Universitat de les Illes Balears

Escola Politècnica Superior

Memoria del Trabajo de Fin de Grado

Segmentación semántica multiclase de imágenes submarinas utilizando redes neuronales profundas

Iván Riutort Ozcariz

Grado de Ingeniería electrónica industrial y automática

Año académico 2018-19

DNI del alumno: 43119079J

Trabajo tutelado por Yolanda González Cid
Departamento de Ciencias Matemáticas e Informática

Se autoriza a la Universidad incluir este trabajo en el Repositorio Institucional para su consulta en acceso abierto y difusión en línea, con finalidades exclusivamente académicas y de investigación	Autor		Tutor	
	Sí	No	Sí	No
	X		X	

Palabras clave del trabajo:
Aprendizaje profundo, segmentación semántica, robótica submarina, Posidonia oceanica

Sumario

Índice de tablas

Índice de ilustraciones

Acrónimos

Po: *Posidonia oceanica*

CO2: Dióxido de carbono

IA: Inteligencia Artificial

SONAR: Navegación por sonido (*Sound Navigation And Ranging*)

AUV: Vehículo autónomo submarino (*Autonomous Underwater Vehicle*)

TFG: Trabajo de Fin de Grado

ML: Aprendizaje automático (*Machine learning*)

DL: Aprendizaje profundo (Deep Learning)

RGB: código de color rojo, verde y azul (*Red, Green, Blue*)

TP: Verdadero positivo (*True Positive*)

FP: Falso positivo (*False Positive*)

TN: Verdadero negativo (*True Negative*)

FN: Falso negativo (*False Negative*)

UIB: Universitat de les Illes Balears

1 Resumen

La *Posidonia oceanica* es una planta endémica del mar Mediterráneo. Los sucesivos estudios que se realizan manifiestan un descenso significativo de su presencia en este mar. La monitorización y mapeo de los fondos marinos es imprescindible para poder realizar un seguimiento de la evolución de la Posidonia y así poder realizar acciones para intentar revertir esta situación.

En este *Trabajo de Fin de Grado* (TFG) se ha implementado una red neuronal profunda (*Deep Neural Network*) para realizar una segmentación semántica de imágenes del fondo marino ofreciendo mejoras respecto a las técnicas utilizadas actualmente para obtener esta información.

La red neuronal desarrollada permite distinguir en las imágenes del fondo marino las zonas de Posidonia, roca y arena. También se ha explorado en este trabajo la posibilidad de identificar otros elementos presentes como la Posidonia muerta, tanto la propia planta como sus hojas sueltas dispersas por el fondo.

2 Agradecimientos

Querría agradecer el apoyo de Miguel Martin Abadal para la realización de este trabajo. A parte de la inestimable ayuda durante la realización de este trabajo, también ha contribuido significativamente en la redacción y presentación de un artículo en el congreso internacional OCEANS 2019 (https://www.oceans19mtsieeemarseille.org/) basado en los resultados de este trabajo. Esto ha permitido que este trabajo tenga una difusión mayor de lo que yo podría esperar.

3 Introducción

La *Posidonia oceanica* es una planta acuática, endémica del mediterráneo y de gran valor ecológico. Protege la costa de la erosión, es la base del ecosistema de multitud de especies marinas, además oxigena el agua y capta el CO2, por lo que es un elemento que mitiga el cambio climático.

A pesar de ser especie protegida y de realizar acciones para su conservación, la Posidonia sigue disminuyendo debido a la pesca de arrastre, las anclas de las embarcaciones, la contaminación, el cambio climático, etc [1].

Para poder realizar un control y recuperación es necesario poder hacer un seguimiento de la evolución de las praderas de Posidonia. Esta monitorización actualmente se realiza mediante:

- Buzos: por lo que es una tarea muy lenta y costosa.
- Imágenes de satélite: problemas de detección en aguas profundas.
- SONAR: utilizado habitualmente para realizar mapas batimétricos.
- *Vehículo Autónomo Submarino* (AUV) provisto de sensores.

El Trabajo de Fin de Grado realizado identifica el tipo de fondo marino a partir de imágenes submarinas. Esta identificación consiste en indicar en cada píxel de la imagen qué es, en nuestro caso se indicará en cada píxel si es Po, arena, roca, etc. A esta manera de clasificar las imágenes se denomina segmentación semántica multiclase.

Para conseguir esta segmentación semántica se utilizará *Inteligencia Artificial* (IA), concretamente una red neuronal profunda .

Gracias a este TFG se pretende automatizar la detección de la Po en un AUV, que realizará mapas del fondo marino con los que poder realizar el seguimiento de la Po.

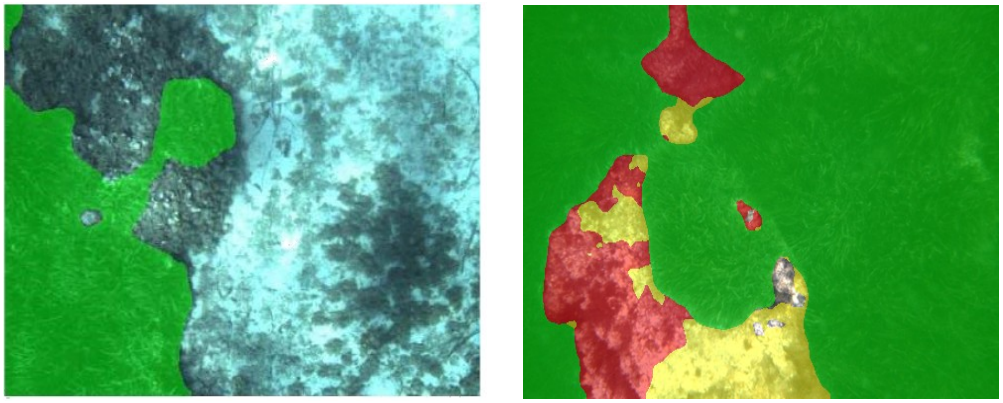
Este trabajo es una evolución de un trabajo realizado previamente por el grupo de investigación Sistemas, Robótica y Visión [2]. La mejora consiste en realizar la segmentación semántica multiclase, es decir, el anterior trabajo sólo identifica en la imagen si en una zona hay Po o no, con este trabajo se consigue identificar además de la Po otro tipos de elementos como si el suelo es de arena o roca.

Los resultados de este trabajo han sido publicados y presentados en la conferencia OCEANS 2019 [3].

En la ilustración 1 se puede ver un ejemplo se segmentación semántica monoclase y otro de multiclase, donde se ha coloreado sobre la imagen original para identificar los distintos tipos de fondo.

En la imagen 1(a) se muestra la clasificación monoclase del trabajo [2], donde el color verde representa zonas de Po y el resto de la imagen sin colorear indica que no es Po.

En la imagen 1(b) es el caso de segmentación semántica multiclase realizada en este TFG. Donde además del color verde para la Po, se colorea en rojo la roca, en amarillo la arena y lo que no se ha sabido identificar no se ha coloreado.



(a) *(b)*

Ilustración 1: Comparación segmentación semántica monoclase (a) y multiclase (b)

El poder identificar además de la Po el tipo de suelo puede ayudar a ver la evolución del crecimiento o regresión de la Po dependiendo del terreno. Así además de poder ayudar a tomar acciones en determinadas zonas también sirve para realizar un seguimiento del resultado de las mismas.

3.1 Objetivo

La preocupación por la conservación de la *Posidonia oceanica (Po)* implica el poder realizar un seguimiento de su evolución y así poder determinar si su presencia está aumentando o disminuyendo en nuestros fondos marinos.

Hoy en día para realizar este seguimiento se utilizan métodos costosos y con limitaciones, que muchas veces también implica una intervención manual.

El Trabajo Final de Grado realizado pretende determinar de manera automática el tipo de fondo marino, a partir de imágenes submarinas, realizando una segmentación semántica gracias a redes neuronales profundas (*Deep Neural Networks*).

4 Conceptos generales

Este TFG hace referencia a conceptos técnicos tanto del campo de la Inteligencia artificial como del procesamiento de imágenes, en este apartado se realiza una introducción a estos conceptos como Deep Learning y convolución entre otros.

4.1 Inteligencia artificial para procesamiento de imágenes

La IA es dotar a una máquina de la capacidad de resolver problemas imitando el comportamiento humano. En el caso concreto de procesamiento de imágenes este comportamiento consiste en poder reconocer patrones de la imagen para poder interpretar qué es lo que se muestra en la misma.

Dentro de la IA artificial hay muchas ramas, es este TFG se utilizan técnicas de Deep Learning, en la ilustración 2 se muestra como está clasificado el Deep Learning dentro de la IA.

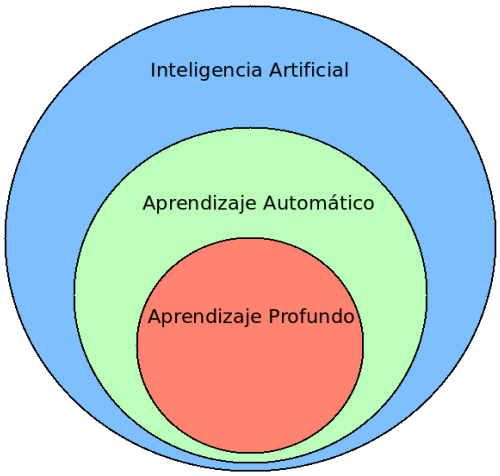


Ilustración 2: Clasificación de inteligencia artificial, aprendizaje automático y aprendizaje profundo

4.2 Aprendizaje automático (ML)

El ML es un subconjunto de la IA que consiste en técnicas para que las máquinas aprendan a resolver los problemas. Dentro del ML existen dos categorías:

- ML supervisado: para que se pueda producir el aprendizaje es necesario proporcionar los resultados que se esperan obtener para unas determinadas entradas.
- ML no supervisado: para el aprendizaje no se conoce el resultado que tiene que dar la información de entrada.

Otro aspecto importante del ML es el entrenamiento, el entrenamiento es una fase en la que la máquina va aprendiendo a realizar la tarea encomendada. Este aprendizaje básicamente consiste en ir ajustando los parámetros de los algoritmos utilizados por la máquina.

Posterior a la fase aprendizaje está la fase de inferencia, en la que una vez se ha entrenado la máquina, realiza la tarea para la que está diseñada a partir de entradas nuevas.

