PROYECTO FIN DE GRADO

TÍTULO: Conteo de objetos con SS-DCNET utilizando imágenes con diferentes niveles de zoom.

AUTORA: Katherine Elizabeth Garcia Gómez

TITULACIÓN: Grado en Ingeniería de Sonido e Imagen

DIRECTORA: Gianna Arencibia Castellanos

TUTORA: Juana Maria Gutiérrez Arriola

DEPARTAMENTO: Ingeniería Telemática y Electrónica

Digitally signed by GUTIERREZ ARRIOLA, JUANA MARIA (FIRMA)
Date: 2023.07.12 16:42:37 +02'00'

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE: Jerónimo López-Salazar Codes

TUTORA: Juana Maria Gutiérrez Arriola

SECRETARIA: Elena Blanco Martín

Fecha de lectura: 26 de Julio de 2023

Calificación:

El Secretario/La Secretaria,

Agradecimientos

Tras un largo viaje por tierras lejanas, cruzando fronteras y superando distancias, agradezco a mi familia, pilares de mi vida: Pedro Luis, Melva Mayela y Karem Stefania, quienes han inspirado mi constancia.

A mi querido esposo, Franz Michelle, mi fiel compañero, tu apoyo en cada paso, lleno de ternura y comprensión, ha enriquecido mi camino.

A mis queridos amigos en la UPM, compañeros leales, compartimos risas y me brindaron fuerzas auténticas. Juntos superamos semanas de intensidad, en armonía y unión, disfrutando momentos de tranquilidad y emoción.

A mí misma, en este viaje de vida y superación, enfrenté retos y obstáculos con determinación. Siempre mantuve mi objetivo en foco, perseverando con entusiasmo, agradeciendo a Dios por guiarme en cada paso.

A mis tutoras, cuyo apoyo fue mi brújula, mi gratitud por su dedicación. Con sus enseñanzas, aprendí y me transformé, en este camino de aprendizaje, siempre les agradeceré.

Resumen

La estimación precisa de la biomasa desempeña un papel fundamental en la acuicultura, ya que permite evaluar el crecimiento de las larvas y peces, determinar la cantidad adecuada de alimento por tanque, evitar el desperdicio de recursos, controlar la densidad poblacional y planificar la cosecha en el momento óptimo. Estos aspectos son esenciales para garantizar la eficiencia y sostenibilidad de las operaciones acuícolas. La estimación de la biomasa proporciona información valiosa para la toma de decisiones estratégicas y contribuye al desarrollo de prácticas más eficientes y responsables en la acuicultura.

Existen diversas tecnologías utilizadas en la estimación de biomasa en acuicultura, como los contadores ópticos infrarrojos y los sensores de movimiento que emplean túneles para el paso de los peces. Sin embargo, la alta inversión requerida para la instalación e implementación de estas soluciones limita su uso generalizado. Por otro lado, el desarrollo de técnicas de aprendizaje automático en los últimos años junto con los avances en su aplicación a imágenes y vídeos ha permitido la implementación eficiente y no intrusiva de algoritmos para abordar el problema del conteo de peces.

En este proyecto se ha abordado el desafío del conteo de poblaciones de peces en la acuicultura mediante el enfoque de aprendizaje automático. Se ha utilizado la red neuronal SS-DCNet, previamente entrenada para la detección de objetos, y se ha adaptado al conteo de larvas de rodaballo mediante el método de *Transfer Learning*. Las redes neuronales convolucionales han demostrado su capacidad para extraer información visual, lo que se ha aprovechado en el conteo de rodaballos en imágenes RGB con diferentes niveles de zoom en tanques de piscifactorías.

Los resultados obtenidos respaldan la viabilidad y precisión del modelo SS-DCNet para el conteo de poblaciones de peces. El sistema desarrollado ha demostrado su capacidad autónoma para estimar el número de rodaballos en cada imagen, obteniendo un error de 3,48% en el caso del modelo sin zoom y el primer conjunto de validación sin zoom, y un error de 14,63% para el segundo conjunto de validación con zoom simulado. Estos resultados representan una mejora significativa en comparación con los métodos tradicionales, reduciendo tanto el tiempo como la complejidad del proceso de conteo. Aunque la precisión puede verse ligeramente comprometida en imágenes con diferentes niveles de zoom, el modelo ha demostrado capacidad de generalización incluso ante imágenes de distinto nivel de zoom al del conjunto de entrenamiento.

