Arquitectura de segmentación DeepLab.

(b) Arquitectura de segmentación U-Net.

Figura 5.10: Desempeño por la función de perdida de coeficiente de las arquitecturas de segmentación, 30 iteraciones. Elaboración propia.

Exactitud

La exactitud de DeepLab comparado con la de U-Net resulta ser más complicados pues, posee varias caídas y como se observa en la Figura 5.10a en la iteración 5 la pérdida es cercana a cero y luego esta bajó hasta llegar a una pérdida de -10, pero al observar la Figura 5.11a la exactitud aumenta cuando disminuye la pérdida a un valor negativo, en cambio cuando la exactitud disminuye en la iteración 25, la pérdida aumenta a casi 50 (iteración entre 15 y 20). En U-Net luego de la iteración 25 sucede una pérdida negativa alcanzando casi -3 en la validación, pero este tampoco se ve reflejado en la exactitud (ver Figura 5.11). Esta dispersión en las iteraciones con el entrenamiento y validación provoca confusión, ya que la pérdida influye en estas métricas y se complica aún más con los valores negativos.



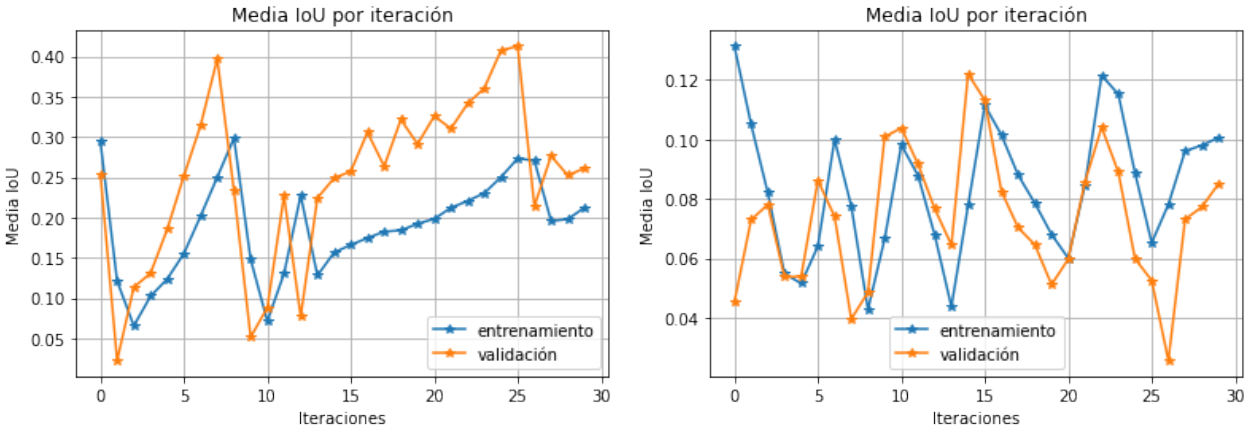
(a) Arquitectura de segmentación DeepLab.

(b) Arquitectura de segmentación U-Net.

Figura 5.11: Desempeño de la métrica exactitud entre $[0, 1]$ de las arquitecturas de segmentación, 30 iteraciones. Elaboración propia.

Intersección sobre Unión (IoU)

DeepLab varia más que U-Net en cuanto a la media IoU, gracias a ello la validación alcanza a tomar valores de un 0.4, mientras que U-Net solo en la primera iteración obtiene su resultado más elevado (ver Figura 5.12).



(a) Arquitectura de segmentación DeepLab.

(b) Arquitectura de segmentación U-Net.

Figura 5.12: Desempeño de la métrica IoU entre $[0, 1]$ de las arquitecturas de segmentación, 30 iteraciones. Elaboración propia.

5.2.1. Segmentación semántica del conjunto de prueba

DeepLab posee mejores resultados que U-Net, pero a pesar de ello no son satisfactorios si se quiere implementar, Se observa (ver Cuadro 5.2) que la exactitud es un valor realmente alto, no representa ser una métrica confiable, porque su resultado es engañoso por ser tan elevado a diferencia de las otras métricas.

Metrica	DeepLab	U-Net
Exactitud	**0.877**	0.786
IoU	**0.261**	0.085
Exhaustividad	**0.301**	0.136
Precisión	**0.523**	0.191
Valor-F	**0.350**	0.151

Cuadro 5.2: Resultados de la clasificación de arquitecturas de segmentación con el conjunto de prueba. Elaboración propia.