En este trabajo se utiliza aprendizaje supervisado, por lo que se requiere una colección de imágenes que se usarán como entrada durante el entrenamiento y también se necesita otra colección correspondiente al resultado que se espera obtener por cada imagen de entrada. En la figura 3 se muestra el esquema utilizado, durante el entrenamiento se van ajustando los parámetros de la red para generar el modelo final y durante la inferencia se realiza una predicción a partir de imágenes nuevas [4][5].

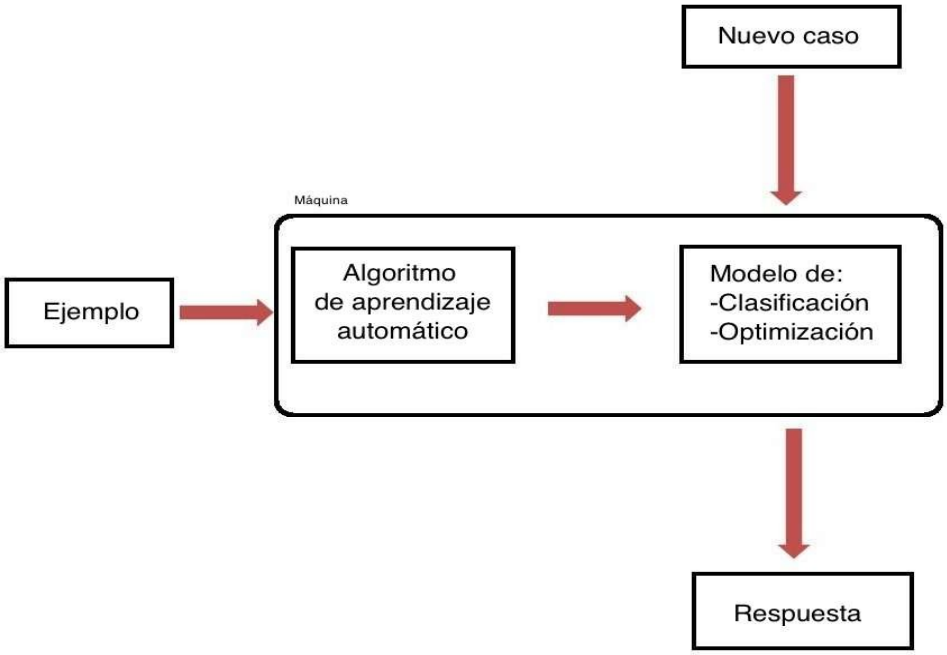


Ilustración 3: Esquema de aprendizaje automático

4.3 Aprendizaje profundo (DL)

Es un subtipo de ML en el que el modelo utilizado consta de varias capas en cascada, donde cada capa realiza transformaciones a la información de entrada hasta obtener la predicción del resultado final. En los últimos años se ha ido extendiendo la utilización de técnicas de DL en el ámbito de la visión por computador, sobre todo, gracias al incremento de la capacidad computacional de los nuevos procesadores, ya que los algoritmos en los que se basan se definieron hace más de dos décadas [4][6].

4.4 Redes Neuronales Artificiales

En IA las redes neuronales pretenden modelar el funcionamiento de las neuronas biológicas. La unidad básica en una red neuronal artifical es el perceptrón, el cual sería el equivalente a la neurona biológica. La ilustración 4 compara los esquemas de la neurona biológica (a) y del perceptrón (b).

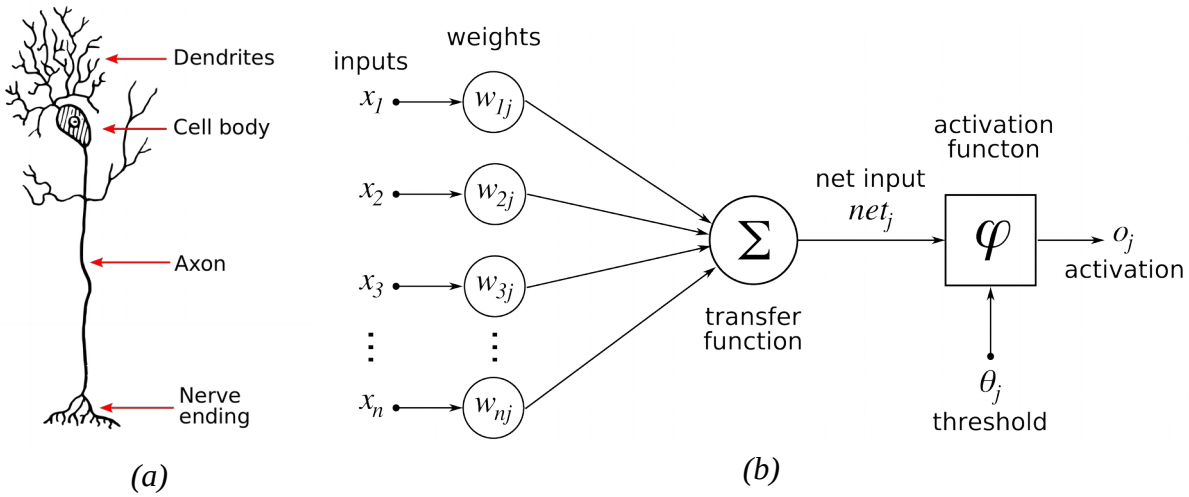


Ilustración 4: Esquema neurona (a) y perceptrón (b)

Una neurona capta las señales que le llegan por las dendritas y si estas señales son lo suficiente intensas provocan una respuesta de la neurona a través del axón.

Un perceptrón básicamente recibe multiples señales de entrada las cuales son multiplicadas por un peso y sumadas entre ellas. Si la suma de las entradas supera un umbral mínimo el perceptrón devolverá un valor de salida [6].

Una red neuronal artificial consta de la agrupación en capas e interconexión de multitud de estos perceptrones como se muestra en la ilustración 5.

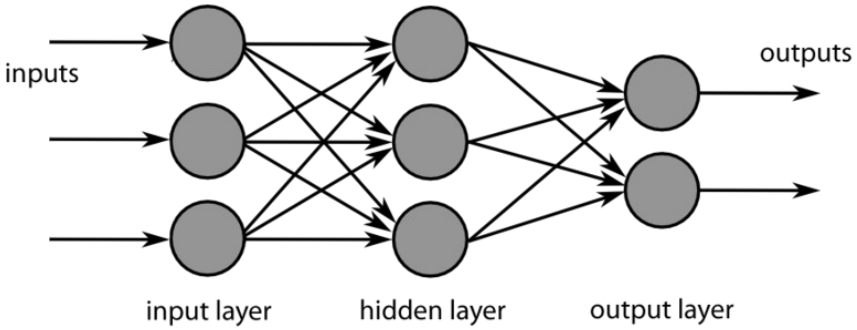


Ilustración 5: Esquema de red neuronal artificial

4.5 Aprendizaje profundo (DL)

Es un subtipo de ML en el que el modelo utilizado consta de varias capas en cascada, donde cada capa realiza transformaciones a la información de entrada hasta obtener la predicción del resultado final. En los últimos años se ha ido extendiendo la utilización de técnicas de DL en el ámbito de la visión por computador, sobre todo, gracias al incremento de la capacidad computacional de los nuevos procesadores, ya que los algoritmos en los que se basan se definieron hace más de dos décadas [4][6].

4.5.1 Red neuronal profunda

Son un tipo de redes neuronales artificiales donde la información de entrada es procesado por varias capas hasta obtener los resultados deseados. En la ilustración 6 se muestra un esquema básico de este tipo de redes.

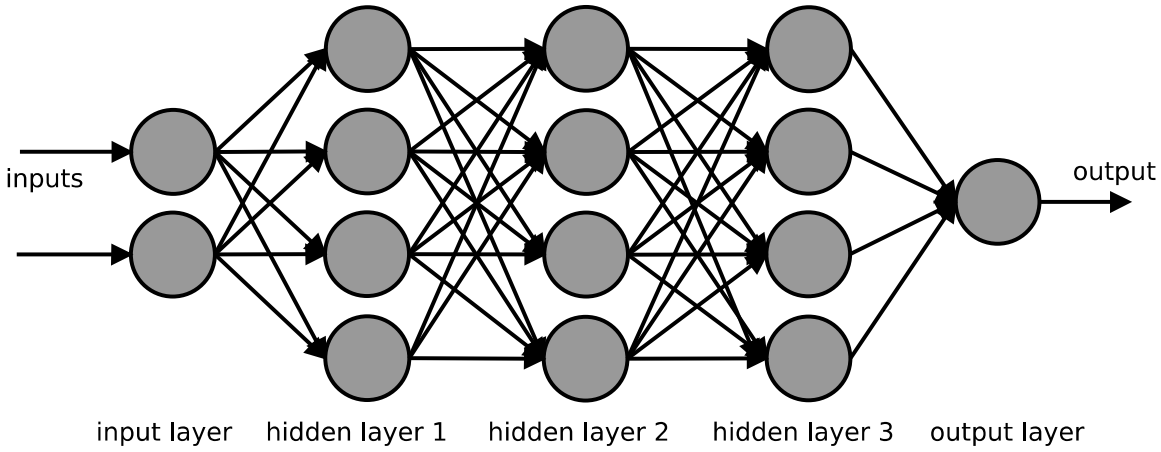


Ilustración 6: Esquema de red neuronal profunda

4.5.2 Red neuronal convolucional

Para el caso de tratamiento de imágenes se suele utilizar un tipo de red neuronal denominada convolucional. Estas redes consisten básicamente en varias capas que realizan lo siguiente:

- Convolución: es una técnica que suelen utilizar habitualmente programas de edición de imágenes para aplicar filtros. La convolución consiste en realizar una serie de operaciones sobre la imagen de entrada a partir de un filtro que denominamos kernel. Para un kernel de $n \times n$ el resultado de la convolución del píxel $Y_{x,y}$ se calcularía de la siguiente manera para un píxel que no sea de la frontera y para un kernel de dimensiones impares [4]:

$$Y_{x,y}(X) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} X_{x+i-\frac{n-1}{2}, j+i-\frac{n-1}{2}} \cdot K_{i,j}$$

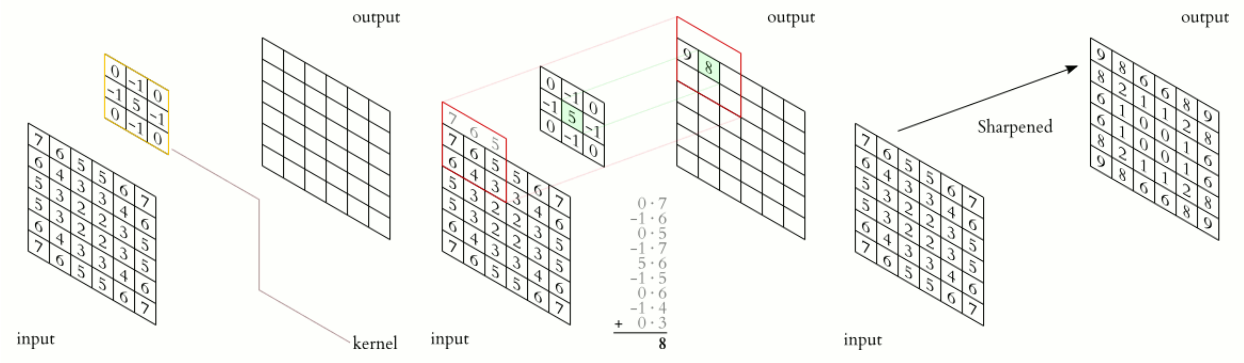
  donde:

  - X: es la matriz bidimensional con la imagen de entrada.