Se identificó que la principal fuente de error en el modelo fue su limitación para lidiar con situaciones de oclusión, así como su dependencia de los parámetros de desviación estándar y $C_{máx}$ en la red SS-DCNet. Sin embargo, es importante tener en cuenta que el modelo fue entrenado con un conjunto de entrenamiento de solo diecinueve (19) imágenes, lo que sugiere que los resultados podrían mejorarse significativamente al expandir este conjunto de datos y garantizar un correcto etiquetado de las imágenes. Ampliar el conjunto de entrenamiento y mejorar el etiquetado permitirá al modelo adquirir una mayor capacidad de generalización y mejorar su precisión en la estimación del conteo de peces.

Abstract

Accurate estimation of biomass plays a crucial role in aquaculture as it allows for evaluating the growth of larvae and fish, determining the appropriate amount of feed per tank, avoiding resource wastage, controlling population density, and planning optimal harvests. These aspects are essential for ensuring the efficiency and sustainability of aquaculture operations. Biomass estimation provides valuable information for strategic decision-making and contributes to the development of more efficient and responsible practices in aquaculture.

Various technologies are used in biomass estimation in aquaculture, such as infrared optical counters and motion sensors that utilize tunnels for fish passage. However, the high investment required for the installation or implementation of these solutions limits their widespread use.

The development of machine learning techniques in recent years, along with advances in their application to images and videos, has enabled efficient and non-intrusive implementation of algorithms to address the fish counting problem. In this project, we have addressed the challenge of fish population counting in aquaculture using a machine learning approach. We have implemented the SS-DCNet neural network, pre-trained for object detection, and adapted it for counting turbot larvae through Transfer Learning. Convolutional neural networks have displayed their remarkable capability to extract visual information, enabling their application in the counting of turbot populations in RGB images across a range of zoom levels within fish tanks.

The results obtained support the feasibility and accuracy of the SS-DCNet model for fish population counting. The developed system has demonstrated its autonomous capability to estimate the number of turbot in each image, achieving low errors of 3.48% for the model without zoom and the first validation set without zoom, and 14.63% for the second validation set with simulated zoom. These results represent a significant improvement compared to traditional methods, reducing both the time and complexity of the counting process. Although the precision in counting may be slightly compromised in images with different zoom levels, the model has demonstrated the ability to generalize even with images of varying zoom levels compared to the training set.

The primary source of error identified in the model was its limitation in handling occlusion situations, as well as its dependence on the standard deviation and $C_{máx}$ parameters in the SS-DCNet network. However, it is important to note that the model was trained with only nineteen (19) images, suggesting that the results could be significantly improved by expanding this dataset and ensuring proper image labeling. Expanding the training set and improving labeling will allow the model to acquire greater generalization capacity and improve its precision in fish counting estimation.

Índice de contenidos:

Índice de figuras

Índice de tablas

Lista de acrónimos:

SS-DCNet	Supervised Spatial Divide and Conquer Net
RF	Random Forest
SVM	Support Vector Machines
MSE	Mean Square Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
SGD	Stochastic Gradient Descent
ODS	Objetivos de Desarrollo Sostenible
ADAM	Adaptive Moment Estimation
CNN	Convolutional Neural Network
FC	Fully Connected
RGB	Red Green Blue
VGG	Visual Geometry Group
R-CNN	Region with Convolutional Neural Networks
RPN	Region Proposal Network
YOLO	You Only Look Once
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
MAPE	Mean Absolute Percentage Error

1. Introducción

La estimación de la biomasa es crucial en la acuicultura, ya que permite evaluar el crecimiento de las larvas o peces y determinar la cantidad de alimento necesaria para cada tanque. Esto evita el desperdicio de recursos, garantiza un control adecuado de la densidad poblacional y permite planificar la cosecha en el momento óptimo. Estos aspectos son fundamentales para lograr la eficiencia y sostenibilidad de las operaciones acuícolas [1]. Además, la estimación precisa de la biomasa proporciona información valiosa para la toma de decisiones estratégicas y contribuye al desarrollo de prácticas más eficientes y responsables en la acuicultura.