5.2.2. Ejemplos de segmentación semántica, conjunto de prueba

Generalizando los resultados de ambas arquitecturas de segmentación con el conjunto de imágenes de prueba (satélites) se observa una diferencia enorme en la clasificación con respecto a la máscara, porque estas arquitecturas segmentan con clases que no son parte de la imagen, analizando las Figuras 5.13, 5.14 y 5.15 se puede asumir que ambas arquitecturas realizan un buen

desempeño, pero las máscaras poseen un diseño que “anula” otras clases presentes en la imagen, los modelos reconocen estas clases y las segmenta disminuyendo su desempeño.

Segmentación semántica por DeepLab

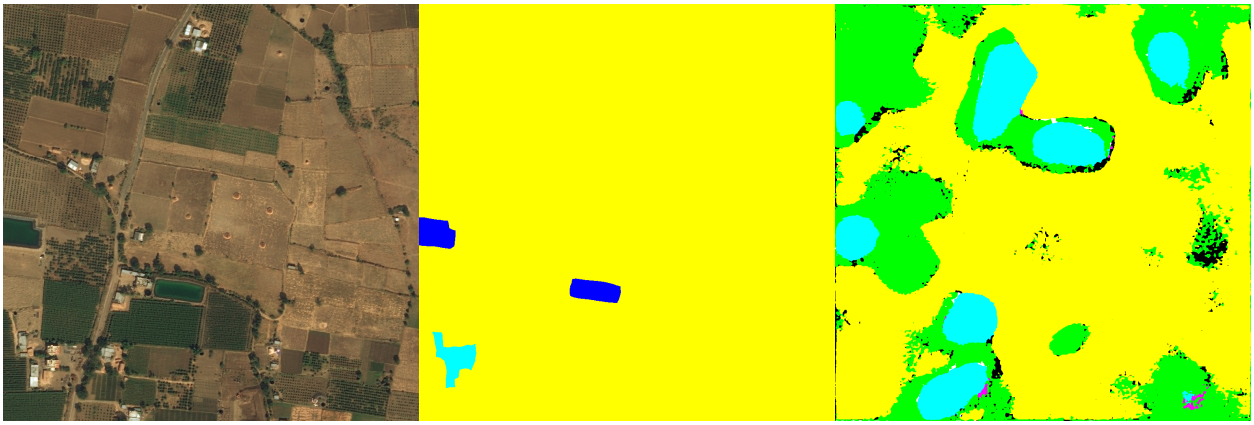


Figura 5.13: Resultado de segmentación semántica al conjunto de prueba imágenes por satélites, arquitectura DeepLab. Elaboración propia.

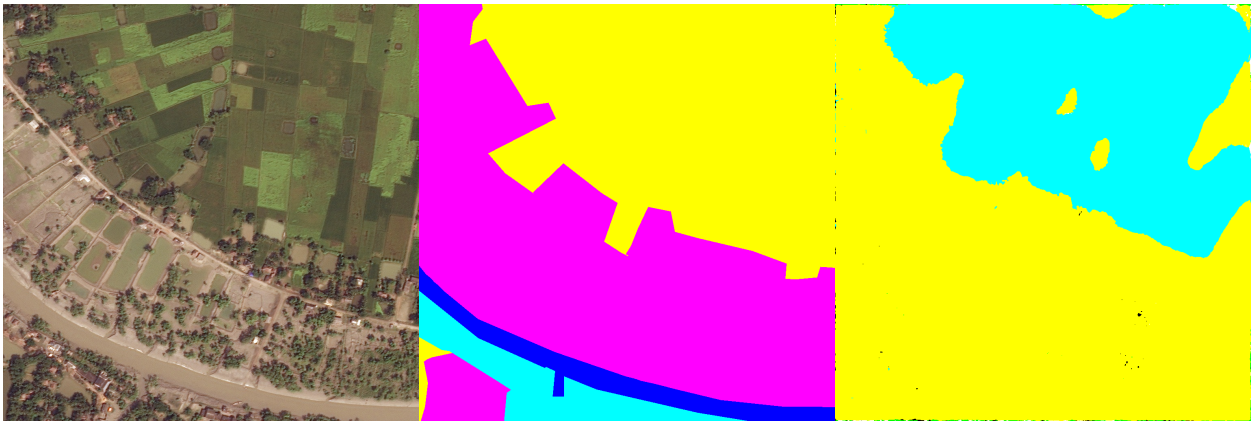


Figura 5.14: Resultado de segmentación semántica al conjunto de prueba imágenes por satélites, arquitectura DeepLab. Elaboración propia.