- Y: es la matriz bidimensional de salida.
- x,y: representan las coordenadas de un píxel de la imagen.
- K: matriz bidimensional que representa el kernel.

El resultado de una convolución en el tratamiento de imágenes se suele denominar mapa de características, ya que normalmente se aplican para resaltar características de la imagen, como por ejemplo, los límites de los objetos de la imagen.

La ilustración 7 muestra un ejemplo de como calcular la convolución de una matriz bidimensional. Se puede apreciar el inicio de la convolución con la matriz de entrada y el kernel que se utilizará (a). La imagen (b) es un ejemplo de multiplicación de una posición de la entrada con el kernel para calcular un píxel de salida. Finalmente en la imagen (c) se obtiene el resultado final de la convolución.



(a) *(b)* *(c)*

Ilustración 7: Ejemplo de convolución

- Pooling: consiste en realizar una reducción de la matriz obtenida con la convolución. Por ejemplo, quedándose para un grupo de píxeles únicamente con el píxel de valor mayor (Max Pooling) [4], como se muestra en la ilustración 8:

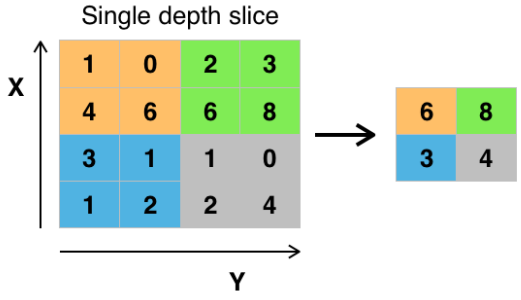


Ilustración 8: Ejemplo de Max Pooling

En la ilustración 9 se muestra un esquema de ejemplo de utilización de una red neuronal convolucional. Ilustra como a partir de una imagen de entrada se le aplican distintas transformaciones de convolución y pooling obteniendo mapas de características y finalmente el resultado deseado.

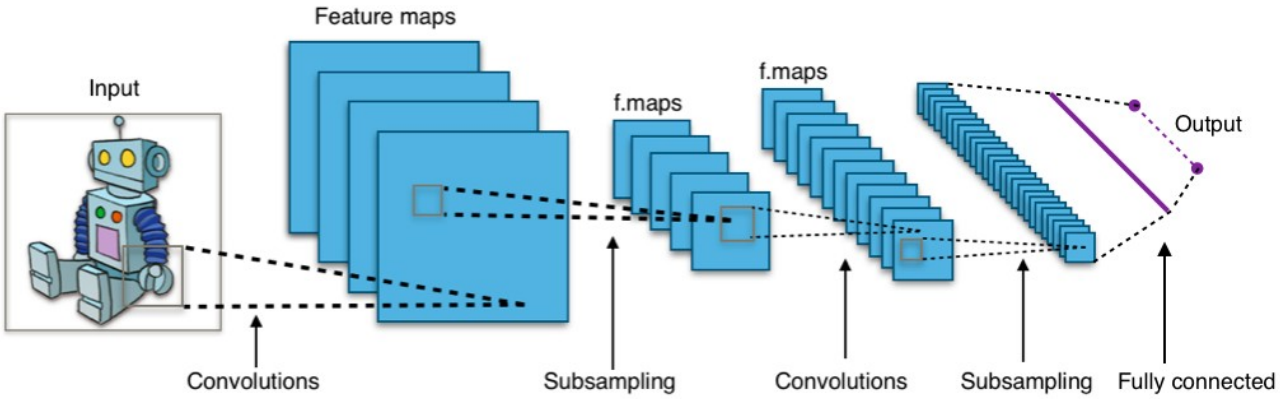
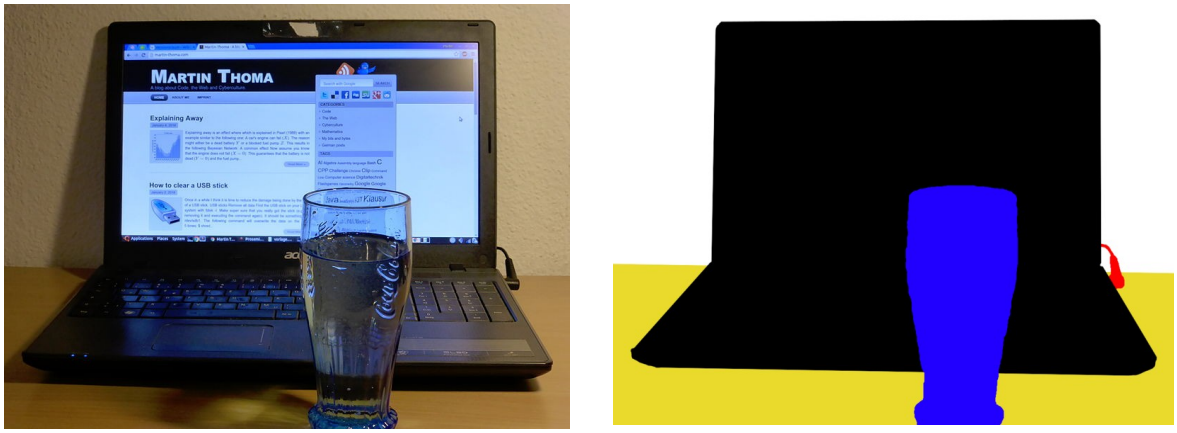


Ilustración 9: Esquema de ejemplo de red neuronal convolucional

4.6 Segmentación Semántica

La segmentación semántica consiste en identificar qué es cada píxel de una imagen. En el ejemplo de la ilustración 10 se muestra como se realiza esta segmentación semántica a partir de una imagen (a), donde a cada píxel de la imagen se le asigna un color dependiendo de lo que representa. De esta manera se clasifica por colores los elementos de la imagen. En el ejemplo de la ilustración 10 (b) el color azul sería un vaso, el negro un portátil, el rojo un cable, el amarillo una mesa y el blanco para el fondo.

A la clase de fondo también se le suele denominar background, y suele utilizarse para los elementos que no están tipificados en ningún color.



(a) *(b)*

Ilustración 10: Ejemplo de segmentación semántica multiclase

Dentro de la segmentación semántica se puede distinguir dos casos:

- Segmentación semántica monoclase: donde únicamente se identifica un elemento y el resto es background.
- Segmentación semántica multiclase: donde además del background hay más de un elemento para clasificar.

Para poder conseguir automáticamente esta segmentación semántica se suele recurrir a técnicas de DL mediante la utilización de redes neuronales convolucionales.

5 Desarrollo del trabajo fin de grado

En este proyecto se ha implantado la red neuronal necesaria para conseguir la segmentación semántica de las imágenes del fondo marino. Para conseguirlo se han utilizado las librerías Tensorflow y la programación se ha realizado en Python.

El código del TFG es una evolución de un trabajo para detectar Posidonia realizado por el grupo de investigación Sistemas, Robótica y Visión [2], que a su vez es una adaptación de otro trabajo para detectar carreteras [7].

Además de la parte de programación de la red neuronal se han tenido que preparar los datos para los entrenamientos y definir las métricas necesarias para validar los resultados y comprobar la calidad de los modelos obtenidos.

5.1 Colección de datos

Para este trabajo se han utilizado 302 imágenes obtenidas de grabaciones realizadas por cámaras montadas en un AUV. Las imágenes son del fondo marino de las Islas Baleares, sobre todo, de Mallorca. Estas imágenes están tomadas con distintas condiciones de luz y turbulencia del agua, esta variedad consigue que la red entrenada sea más versátil.

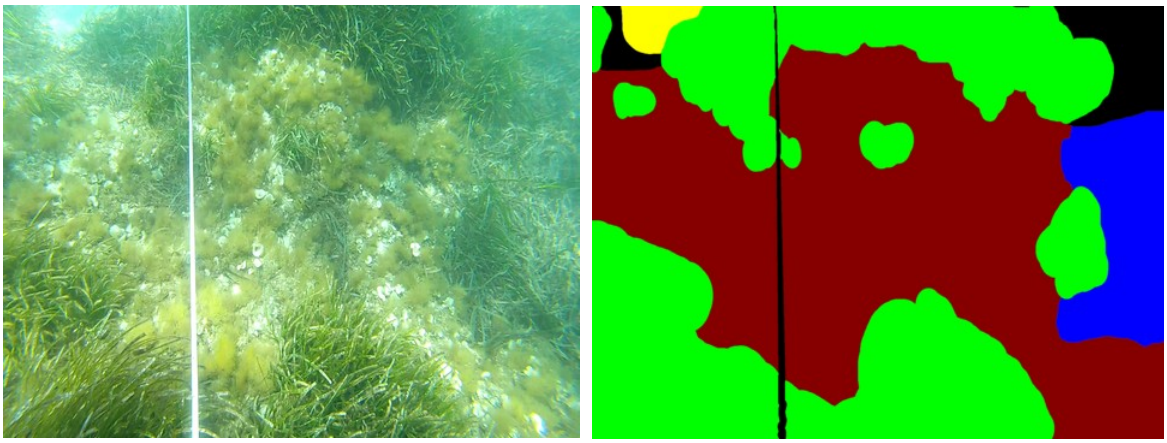
Aunque las imágenes originales pueden tener distintas dimensiones, antes de realizar el entrenamiento se redimensionan a 480x360 que es el tamaño que admite la red neuronal implementada.