La estimación de la población de peces en una piscifactoría se realiza generalmente mediante procedimientos tales como la aproximación visual, debido a que estos procedimientos requieren siempre de la intervención humana, suelen consumir mucho tiempo y resultar laboriosos e invasivos. Con el pasar de los años y la evolución de las tecnologías se han utilizado diversos métodos para el conteo de poblaciones de peces siendo estos menos invasivos y más expeditos. Algunas de las tecnologías utilizadas son las siguientes: contadores ópticos infrarrojos y sensores de movimiento, ambas tecnologías hacen uso de un túnel que cada ejemplar debe atravesar, sin embargo, la precisión de los métodos de conteo antes mencionados puede verse afectada por la turbidez del agua, la velocidad a la que los peces se mueven y por oclusión. En ese orden de ideas, otra tecnología empleada es el sondeo acústico, el cual trabaja con la detección de ecos para la estimación de la biomasa, sin embargo, las ecosondas usadas para estos sistemas son de muy alto costo. [2]

En los últimos años, el desarrollo de técnicas de aprendizaje automático ha experimentado avances significativos. Estos avances, junto con su aplicación en el análisis de imágenes y vídeos, han brindado la oportunidad de implementar algoritmos eficientes y no intrusivos para abordar el desafío del conteo de peces. Esta aplicación de la inteligencia artificial ha permitido obtener resultados precisos y confiables en el conteo de poblaciones de peces, lo que tiene un impacto positivo en la acuicultura y en la gestión de los recursos acuáticos [1].

En este trabajo se propone diseñar un sistema basado en Inteligencia Artificial que permita estimar la biomasa de peces presentes en diversos tanques de piscifactorías, mediante el análisis de imágenes RGB. Esta propuesta se divide en varias etapas fundamentales: en primer lugar, se buscará profundizar en la comprensión de la arquitectura SS-DCNet; a continuación, se creará una base de conocimiento compuesta por imágenes manualmente etiquetadas; posteriormente, se diseñarán experimentos utilizando imágenes capturadas a diferentes distancias del agua y con variadas densidades de peces; y finalmente, se propondrá un algoritmo capaz de estimar los parámetros del modelo a partir de las características extraídas de la imagen. Con este proyecto, se espera proporcionar una alternativa viable y no intrusiva para el conteo de poblaciones de peces, generando un impacto significativo tanto en el sector pesquero como en la preservación de los ecosistemas marinos.

Con el fin de alcanzar los objetivos se desea implementar una red neuronal preexistente y entrenada para el reconocimiento de múltiples tipos de objetos. Este proyecto es una continuación del PFG titulado Conteo de Objetos en Imágenes RGB con Redes Convolucionales Mediante División Espacial, realizado por Alejandro González Fernández. [3] El modelo SS-DCNet (*Supervised Spatial Divide and Conquer Network*) se basa en la estimación de densidad mediante la división espacial, y ha demostrado ser uno de los algoritmos más precisos, no solo en el conteo de personas, sino también en la contabilización de vehículos. La aplicación de este modelo en [3] ha demostrado que es una alternativa viable y no intrusiva para el conteo de poblaciones de peces. Debido a que la red SS-DCNet

ha sido previamente entrenada para para la detección de objetos, se aplica el método de Transfer Learning, con el fin de utilizar la información que posee la red para el conteo de objetos tales como vehículos o personas y usarlo en el conteo de larvas de rodaballo.

Este proyecto está conformado por 8 secciones. En primer lugar, se encuentra el marco tecnológico, en el cual se describen las tecnologías que forman parte de este proyecto, los campos de la inteligencia artificial, el funcionamiento de las redes neuronales convolucionales. La siguiente sección describe las especificaciones y restricciones del sistema desarrollado. Seguidamente se encuentra la descripción de la solución propuesta en donde se describe el modelo a utilizar, se analiza la red SS-DCNet y su funcionamiento, parámetros importantes de la red y la generación del conjunto de datos de entrenamiento y validación utilizados. Posteriormente se describe y analiza los resultados obtenidos. En la siguiente sección se describe el impacto del proyecto en la sostenibilidad, seguido del presupuesto necesario para su ejecución. Seguidamente se describen las conclusiones que nos llevan las posibles líneas futuras de trabajo según los resultados obtenidos. Se incluye también la lista de referencias bibliográficas consultadas durante la realización del proyecto y finalmente anexos que contienen el manual de usuario de ejecución de los códigos utilizados en este proyecto.