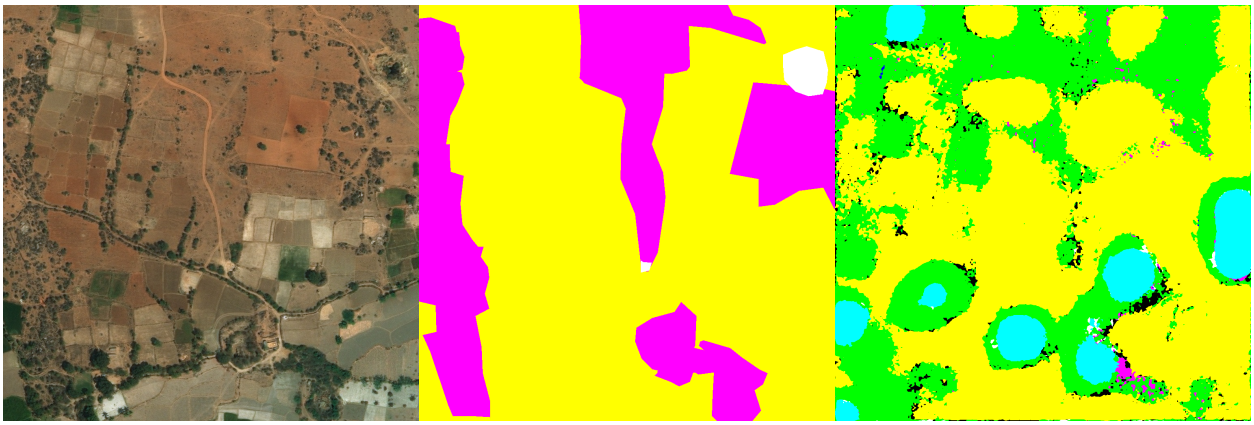


Figura 5.15: Resultado de segmentación semántica al conjunto de prueba imágenes por satélites, arquitectura DeepLab. Elaboración propia.

Segmentación semántica por U-Net

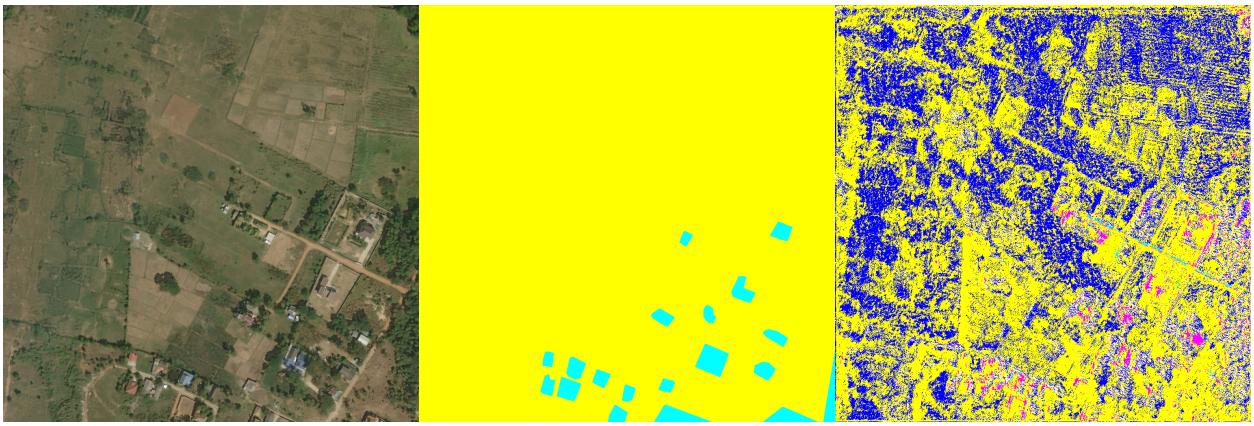


Figura 5.16: Resultado de segmentación semántica al conjunto de prueba imágenes por satélites, arquitectura U-Net. Elaboración propia.