Las imágenes se han dividido en dos colecciones:

- 242 imágenes para el entrenamiento (80%).
- 60 imágenes para los tests (20%).

También se ha tenido que realizar el etiquetado de todas las imágenes. Este etiquetado consiste en realizar de manera manual una segmentación semántica por cada imagen, de esta manera dispondremos de los resultados deseados que necesita la red neuronal para entrenarla y validarla.

Para realizar el etiquetado de una imagen hay que definir un color para cada clase que se quiere identificar. Una vez que se tienen definidas las clases hay que colorear cada píxel de la imagen original con su color correspondiente. En la ilustración 11 se muestra un ejemplo de como a partir de una imagen original (a) se edita para obtener una imagen etiquetada (b).



(a) *(b)*

Ilustración 11: Etiquetado de imágenes

Los colores que se han definido para cada clase que se quiere identificar en las imágenes submarinas se muestran en la tabla 1.

Nombre	Descripción	Color
Po-a	Mata de Po viva	
Po-d	Hojas de Po muerta	
Rock	Fondo rocoso limpio	
Rock-d	Fondo rocoso con algas u otros elementos	
Sand	Fondo de arena limpia	
Sand-d	Fondo de arena con algas u otros elementos	
Matte-s	Mata de Po muerta en fondo de arena	
Matte-r	Mata de Po muerta en fondo de roca	
Background	Fondo que no corresponde a ninguna de las clases que se quiere clasificar o que no se ha sabido identificar	

Tabla 1: Correspondencia de clases y colores

5.2 Arquitectura utilizada

Para implementar la segmentación multiclase me he basado en una red neuronal multicapa, en la que en la entrada de la misma es una imagen en color RGB a la que se le aplican varias transformaciones como se muestra en la ilustración 12 para conseguir la segmentación semántica multiclase [2].

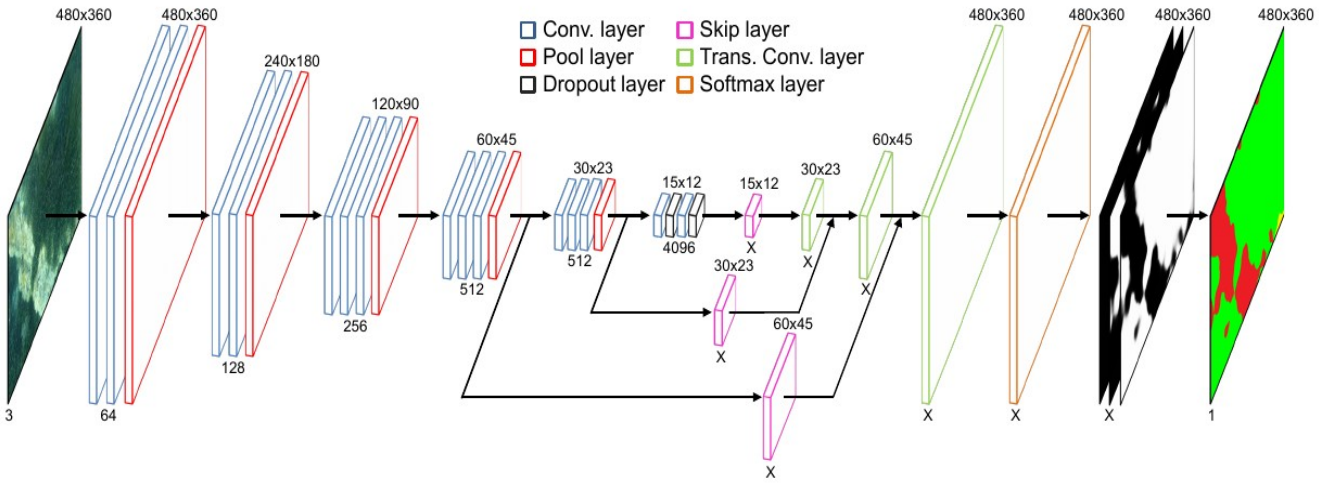


Ilustración 12: Arquitectura de capas de la red neuronal utilizada

En la ilustración 12, el número situado encima de las capas representa las dimensiones de los mapas de características que se utilizan en esa capa. El número de debajo de la capa representa la cantidad de mapas de características que tiene esa capa, en los casos en que se muestra una X corresponde al número de clases que se quieren identificar.

La estructura de capas se divide en dos fases:

- Encoder: en esta primera fase de codificación se extraen las características de la imagen mediante capas de convolución y pool.
- Decoder: la segunda fase de decodificación, se reconstruye la imagen mediante convolución traspuesta, de esta manera se obtiene la clasificación de cada píxel de la imagen.

5.2.1 Encoder

Las dimensiones de las imágenes a tratar serán de 480x360 píxeles en RGB. En la primera etapa se recibe la imagen y es procesada por la primera capa de convolución. Esta primera etapa consta de 2 capas de 64 convoluciones y una capa que realiza el Max_Pool para reducir el tamaño de las características a la mitad (240x180 píxeles).

La segunda etapa vuelve a constar de dos capas de convolución, aunque ahora se doblan el número de convoluciones. Al terminar de realizar las convoluciones se vuelve a aplicar un Max_Pool para reducir otra vez las dimensiones a la mitad (120x90 píxeles).

Las etapas tercera y cuarta se sigue el mismo procedimiento añadiendo una tercera capa de convolución, se doblan el número de convoluciones y se reduce las dimensiones a la mitad.

En la quinta etapa se mantienen el número de convoluciones y se reducen las dimensiones a la mitad.

En la sexta etapa además de las capas de convolución se incluyen unas capas de dropout que se utilizan únicamente durante el entrenamiento para evitar el *overfitting*. En esta etapa los mapas de características se han reducido a un tamaño de 15x12.

5.2.2 Decoder

El decodificador contiene un mapa por cada clase que se quiere clasificar. A medida que se avanzan las etapas del decodificador se aumenta el tamaño de los mapas hasta llegar a la dimensión de la imagen original.

La capa de skip (skip layer) básicamente consiste en una convolución a partir del mapa de características para obtener un nuevo mapa por clase.

En la séptima etapa consta de una capa de skip por lo que a partir de la sexta etapa se obtiene un mapa por clase.

La etapa octava consta de una convolución traspuesta que aumenta el tamaño de la imagen.

En la novena etapa se realiza la convolución traspuesta de la suma de las capas de la etapa anterior y la capa de skip de la quinta etapa volviendo a aumentar el tamaño de la imagen.

La décima etapa se realiza la convolución traspuesta de la suma de las capas de la etapa anterior y la capa de skip de la cuarta etapa, aumentando el tamaño al de la imagen original.

En la undécima etapa se aplica la función softmax para normalizar el resultado obtenido.

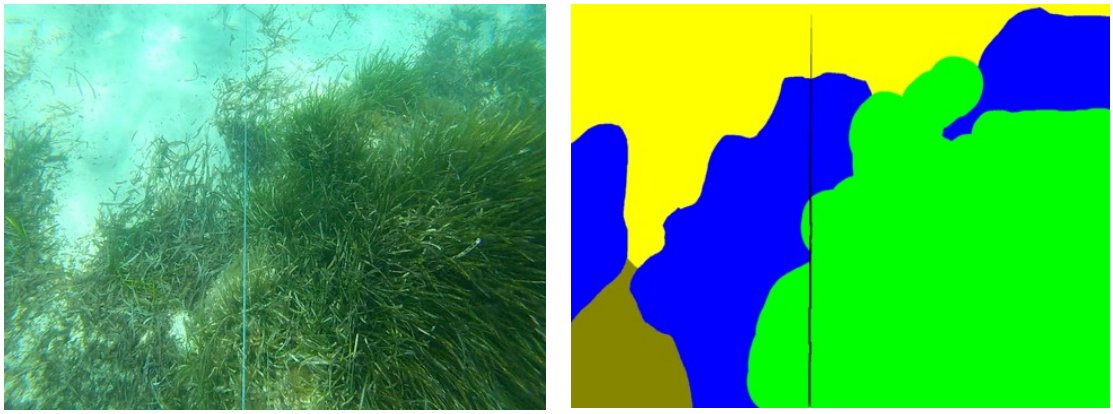
Finalmente en la última etapa obtenemos un mapa por clase, el valor de cada posición de este mapa representa la probabilidad que tiene ese píxel de ser de la clase del mapa. Por lo que para decidir de qué clase es cada píxel se escoge el de la clase con mayor valor.

Como a cada clase se le asigna un color al final se obtiene una imagen donde cada píxel está clasificado.

5.3 Entrenamiento

Para calcular los valores del modelo de la red neuronal se realizará una fase de entrenamiento supervisado. Este entrenamiento consiste en ir ajustando los parámetros del modelo a partir de una colección de imágenes con sus correspondientes imágenes etiquetadas.

En la ilustración 13 se aprecia el como a partir de una imagen (a) se ha realizado su etiquetado (b) para poder realizar el entrenamiento. Cada color representa una clase, el verde es Posidonia, el azul Posidonia muerta, el amarillo arena y el marrón arena con otros elementos.



(a) *(b)*

Ilustración 13: Etiquetado de imágenes para el entrenamiento

El entrenamiento consiste en procesar las imágenes y comparar el resultado obtenido con el resultado deseado utilizando una función de pérdida. Al aplicar la función de pérdida se obtiene el error del modelo que a partir del cual se ajustan los pesos del modelo. Este procedimiento se vuelve a realizar varias veces para ir ajustando progresivamente los pesos. Utilizando un algoritmo denominado back propagation, que gracias al gradiente de la función de pérdida, se calculan los pesos del modelo.

Para evitar el *overfitting* y mejorar el resultado del modelo obtenido se suelen añadir capas de dropout. El *overfitting* es un problema que puede aparecer en las redes neuronales, lo que provoca es que el modelo obtenido después del entrenamiento funciona correctamente para las imágenes utilizadas en el entrenamiento, pero no para las nuevas. Las capas de dropout lo que hacen es desactivar aleatoriamente algunas conexiones de la red neuronal. De esta manera se fuerza a que la red entrenada sea más versátil, gracias a que se fuerza la activación de más conexiones al identificar las características de la imagen [4][5].

En la ilustración 14 se muestra un esquema de cómo se entrena el modelo para una vez entrenado poder realizar la inferencia de nuevas imágenes.