2. Marco tecnológico

En este capítulo se describen diversos métodos utilizados para el conteo de poblaciones de peces en acuicultura, se explica detalladamente el uso de las redes neuronales en dicha área y se ahonda sobre los algoritmos utilizados en cada etapa del proceso de conteo de objetos al utilizar redes neuronales para conteo de objetos.

2.1. Métodos de conteo utilizados en acuicultura [1]

2.1.1. Tecnología de sensores

Para el conteo de peces a través de tecnología de sensores se utiliza contadores ópticos infrarrojos. El correcto funcionamiento de esta tecnología precisa de un túnel que posee un receptor de haz infrarrojo, el cual está permanentemente apuntado por un haz infrarrojo emisor, cuando el haz no llega al receptor se asume que un pez ha atravesado el túnel. Otro procedimiento utilizado para estimar el número de peces en un tanque es pesar el total de los ejemplares y en el momento de la cosecha determinar el peso promedio del pez, este procedimiento no es preciso debido a que los tamaños y pesos de cada ejemplar varían.

En general los métodos de conteo usando sensores presentan altos errores de precisión si la turbidez del agua aumenta, ya que la penetración del haz de luz disminuye, y también si la tasa de paso en la que los peces atraviesan el túnel es muy rápida ya que no es capaz de detectar oclusión entre peces.

2.1.2. Tecnología acústica

Esta tecnología se soporta en la baja atenuación que causa el agua en las ondas sonoras. Se utilizan ecosondas para enviar pulsos acústicos al entorno, al encontrarse con un objeto de una densidad determinada la onda sonora generada por el pulso será reflejada con mayor o menor fuerza, según la densidad del objeto, por lo tanto, se podrá estimar la densidad del objeto. Un método también utilizado es el de la imagen acústica, utiliza un sonar para crear a través de las reflexiones de los pulsos acústicos emitidos una imagen del entorno, y sobre esta imagen se realiza el conteo. Estos sistemas son una alternativa a las tecnologías de sensores ya que no dependen de un túnel ni de la turbidez del agua para su buen funcionamiento. Sin embargo, los objetos han de tener un tamaño y densidad considerables para ser captados por la sonda.

2.1.3. Tecnología de visión por computador enfocado a imágenes

Los métodos de segmentación de imágenes usados en tecnologías de visión por computador se utilizan para dividir una imagen en regiones o segmentos con características similares. Una vez teniendo cada sección se hace más fácil identificar y separar objetos presentes en la imagen o del fondo de la imagen. Estos métodos son no invasivos para con los organismos en acuicultura, se mencionan a continuación algunos de ellos.

En la **segmentación basada en umbral** se crean máscaras de la imagen teniendo en cuenta un valor de umbral en la escala de grises de esta. Los píxeles cuyo valor esté por encima del umbral se consideran como parte del objeto y el resto será la imagen de fondo. Por otro lado, la segmentación por detección de bordes implica identificar variaciones bruscas en la intensidad de los píxeles que indican los límites de los objetos. En ambos casos se busca diferenciar el fondo de la imagen de los objetos para su posterior conteo, sin embargo, estos métodos son muy dependientes de la nitidez de la imagen y de la turbidez del agua.

En la **detección de objetos** en imágenes se persigue el objetivo de determinar la posición y la clase de objetos presentes en una imagen. Cada objeto tendrá una caja delimitadora (detector) predicha por los algoritmos de detección. El proceso utiliza subventanas de diferentes tamaños que se deslizan sobre la imagen para identificar los objetos candidatos, posteriormente con el uso de clasificadores se determina si estas regiones serán de interés o no. Los clasificadores deben ser adecuados para una rápida y precisa identificación de los objetos. Para que los clasificadores y detectores puedan identificar y localizar objetos en imágenes o vídeos se utiliza el aprendizaje automático o de máquina. Los modelos se entrenan utilizando conjuntos de datos etiquetados, que contienen la información sobre la ubicación y la clase de cada objeto en las imágenes de entrenamiento.