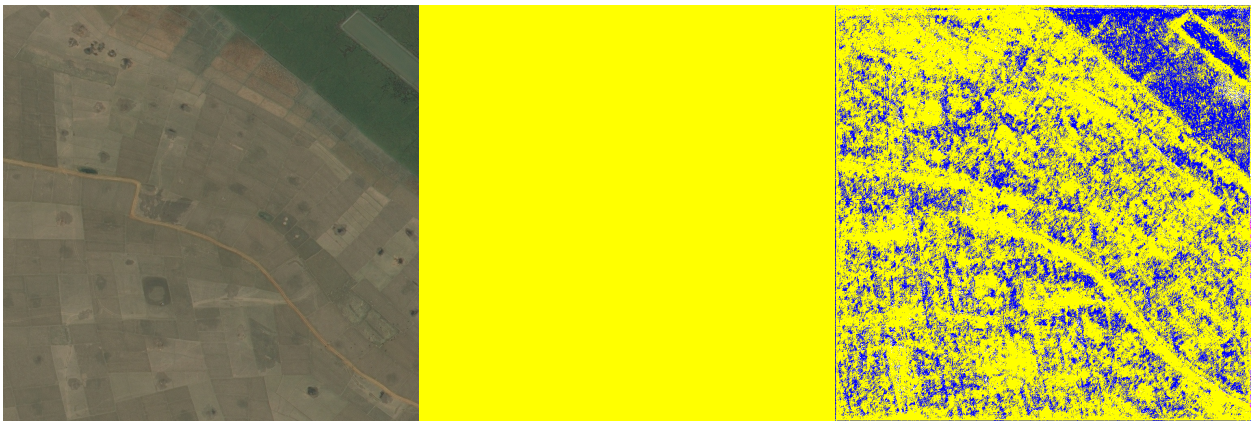


Figura 5.17: Resultado de segmentación semántica al conjunto de prueba imágenes por satélites, arquitectura U-Net. Elaboración propia.

Figura 5.18: Resultado de segmentación semántica al conjunto de prueba imágenes por satélites, arquitectura U-Net. Elaboración propia.

Capítulo 6

Conclusiones

En el desarrollo de este trabajo se analizaran las posibilidades que entrega el desarrollo de modelo del segmentación, pero se debe de contar no solo con conocimientos de las arquitecturas, también con metodologías como la función de activación, tipos de convoluciones y funciones de pérdida, porque poseer este conocimiento ayuda a preparar un modelo para un conjunto de datos, y además se deben adaptar técnicas para el desarrollo de máscaras y generar conjuntos de imágenes propios.

El objetivo principal de este trabajo fue evaluar dos metodologías del *deep learning* para segmentar imágenes aéreas. Se logró implementar y evaluar el desempeño de las arquitecturas de segmentación de U-Net y DeepLab, pues ambos desempeños fueron satisfactorios tanto en drones como en satélites, pese a que no se reflejen en las métricas la clasificación vista en las Figuras resultan ser esperadas, ya que segmentan las clases etiquetadas.

La aplicación de las dos arquitecturas con las imágenes aéreas obtenidas por dron fueron realmente satisfactorias, U-Net logra destacarse por solo milésimas, por ello se puede considerar la velocidad que se toma en entrenar, ya que esta es significativa al momento de requerir resultados en un tiempo "acotado". DeepLab se destaca en ello gracias a los métodos de "convolución atroz y pirámide de agrupación" logra obtener una mayor información de características y generalizar en un menor tiempo que U-Net, aun que ambas arquitecturas fueron entrenadas con 100 iteraciones, DeepLab solo necesita la mitad de esta, ya que en ese punto comenzó a estabilizarse debido a no poder aprender nuevos patrones, en cambio U-Net inicio su estabilidad a partir de la iteración 80.

En la aplicación de las imágenes aéreas obtenidas por satélite, ambas arquitecturas tuvieron un mal comportamiento, DeepLab tuvo un mejor desempeño que U-Net, pero aun así no alcanzó un desempeño satisfactorio. Este desempeño se puede mejorar en ambos, porque a pesar que el conjunto de imágenes poseía 8 clases, las imágenes no sobrepasaban las 3 clases y esto influye en la función de activación que se utiliza para la clasificación, en este trabajo se empleó la Softmax que es reconocida por la clasificación multiclase, por ello que las arquitecturas al segmentar las imágenes clasificaban con etiquetas que no pertenecen a la máscara, pero si a la imagen original, entonces para mejorar el desempeño es necesario utilizar una función de activación binomial por la cantidad de etiquetas que poseen las máscaras, no solo hay guiándose por la cantidad total de clases que posea un conjunto, también hay que revisar la "media" de clases que posean las máscaras para identificar una función de activación que se adapte a la cantidad de clases.