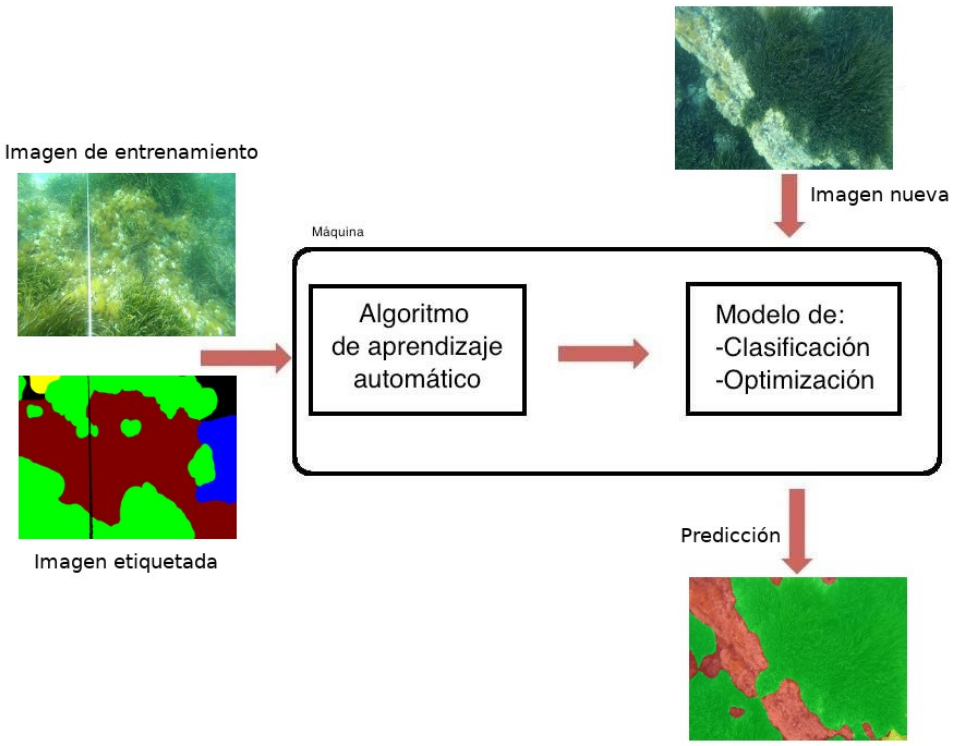


Ilustración 14: Esquema de entrenamiento del modelo

5.3.1 Hiperparámetros utilizados en el entrenamiento

Hay una serie de parámetros que se utilizan durante el entrenamiento que influyen directamente en el resultado del modelo entrenado. Algunos de estos parámetros son:

- *Data Augmentation*: consiste en aumentar la cantidad de datos utilizados durante el entrenamiento. Para ello se realizan modificaciones del contraste y del brillo de las imágenes utilizadas para el entrenamiento y así aumentar la variedad de las imágenes y reducir el *overfitting*.
- *Learning Rate*: afecta en cómo converge el modelo al resultado durante cada paso. Un mayor tamaño puede conseguir alcanzar antes el resultado final y valores más pequeños obtienen resultados más óptimos. Por lo que se necesita determinar un valor apropiado que llegue a un compromiso entre velocidad y calidad de los resultados.
- Número de iteraciones: define el número de veces que se realiza el *back propagation* y entrena de nuevo.

La idoneidad de estos parámetros depende de cada problema que se quiere solucionar. En nuestro caso se han utilizado los mismos que en el trabajo [2], ya que estamos ante el mismo problema, se utilizan los mismos tipos de imágenes y se utiliza la misma arquitectura simplemente ampliando el número de clases a identificar. Durante el entrenamiento se utiliza *data augmentation*, un *learning rate* de 1e-05 y 16000 iteraciones.

5.4 Modelos

Debido a las diferentes y variadas clases presentes en el conjunto de imágenes, se decide realizar diferentes entrenamientos cada uno de ellos agrupando el total de las clases de forma diferente.

De esta forma, dado que los datos de entrada a la red son distintos, se obtendrán diferentes modelos. Las agrupaciones realizadas para los distintos modelos entrenados se muestran en la tabla 2.

Todas las clases			Modelo III		Modelo II		Modelo I	
Nombre	Descripción	Color	Nombre	Color	Nombre	Color	Nombre	Color
Po-a	Mata de Po viva		Po-a		Po-a		Po	
Po-d	Hojas de Po muerta		Po-d		Po-d			
Rock	Fondo rocoso limpio		Rock		Rock		Rock	
Rock-d	Fondo rocoso con otros elementos							
Matte-r	Mata de Po muerta en fondo de roca		Matte					
Matte-s	Mata de Po muerta en fondo de arena				Sand		Sand	
Sand	Fondo de arena limpia		Sand					
Sand-d	Fondo de arena con otros elementos							
Background	Fondo que no corresponde a ninguna de las clases que se quiere clasificar o que no se ha sabido identificar		Background		Background		Background	

Tabla 2: Agrupación de clases para los modelos utilizados

Debido a que las imágenes etiquetadas como Sand y Sand-d son muy parecidas, y además no hay suficientes imágenes para que el entrenamiento sea eficaz para distinguirlas, se ha decidido agruparlas todas en la misma clase Sand. Lo mismo ocurre con la clase Rock y Rock-d que se han agrupado en el tipo Rock.

5.4.1 Modelo I

Al realizar el entrenamiento del modelo I se ha decidido realizar una mínima clasificación, intentando prescindir de las clases que menos aparecen en las imágenes, agrupándolas con la clase que más se le parece. Por eso, se ha agrupado la Po viva (Po-a) y las hojas de Po muerta (Po-d) en una misma clase, todos los fondos de tipo rocoso con la clase Rock y todos los de arena con la clase Sand. En la tabla 3 se muestra la clasificación utilizada para el modelo I.

Nombre	Descripción	Color
Po	Mata de Po viva y hojas de Po muerta	
Rock	Fondo rocoso	
Sand	Fondo de arena	
Background	Fondo que no corresponde a ninguna de las clases que se quiere clasificar o que no se ha sabido identificar	

Tabla 3: Clasificación utilizada en el modelo I

5.4.2 Modelo II

Al realizar el modelo II se aumentó en una clase más la clasificación respecto al modelo I. La clase Po se separa la Po viva (Po-a) y las hojas de Po muerta (Po-d). La tabla 4 muestra la clasificación utilizada para el modelo II.

Nombre	Descripción	Color
Po-a	Mata de posidonia viva	
Po-d	Hojas de posidonia muerta	
Rock	Fondo rocoso	
Sand	Fondo de arena	
Background	Fondo que no corresponde a ninguna de las clases que se quiere clasificar o que no se ha sabido identificar	

Tabla 4: Clasificación utilizada en el modelo II

5.4.3 Modelo III

En el modelo III se vuelve a aumentar una clase más respecto al modelo II. Se añade la clase Matte para representar las matas de Po muerta. En la tabla 5 se muestra la clasificación utilizada para el modelo III.

Nombre	Descripción	Color
Po-a	Mata de Po viva	
Po-d	Hojas de Po muerta	
Rock	Fondo rocoso	
Sand	Fondo de arena	
Matte	Mata de Po muerta	
Background	Fondo que no corresponde a ninguna de las clases que se quiere clasificar o que no se ha sabido identificar	

Tabla 5: Clasificación utilizada en el modelo III

5.5 Métricas de una red neuronal

Para determinar la calidad de los resultados obtenidos en la clasificación de la red neuronal hay que definir una serie de métricas, estas métricas básicamente consisten en una serie de cálculos que se realizan sobre los resultados para poder tener información objetiva del comportamiento de la red neuronal.

Para extraer estas métricas se suele utilizar una tabla denominada matriz de confusión [8] que resume de una manera sencilla las predicciones realizadas por la red neuronal.

La matriz de confusión es una tabla donde se muestra en cada columna el número de veces que se ha predicho una clase y en cada fila el valor real de dichas predicciones. En esta sección se utilizará un ejemplo tipo para explicar las métricas de validación utilizadas en este trabajo, para este ejemplo se ha definido las clases Perro, Gato, Ratón y Back. En la tabla 6 he creado una matriz de confusión que me servirá de ejemplo para mostrar las métricas utilizadas.

	Predicción (píxeles)				
Real (píxeles)		**Perro**	**Gato**	**Ratón**	**Back**
	Perro	320	23	20	12
	Gato	20	700	13	11
	Ratón	100	0	150	8
	Back	10	0	30	300

Tabla 6: Ejemplo de matriz de confusión

A partir de la matriz de confusión del ejemplo de la Tabla 6 se puede extraer que para la predicción de Perro:

- 320 píxeles se han predicho como de la clase Perro y realmente son Perro.
- 20 píxeles se han predicho tipo Perro y eran Gato en realidad.

- 100 píxeles se han predicho tipo Perro y eran Ratón en realidad.
- 10 píxeles se han predicho tipo Perro y eran Back en realidad.

Idealmente todas las predicciones deberían coincidir con la realidad, pero de hecho esto no se suele poder conseguir. A partir de esta matriz se puede obtener información básica del sistema [8]:

- **True Positive (TP)**: indica cuantas veces se ha predicho una clase correctamente. Por ejemplo, para la clase Perro indicaría cuantos píxeles se han predicho como Perro y realmente eran Perro.
- **True Negative (TN)**: indica cuantas veces no se ha predicho una clase y realmente no era esa clase. Por ejemplo, para la clase Perro indicaría cuantos píxeles no se han predicho como Perro y realmente no eran Perro.
- **False Positive (FP)**: indica cuantas veces se ha predicho una clase, pero se ha cometido un error en la predicción. Por ejemplo, para la clase Perro indicaría cuantos píxeles se han predicho como Perro, pero realmente no eran Perro.
- **False Negative (FN)**: indica cuantas veces no se ha predicho una clase, pero se ha cometido un error en la predicción. Por ejemplo, para la clase Perro indicaría cuantos píxeles no se han predicho como Perro, pero realmente sí que eran Perro.