Referencias

Ai, B. (2019, 11). *Intro a las redes neuronales convolucionales.* Descargado de https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8

Alvaro. (2020, 06). *¿Qué es el sobreajuste u overfitting y por qué debemos evitarlo?* Descargado de https://machinelearningparatodos.com/que-es-el-sobreajuste-u-overfitting-y-por-que-debemos-evitarlo/

ArcGIS Developers. (s.f.). *How U-Net works? ArcGIS for Developers.* Descargado 2021-05-17, de https://developers.arcgis.com/python/guide/how-unet-works/

Bagnato, N. (2020, 06). *Convolutional Neural Networks: La Teoría explicada en Español.* Descargado de https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

Barrera, A., Guindel, C., García, F., y Martín, D. (2018). Análisis, evaluación e implementación de algoritmos de segmentación semántica para su aplicación en vehículos inteligentes. *Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de Septiembre de 2018*.

Barrios, J. (2019, 06). *Redes neuronales convolucionales son un tipo de redes neuronales.* Descargado de https://www.juanbarrios.com/redes-neurales-convolucionales/

Basavarajaiah, M. (2019, 08). *Maxpooling vs minpooling vs average pooling.* Descargado de https://medium.com/@bdhuma/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9

Berrondo Urruzola, A. (2020). Detección de carreteras en imágenes de reconocimiento remoto mediante deep learning.

Calvo, D. (2018, 12). *Backpropagation – Redes neuronales.* Descargado de https://www.diegocalvo.es/backpropagation-redes-neuronales/

Celdrán, H. (2012, 01). *El ser humano se perfecciona copiando a la naturaleza.* Descargado de https://www.20minutos.es/noticia/1275991/0/ser-humano/perfecciona/copia-naturaleza/

Chauhan, N. S. (2020, 09). *Métricas De Evaluación De Modelos En El Aprendizaje Automático.* Descargado de https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., y Yuille, A. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, *40*(4), 834–848.

Chicchón, M. (2018). Fusión de datos para segmentación semántica en aplicaciones urbanas de teledetección aérea usando algoritmos de aprendizaje profundo.

Contaval. (2016, 02). *¿Qué es la visión artificial y para qué sirve?* Descargado de https://www.contaval.es/que-es-la-vision-artificial-y-para-que-sirve/

Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., ... Raskar, R. (2018, June). Deepglobe 2018: A challenge to parse the earth through satellite images. En *The ieee conference on computer vision and pattern recognition (c.v.p.r.) workshops.*

Díaz Bou, P. (2021). *Desarrollo de soporte de redes recurrentes en plataforma de entrenamiento* (Tesis Doctoral no publicada). Universitat Politècnica de València.

Durán, J. (2017). Redes neuronales convolucionales en r: Reconocimiento de caracteres escritos a mano.

Durán, J. (2019, 10). *Técnicas de Regularización Básicas para Redes Neuronales.* Descargado de https://medium.com/metadatos/t%C3%A9cnicas-de-regularizaci%C3%B3n-b%C3%A1sicas-para-redes-neuronales-b48f396924d4

Dwivedi, P. (2019, 03). *Semantic Segmentation — Popular Architectures - Towards Data Science.* Descargado de https://towardsdatascience.com/semantic-segmentation-popular-architectures-dff0a75f39d0

Fei-Fei, L., Deng, J., y Li, K. (2009). Imagenet: Constructing a large-scale image database. *Journal of vision*, *9*(8), 1037–1037.

García Higuera, R., y cols. (2020). Estudio de utilización de redes neuronales convolucionales en tensorflow para segmentación de imagen médica.

Gavilán, I. (2020, 05). *Catálogo de componentes de redes neuronales (III): funciones de pérdida.* Descargado de https://ignaciogavilan.com/catalogo-de-componentes-de-redes-neuronales-iii-funciones-de-perdida/

Hebb, D. (1949). *0. the organization of behavior.* New York: Wiley.

Heras, J. M. (2020, 10). *Precision, Recall, F1, Accuracy en clasificación.* Descargado de https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/

ICG - DroneDataset. (s.f.). Descargado 2019-05-17, de https://www.tugraz.at/index.php?id=22387

Jiang, Z., y Shekhar, S. (2017). *Spatial big data science.* Springer.