Gracias a estos parámetros se obtienen las métricas que nos indican cómo de bien clasifica la red neuronal [5][8], a continuación se muestran las métricas que se han utilizado durante la validación de este TFG:

- Accuracy (Exactitud): indica de manera general cuántas veces el algoritmo acierta. Es la relación entre las veces en que la predicción ha sido correcta entre todas las predicciones realizadas.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precisión: indica las veces que el algoritmo acierta cuando se ha predicho un positivo. Es la relación de las predicciones positivas acertadas entre todas las predicciones positivas.

$$Precision = \frac{TP}{TP + FP}$$

- Recall (Exhaustividad): indica las veces que el algoritmo acierta un positivo. Es la relación entre las predicciones positivas realizadas y la cantidad de positivos reales.

$$Recall = \frac{TP}{TP + FN}$$

- F_1: indicador que mezcla los conceptos de precisión y recall.

$$F_1 = \frac{2\,TP}{2\,TP + FP + FN}$$

5.6 Validación

Una vez entrenados los modelos se realizará la validación de los mismos con el conjunto de imágenes de evaluación. Para ello, se calculará la matriz de confusión y a partir de ella se extraerán los valores TP, TN, FP y FN de cada clase. Una vez obtenidos estos parámetros se calcularán las métricas definidas en el apartado 5.5 para cada una de las clases.

Para obtener la matriz de confusión se comprueba por cada píxel de todas las imágenes de evaluación a que zona de la matriz de confusión pertenece. La matriz de confusión final será la que se obtiene una vez evaluados todos los píxeles.

A partir de la matriz de confusión se obtienen los valores TP, TN, FP y FN de cada clase:

- TP: corresponde al valor de la celda donde coincide la clase predicha con la clase real, en la tabla 7 se muestra resaltado en rojo el TP de la clase Gato.

	Predicción (píxeles)				
Real (píxeles)		**Perro**	**Gato**	**Ratón**	**Back**
	Perro	320	23	20	12
	Gato	20	700	13	11
	Ratón	100	0	150	8
	Back	10	0	30	300

Tabla 7: Obtención del TP de la matriz de confusión (TP=700)

- TN: corresponde a todas las celdas que no son ni de la clase predicha ni de la clase real. En la tabla 8 se muestran resaltados en rojo los TN de la clase Gato.

	Predicción (píxeles)				
Real (píxeles)		**Perro**	**Gato**	**Ratón**	**Back**
	Perro	320	23	20	12
	Gato	20	700	13	11
	Ratón	100	0	150	8
	Back	10	0	30	300

Tabla 8: Obtención del TN de la matriz de confusión (TN = 950)

- FP: corresponde a todas las celdas de la clase predicha exceptuando la celda que coincide con la clase real. En la tabla 9 se muestran resaltados en rojo los FP de la clase Gato.

		Predicción (píxeles)			
Real (píxeles)		**Perro**	**Gato**	**Ratón**	**Back**
	Perro	320	23	20	12
	Gato	20	700	13	11
	Ratón	100	0	150	8
	Back	10	0	30	300

Tabla 9: Obtención del FP de la matriz de confusión (FP = 23)

- FN: corresponde a todas las celdas de la clase real exceptuando la clase predicha. En la tabla 10 se muestran resaltados en rojo los FN de la clase Gato.

		Predicción (píxeles)			
Real (píxeles)		**Perro**	**Gato**	**Ratón**	**Back**
	Perro	320	23	20	12
	Gato	20	700	13	11
	Ratón	100	0	150	8
	Back	10	0	30	300

Tabla 10: Obtención del FN de la matriz de confusión (FN = 44)

Una vez con estos valores se pueden obtener las métricas de Accuracy, Precisión, Recall y F_1 por cada clase. Para obtener los valores genéricos de cada métrica simplemente se realiza la media de los valores obtenidos por cada clase.

Por ejemplo, para la clase Gato de la matriz de ejemplo de la tabla 6 se obtiene:

$$Accuracy_{Gato} = \frac{TP_{Gato} + TN_{Gato}}{TP_{Gato} + TN_{Gato} + FP_{Gato} + FN_{Gato}} = \frac{700 + 950}{700 + 950 + 23 + 44} = 0.96098$$

$$Precisión_{Gato} = \frac{TP_{Gato}}{TP_{Gato} + FP_{Gato}} = \frac{700}{700 + 23} = 0.96819$$

$$Recall_{Gato} = \frac{TP_{Gato}}{TP_{Gato} + FN_{Gato}} = \frac{700}{700 + 44} = 0.94086$$

$$F_{1_{Gato}} = \frac{2\,TP_{Gato}}{2\,TP_{Gato} + FP_{Gato} + FN_{Gato}} = \frac{2 \cdot 700}{2 \cdot 700 + 23 + 44} = 0.95433$$

Para calcular el valor general del modelo se realiza la media de todas las clases, por ejemplo, para el caso del Accuracy sería:

$$Accuracy = \frac{Accuracy_{Perro} + Accuracy_{Gato} + Accuracy_{Ratón} + Accuracy_{Back}}{4}$$

Finalmente también se normaliza la matriz de confusión para poder realizar una interpretación más sencilla de cómo se comporta el modelo. Para normalizar la matriz de confusión se calcula para cada clase real el porcentaje de veces que se ha predicho en cada una de las clases. Por ejemplo, la tabla 11 muestra la matriz de confusión normalizada del ejemplo de la tabla 6.

		Predicción			
Real (%)		**Perro**	**Gato**	**Ratón**	**Back**
	Perro	85.3	6.1	5.3	3.2
	Gato	2.7	94.1	1.7	1.5
	Ratón	38.8	0	58.1	3.1
	Back	2.9	0	8.8	88.2

Tabla 11: Ejemplo de Matriz de confusión normalizada

6 Resultados

Al realizar los entrenamientos con las colecciones se obtienen los valores para los modelos. Una vez con los modelos entrenados se realiza la inferencia con las colecciones de validación y se obtiene la matriz de confusión y las métricas de los modelos.

6.1 Modelo I

En la tabla 12 se muestra la matriz de confusión para el modelo I y en la tabla 13 las métricas obtenidas a partir de la matriz de confusión.

		Predicción (píxeles)			
Real (píxeles)		**Po**	**Rock**	**Sand**	**Background**
	Po	4975872	61269	68920	3522
	Rock	149276	3676919	63282	22022
	Sand	18929	2176	536556	3172
	Background	172135	333613	7537	100000

Tabla 12: Matriz de confusión del modelo I

	Área (%)	Accuracy(%)	Precisión(%)	Recall(%)	F_1(%)
Po	50.1	95.4	93.6	97.4	95.5
Rock	38.4	93.8	90.3	94.0	92.1
Sand	5.5	98.4	79.3	95.7	86.7
Background	6.0	94.7	77.7	16.3	27.0

Tabla 13: Métricas del modelo I

A raíz de los resultados se puede comprobar que las métricas tanto de la Po como de la roca son relativamente buenas, por encima del 90% en todas las métricas.

En cuanto a la clase arena por lo general los resultados son buenos, pero tiene problemas de precisión (79.3%), esto quiere decir que el modelo se suele equivocar más o menos 1 vez de cada 5 cuando predice que un píxel es arena. También el valor de F_1 no es muy alto por eso vale 86.7%, ya que se ve afectado por la precisión.

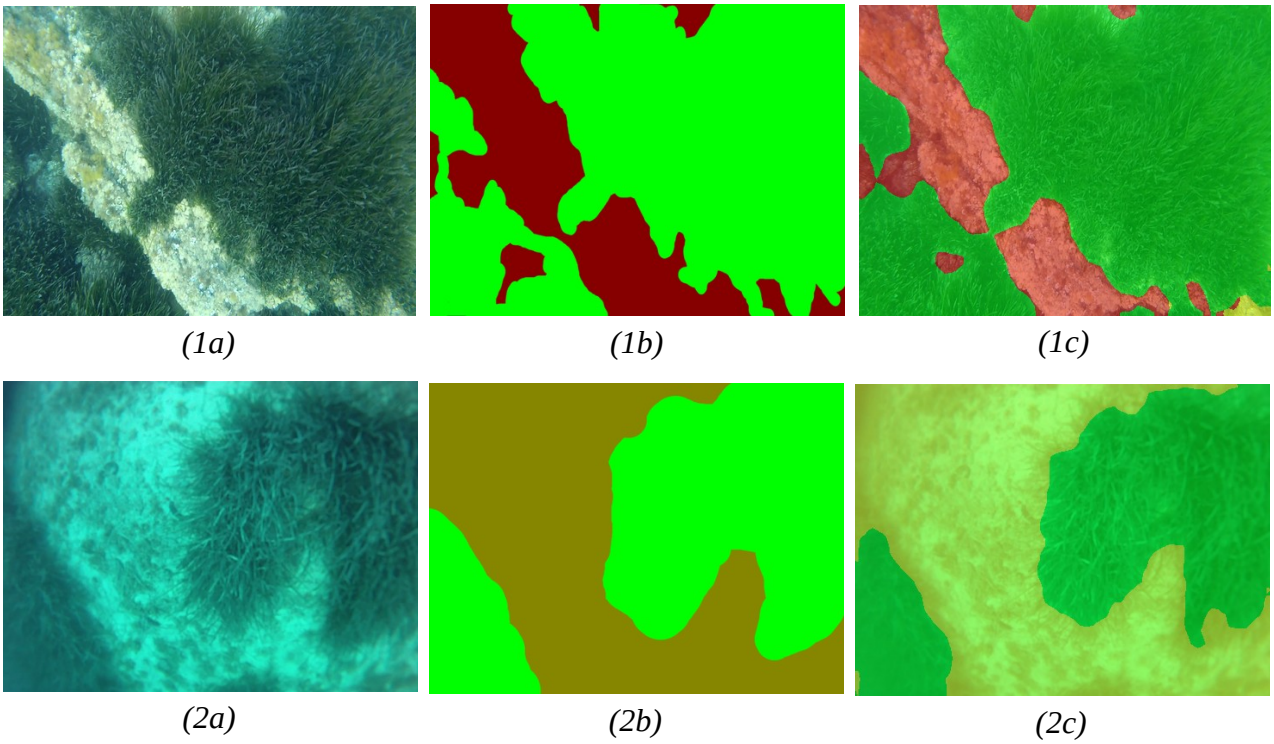
La que peor resultados obtiene es la clase Background, la precisión es relativamente buena, pero el recall es bastante bajo. Esto quiere decir, que cuando se predice un background suele acertar, pero que la mayoría de veces que hay un background no se detecta.