Koech, K. E. (2021, 11). *Cross-Entropy Loss Function - Towards Data Science.* Descargado de https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e

Krizhevsky, A., Sutskever, I., y Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. En *Proceedings of the 25th international conference on neural information processing systems - volume 1* (p. 1097–1105). Red Hook, NY, USA: Curran Associates Inc.

Ku, E. (2020a, 01). *ML In Detail 1: PSPNet - Analytics Vidhya.* Descargado de https://medium.com/analytics-vidhya/ml-in-detail-1-pspnet-4527036af33b

Ku, E. (2020b, 03). *ML In Detail 2: DeepLab V3 (1) - Analytics Vidhya.* Descargado de https://medium.com/analytics-vidhya/ml-in-detail-2-deeplab-v3-1-ef6d5f5b126a

Ku, E. (2020c, 01). *PSPNet - Analytics Vidhya.* Descargado de https://medium.com/analytics-vidhya/ml-in-detail-1-pspnet-4527036af33b

Lamba, H. (2019, 02). *Understanding Semantic Segmentation with UNET - Towards Data Science.* Descargado de https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

López, R. F., y Fernandez, J. M. (2008). *Las redes neuronales artificiales.* Netbiblo.

Loss Function Library - Keras PyTorch. (2021, 07). Descargado de https://www.kaggle.com/bigironsphere/loss-function-library-keras-pytorch#Loss-Function-Reference-for-Keras-&-PyTorch

Ordaz, O. P., y Flores, D. M. (2019). Segmentación semántica para reconocimiento de escenas. *FINGUACH. Revista de Investigación Científica de la Facultad de Ingeniería de la Universidad Autónoma de Chihuahua*, *6*(19), 6–7.

Pal, S. (2020, 05). *Semantic Segmentation Tutorial — Semantic Segmentation Model.* Descargado de https://www.analyticsvidhya.com/blog/2019/02/tutorial-semantic-segmentation-google-deeplab/

Palomino, N., y Concha, U. (2009). Técnicas de segmentación en procesamiento digital de imágenes. *Revista de investigación de Sistemas e Informática*, *6*(2), 9–16.

Programador clic. (2018, 07). *Estandarización y normalización de imagen.* Descargado de https://programmerclick.com/article/8785332947/

Ronneberger, O., Fischer, P., y Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. En *International conference on medical image computing and computer-assisted intervention* (pp. 234–241).

Rubiales, A. (2020, 10). *Explicación de las Funciones de activación en Redes Neuronales y práctica con Python.* Descargado de https://rubialesalberto.medium.com/explicaci%C3%B3n-funciones-de-activaci%C3%B3n-y-pr%C3%A1ctica-con-python-5807085c6ed3

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., y Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. En *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4510–4520).

Shotton, J., Johnson, M., y Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. En *2008 ieee conference on computer vision and pattern recognition* (pp. 1–8).

Siddique, N., Sidike, P., Elkin, C., y Devabhaktuni, V. (2020). U-net and its variants for medical image segmentation: theory and applications. *arXiv preprint arXiv:2011.01118*.

Subramanyam, V. (2021, 01). *IOU (Intersection over Union) - Analytics Vidhya.* Descargado de https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef

Systems. (2020, 01). *Redes neuronales o el arte de imitar el cerebro humano.* Descargado de https://magiquo.com/redes-neuronales-o-el-arte-de-imitar-el-cerebro-humano/

Temp. (2020, 12). *Unsampling: Unpooling and Transpose Convolution.* Descargado de https://medium.com/jun-devpblog/dl-12-unsampling-unpooling-and-transpose-convolution-831dc53687ce

Tipos de núcleos de convolución: simplificados. (2020a, 12). Descargado de https://ichi.pro/es/tipos-de-nucleos-de-convolucion-simplificados-264161077525559

Tipos de núcleos de convolución: simplificados. (2020b, 12). Descargado de https://ichi.pro/es/tipos-de-nucleos-de-convolucion-simplificados-264161077525559

Verma, S. (2021, 10). *Understanding 1D and 3D Convolution Neural Network — Keras.* Descargado de https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610

Xiang, L. (2019, 01). *Filters in image processing (convolution kernels).* Descargado de https://www.cnblogs.com/Liu-xiang/p/10259861.html

Zhang, H., Dana, K., Shi, J., Zhang, Z., Wang, X., Tyagi, A., y Agrawal, A. (2018). Context encoding for semantic segmentation. En *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 7151–7160).