		Predicción			
Real (%)		**Po**	**Rock**	**Sand**	**Background**
	Po	97.4	1.2	1.3	0.1
	Rock	3.8	94.0	1.6	0.6
	Sand	3.4	0.4	95.7	0.6
	Background	28.1	54.4	1.2	16.3

Tabla 14: Matriz de confusión normalizada del modelo I

En la tabla 14 se muestra la matriz de confusión normalizada, y a partir de ella se ve claramente que la mayoría de veces que hay la clase background se predice como roca y arena antes que como background. Del resto de clases se comprueba que la mayoría de veces se predice correctamente.

En la ilustración 15 se muestran el ejemplo de algunas predicciones realizadas a partir del modelo I. En la primera columna están las imágenes originales (1a) y (2a), en la segunda columna se muestran las imágenes etiquetadas manualmente (1b) y (2b) y finalmente en la tercera columna es la predicción realizada por el modelo durante la inferencia (1c) y (2c). Esta predicción se muestra superpuesta a la imagen real.



(1a) *(1b)* *(1c)*

(2a) *(2b)* *(2c)*

Ilustración 15: Predicciones modelo I

En las imágenes de la ilustración 15 se puede apreciar que la Po, la roca y la arena se han detectado con bastante precisión.

6.2 Modelo II

En la tabla 15 se muestra la matriz de confusión para el modelo II y en la tabla 16 las métricas obtenidas a partir de la matriz de confusión.

		Predicción (píxeles)				
Real (píxeles)		**Po-a**	**Po-d**	**Rock**	**Sand**	**Background**
	Po-a	4994161	508	59681	15760	1711
	Po-d	14936	7201	7486	8139	0
	Rock	148326	0	3759054	3954	165
	Sand	17207	28432	55343	459195	656
	Background	180130	0	383701	2186	47268

Tabla 15: Matriz de confusión del modelo II

En este modelo se añade una clase más, distinguiendo entre Po viva y muerta. Pero la cantidad de imágenes que contienen Po muerta es significativamente inferior a la viva.

	Área (%)	Accuracy (%)	Precisión (%)	Recall (%)	F_1 (%)
Po-a	49.7	95.7	93.3	98.5	95.8
Po-d	0.4	99.4	19.9	19.1	19.5
Rock	38.4	93.5	88.1	96.1	91.9
Sand	5.5	98.7	93.9	81.9	87.5
Background	6.0	94.4	94.9	7.7	14.3

Tabla 16: Métricas del modelo II

La Po-a, Rock y Sand tienen un buen comportamiento, similar al del modelo I. La clase Background también se comporta de manera similar, aunque ha empeorado un poco más su comportamiento respecto al modelo I.

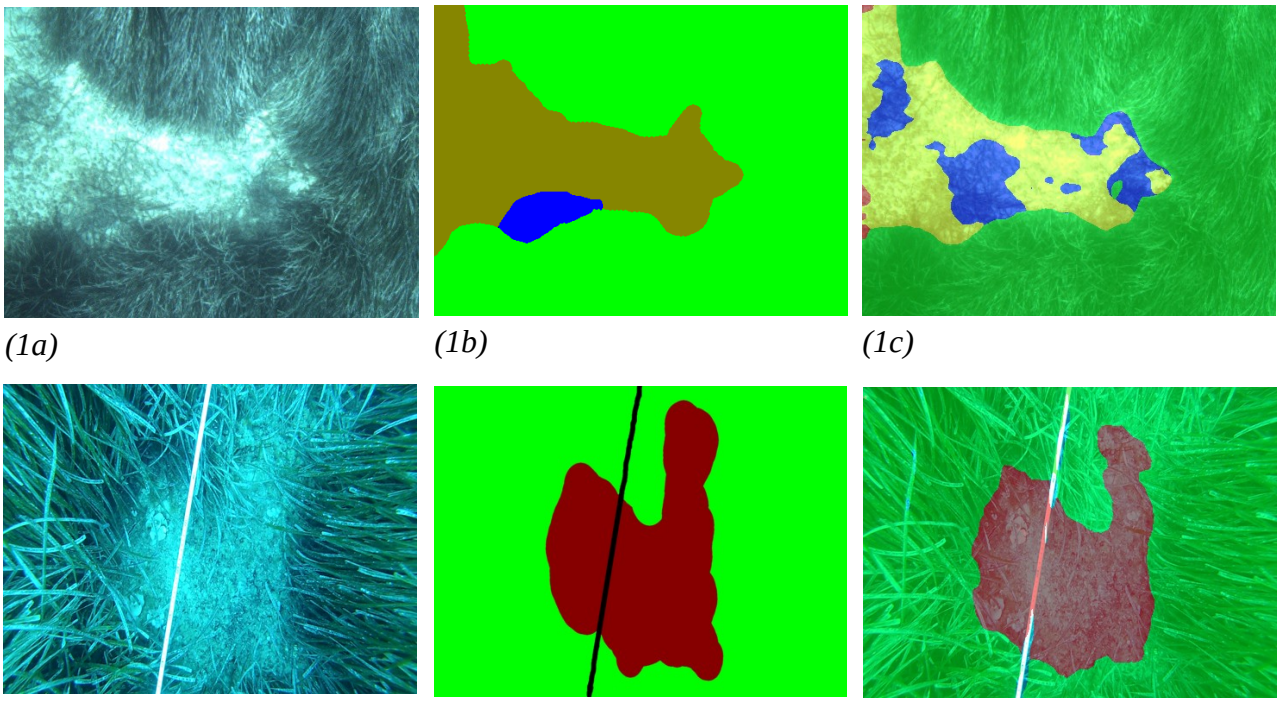
La nueva clase añadida Po-d tiene un comportamiento pobre similar al Background. La precisión, recall y F_1 tienen valores por debajo del 20%, lo que se traduce en que no detecta muy bien esta clase.

		Predicción				
Real (%)		**Po-a**	**Po-d**	**Rock**	**Sand**	**Background**
	Po-a	98.5	0.0	1.2	0.3	0.0
	Po-d	39.6	19.1	19.8	21.6	0.0
	Rock	3.8	0.0	96.1	0.1	0.0
	Sand	3.1	5.1	9.9	81.9	0.1
	Background	29.4	0.0	62.6	0.4	7.7

Tabla 17: Matriz de confusión normalizada del modelo II

De la matriz de confusión normalizada de la tabla 17 se aprecia que la mayoría de veces que se tendría que haber predicho Po-d se predijo como Po-a, esto es debido a su gran similitud. También se puede ver que muchas veces que se predijo Po-d realmente era Sand, esto posiblemente es debido a que en las imágenes del entrenamiento la mayoría de veces la Po-d estaba sobre arena.

En la ilustración 16 se muestran el ejemplo de algunas predicciones realizadas a partir del modelo II. Al igual que en la ilustración 15 en la primera columna están las imágenes originales (1a) y (2a), en la segunda las etiquetadas manualmente (1b) y (2b) y finalmente la predicción realizada por el modelo durante la inferencia (1c) y (2c).



(1a) *(1b)* *(1c)*

(2a) *(2b)* *(2c)*

Ilustración 16: Predicciones modelo II

En la ilustración 16 se puede ver en la figura (1c) como la arena ya no se detecta tan bien como en el modelo I, confundiendo Po-d con arena. En cambio la Po y la roca se puede apreciar que sigue detectándolas con bastante precisión.

6.3 Modelo III

En la tabla 18 se muestra la matriz de confusión para el modelo III y en la tabla 19 las métricas obtenidas a partir de la matriz de confusión.

		Predicción (píxeles)					
Real (píxeles)		**Po-a**	**Po-d**	**Rock**	**Sand**	**Matte**	**Background**
	Po-a	4857920	2298	128020	74794	88	8701
	Po-d	6513	13190	3431	14407	0	221
	Rock	79496	0	3619206	166795	0	40276
	Sand	3958	26914	5332	452948	1750	488
	Matte	5124	24999	0	40745	4301	0
	Background	143885	0	322383	12423	0	134594

Tabla 18: Matriz de confusión del modelo III

En este modelo se ha añadido una clase nueva Matte (mata de Po muerta). Para esta clase al igual que para la Po-d no aparece muchas veces en las imágenes utilizadas durante el entrenamiento.

	Área (%)	Accuracy (%)	Precisión (%)	Recall (%)	F1 (%)
Po-a	49.7	95.6	95.3	95.8	95.5
Po-d	0.4	99.2	19.6	34.9	25.1
Rock	38.3	92.7	88.7	92.7	90.7
Sand	4.8	96.6	59.4	92.2	72.3
Matte	0.7	99.3	70.1	5.7	10.6
Background	6.0	94.8	73.0	21.9	33.8

Tabla 19: Métricas del modelo III

Las clases Po-a, Rock y Sand pese haber empeorado un poco las predicciones, se siguen comportando bastante bien. La clase Sand es la que más ha empeorado aunque sigue comportándose de una manera aceptable.

La nueva clase Matte no se comporta bien, al igual que las clases Po-d y Background. Esto se debe a las pocas imágenes que hay en el entrenamiento para estas clases.

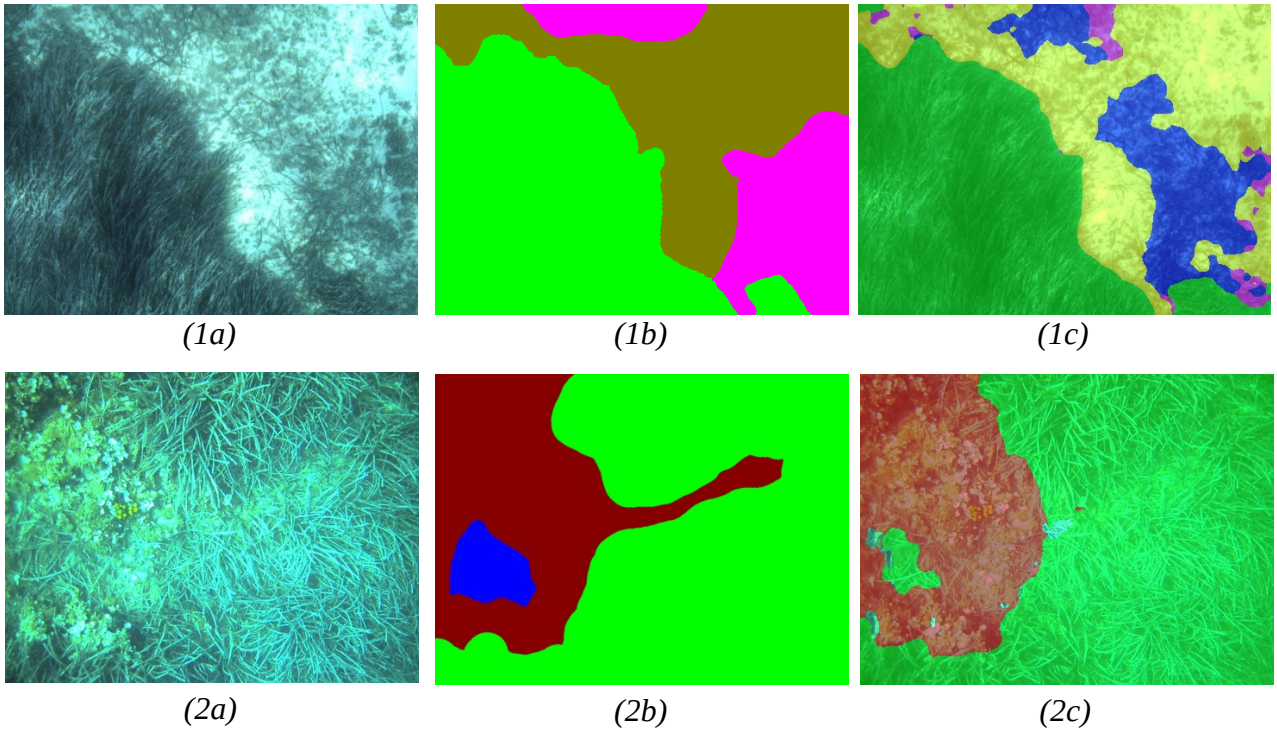
		Predicción					
Real (%)		**Po-a**	**Po-d**	**Rock**	**Sand**	**Matte**	**Background**
	Po-a	95.8	0.0	2.5	1.5	0.0	0.2
	Po-d	17.2	34.9	9.1	38.2	0.0	0.6
	Rock	2.0	0.0	92.7	4.3	0.0	1.0
	Sand	0.8	5.5	1.1	92.2	0.4	0.1
	Matte	6.8	33.3	0.0	54.2	5.7	0.0
	Background	23.5	0.0	52.6	2.0	0.0	21.9

Tabla 20: Matriz de confusión normalizada del modelo III

En la matriz de confusión normalizada de la tabla 20 se aprecia que las clases Po-a, Sand y Rock suelen acertar con las predicciones. En cambio la clase Background y Po-d siguen fallando bastante, la clase Po-d ha mejorado un poco respecto al modelo II pero siguen sin ser valores aceptables y además de confundirse con la Po-d y Sand ahora también se confunde en las predicciones con Matte.

La clase Matte tampoco se comporta bien confundiéndose, sobre todo con la clase Sand y Po-d. Esto es debido al igual que para Po-d y Background a las pocas imágenes que hay para entrenar esta clase.

En la ilustración 17 se muestran el ejemplo de algunas predicciones realizadas a partir del modelo II. Al igual que en la ilustración 15 en la primera columna están las imágenes originales (1a) y (2a), en la segunda las etiquetadas manualmente (1b) y (2b) y finalmente la predicción realizada por el modelo durante la inferencia (1c) y (2c).



(1a) *(1b)* *(1c)*

(2a) *(2b)* *(2c)*

Ilustración 17: Predicciones modelo III

En la ilustración 17 se aprecia que la Rock y la Po se siguen detectando bien, pero la arena se confunde a veces con otras clases y tanto la Po-d como Matte no acaba de identificarlas correctamente.

6.4 Análisis de los resultados

Los resultados obtenidos son relativamente satisfactorios, sobre todo, en la detección de Po, arena y roca. Aunque el resto de clases no se ha conseguido detectarlas de una manera fiable.

El mayor motivo de por qué ocurre esto es que la cantidad de imágenes disponibles utilizadas en los entrenamientos no llegaban a las 250 para todas las clases. Como la mayor parte de las imágenes eran de Po ha propiciado que la detección de esta clase sea la mejor.

Normalmente en este tipo de clasificaciones se suelen utilizar colecciones de imágenes de varios miles por cada clase que se desea poder detectar. Al no disponer de tantas imágenes ha favorecido una disminución de las prestaciones de los resultados obtenidos por los modelos.

Otro error que se ha introducido en la colección de las imágenes ha sido durante el etiquetado. El etiquetado se ha realizado a mano y al etiquetar los límites entre clases no se han perfilado los contornos de manera exhaustiva. Además, personas distintas no marcarían exactamente la misma frontera entre clases en una imagen. También el etiquetado de determinadas clases ha sido complicado, sobre todo, con la clase Matte y con algunas imágenes donde determinadas zonas no tenían calidad suficiente. En estos casos, se producen bastantes discrepancias en cómo dos personas clasificarían las zonas de una misma imagen. Estos errores de etiquetado introducen una disminución en las prestaciones de los modelos entrenados.

7 Conclusión

Pese a que no todas las clases se han conseguido detectar con suficiente fiabilidad, sí que se ha conseguido detectar satisfactoriamente las clases más relevantes (Po, Sand y Rock) obteniendo en estos casos un F_1 entorno al 90%. Además los problemas de clasificación se podrían subsanar ampliando la colección de imágenes utilizadas en el entrenamiento.

El sistema implementado presenta una serie de ventajas respecto a otros procesos de detección del fondo marino, como por ejemplo, permite segmentación por cada píxel de la imagen sin que esta sufra pérdida de información ni requiere ningún tipo de postprocesado permitiendo realizar esta tarea en tiempo real.

La implementación realizada no sólo serviría para clasificar imágenes del fondo marino, cambiando la colección de imágenes y el fichero de configuración del modelo se puede adaptar fácilmente para clasificar otro tipo de imágenes.

Anexo I - Instalación de la aplicación

Para instalar la aplicación en un sistema operativo linux (ubuntu) hay que realizar lo siguiente:

- Instalación de los paquetes de python2.7:

```
~$ sudo apt-get install python2.7 python-pip python-tk virtualenv
```

- Instalación de virtualenv para poder crear un entorno virtual de python

```
~$ pip install virtualenv
```

- Obtener del proyecto de github y descomprimirlo en una carpeta local (llamada por ejemplo Posidonia):

  https://github.com/srv/Posidonia-semantic-segmentation/tree/multiclass

- Creación del entorno virtual para el proyecto:

```
~$ cd Posidonia
~/Posidonia$ virtualenv venv
```

- Activación y configuración del entorno virtual de python

```
~/Posidonia$ source venv/bin/activate
(venv) ~/Posidonia$ pip install -r vgg16fcn8/requirements.txt
```

- Actualización de los enlaces simbólicos

```
(venv) ~/Posidonia$ cd vgg16fcn8/incl/
(venv) ~/Posidonia/vgg16fcn8/incl$ rm *
(venv) ~/Posidonia$ ln -s ../submodules/evaluation/kitti_devkit/ seg_utils
(venv) ~/Posidonia$ ln -s ../submodules/evaluation/ evaluation
(venv) ~/Posidonia$ ln -s ../submodules/tensorflow_fcn/ tensorflow_fcn
```

- Descargar del siguiente enlace las carpetas DATA y RUNS y copiarlas en la carpeta vgg16fcn8:

  https://uibes-my.sharepoint.com/:f:/g/personal/iro403_id_uib_cat/EgqHHLRPLfpNiJuGsDVtnvUBAHJBLrfLgt3NmQ01dNT0uQ?e=hGYA3A

  En la carpeta DATA hay pesos e imágenes necesarias para la aplicación y la carpeta RUNS contiene un modelo ya entrenado para poder realizar la inferencia.

- Ejecutar una evaluación o inferencia:

```
(venv) ~/Posidonia/vgg16fcn8$ python2.7 evaluate.py --RUN multiclass --data_file
DATA/val.txt
```

  El parámetro RUN apunta al modelo que se quiere utilizar para la evaluación. Los modelos se encuentran dentro de la carpeta RUNS.

  El parámetro data-file apunta al fichero que contiene las imágenes que se quieren utilizar para la evaluación o inferencia.

  Una vez ejecutado las predicciones de las imágenes se crearán en la carpeta:

- ./Posidonia/vgg16fcn8/RUNS/multiclass/test_images_prediction/

- Ejecutar un entrenamiento:

```
(venv) ~/Posidonia/vgg16fcn8$ python2.7 train.py --hypes DATA/multiclass_1_light.json
```

  El parámetro hypes corresponde al fichero que contiene los hiperparámetros y las imágenes que se utilizarán en los entrenamientos y en las validaciones. El fichero hypes del ejemplo no es el mismo que se ha utilizado en este trabajo, básicamente se han modificado los hiperparámetros para que el entrenamiento se ejecute más rápido.

  Una vez ejecutado el entrenamiento se creará un modelo entrenado en la carpeta:

  - ./Posidonia/vgg16fcn8/RUNS/multiclass_1_{fecha} (donde {fecha} es el día y la hora en la que se ejecutó el entrenamiento)

Bibliografía

1: Consejo De Gobierno, Decreto 25/2018 de 27 de julio, sobre la conservación de la Posidonia oceánica en las Illes Balears, 2018, http://www.caib.es/eboibfront/pdf/ca/2018/93/1014007
2: Miguel Martin-Abadal, Eric Guerrero-Font, Francisco Bonin-Font and Yolanda Gonzalez-Cid, Deep Semantic Segmentation in an AUV for Online Posidonia Oceanica Meadows Identification, 2018, https://arxiv.org/pdf/1807.03117
3: Miguel Martin-Abadal, Ivan Riutort-Ozcariz, Gabriel Oliver-Codina and Yolanda Gonzalez-Cid, A deep learning solution for Posidonia oceanica seafloor habitat multiclass recognition, 2019, https://proceedings.oceans19mtsieeemarseille.org/session.cfm?sid=32
4: Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning. MIT Press, 2016, www.deeplearningbook.org
5: Google, Curso intensivo de aprendizaje automático, 2019, https://developers.google.com/machine-learning/crash-course/
6: Fran Ramírez, Historia de la IA: Frank Rosenblatt y el Mark I Perceptrón, el primer ordenador fabricado específicamente para crear redes neuronales en 1957, 2018, https://empresas.blogthinkbig.com/historia-de-la-ia-frank-rosenblatt-y-e/
7: Marvin Teichmann, Michael Weber, Marius Zoellner, Roberto Cipolla, Raquel Urtasun, MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving, 2016, https://arxiv.org/pdf/1612.07695v1.pdf
8: Wikipedia, Confusion matrix, 2019, https://en.wikipedia.org/wiki/Confusion_matrix