2.2. Aprendizaje de máquina [4]

La inteligencia artificial explora métodos que se pueden utilizar para que una computadora pueda resolver problemas de la manera en que los humanos lo hacen. El aprendizaje de máquina es un campo de la inteligencia artificial que busca que las computadoras aprendan como lo hace el ser humano. El funcionamiento o aprendizaje se basa en mostrar una cantidad de datos y describir instrucciones a seguir para que, en el futuro al presentar una nueva base de datos, el sistema sepa diseñar los procesos necesarios para cumplir con una tarea asignada. Por otro lado, los sistemas expertos son un subcampo de la inteligencia artificial que se centra en enseñar a las computadoras a imitar la lógica humana mediante programación. A diferencia de otros enfoques de la inteligencia artificial, los sistemas expertos no tienen la capacidad de diseñar nuevas soluciones o tomar decisiones independientes. En cambio, siguen las instrucciones preprogramadas por humanos para cumplir con tareas específicas asignadas. Esta aproximación busca capturar y aplicar el conocimiento experto en un dominio particular, brindando asesoramiento o soluciones basadas en reglas y heurísticas previamente definidas.

Los algoritmos utilizados en aprendizaje de máquina se clasifican según el tipo de datos que son introducidos y evaluados posteriormente, siguiendo esta clasificación se pueden mencionar cuatro tipos de algoritmos de aprendizaje: el aprendizaje supervisado, el aprendizaje no supervisado, el aprendizaje semi-supervisado y el aprendizaje reforzado. Estos sistemas de aprendizaje de máquina utilizan dos tipos de datos, las etiquetas, que será el conjunto de datos que guarda la respuesta real o *ground truth* y las características o *features* serían los datos que se han de analizar de cada imagen o dato de entrada para llevar a cabo la tarea suministrada. [5]

Para el **aprendizaje supervisado** se utiliza tanto las etiquetas como las características, ya que el sistema aprende de los datos etiquetados o *ground truth* y por tanto es capaz de predecir la salida de nuevos datos que le sean presentados.

El **aprendizaje no supervisado**, como su nombre lo indica, no emplea un mecanismo de supervisión, es decir, se introduce la base de datos a utilizar o *dataset* sin haber sido etiquetada ni clasificada. El

algoritmo aprenderá únicamente analizando la base de datos de imágenes para encontrar patrones, similitudes o diferencias entre las imágenes y de esta forma las clasificará. [6]

El **aprendizaje semi-supervisado**, precisa de una base de datos en la que no todas las imágenes han sido etiquetadas para el periodo de entrenamiento, de manera que encontrando similitudes entre las imágenes etiquetadas y no etiquetadas podrá clasificar los datos que no han sido etiquetados. La ventaja de este método es el ahorro de tiempo o dinero al poder usar una base de datos que no esté etiquetada completamente, como resultado se suele perder precisión en el resultado final.

El **aprendizaje reforzado** a diferencia de los antes mencionados, no utiliza una base de datos etiquetada para funcionar. El aprendizaje se realiza a través del ensayo y el error. Si imaginamos un robot que no posee mapa ni instrucciones para realizar su tarea, y con el propósito de salir de un laberinto, si el robot comienza por el camino correcto se le da una recompensa, en caso contrario una penalización. Así, aunque al principio no sabe a dónde dirigirse, con cada movimiento que realice irá aprendiendo el camino que debe tomar. El aprendizaje reforzado proviene de la experiencia previa del sistema y las recompensas y penalizaciones que ha recibido en el pasado. [6]

Según el resultado que se espera recibir del modelo de aprendizaje automático, los algoritmos realizan los siguientes tipos de tareas: [7]

> **Clasificación:** Cuando el resultado que se busca es determinar a qué clase pertenece la entrada de datos entre un número limitado de clases. Por ejemplo, si queremos saber si el día está soleado las dos clases posibles serán sí o no, esto se conoce como clasificación binaria. En ese orden de ideas, también existen las tareas de clasificación con probabilidades en las que el resultado que se obtendrá es la probabilidad de que el conjunto de entrada pertenezca a una o varias de las clases de las estudiadas. Este tipo de resultado se conoce como clasificación multiclase, ya que estudia la probabilidad de que pertenezca a más de dos clases distintas. Sin embargo, generalmente se toma como resultado el mayor valor de probabilidad calculada.
>
> **Regresión**: Cuando se busca obtener un número como resultado al conjunto de datos de entrada, se utiliza la regresión. Por ejemplo, en el conteo de objetos se utilizan técnicas de aprendizaje de máquina para regresión, tales como las redes.

Entre los algoritmos de aprendizaje máquina existentes los más utilizados en la actualidad son los basados en las redes neuronales, por ello esta técnica es explicada con profundidad más adelante. Otras técnicas utilizadas son las siguientes:

> **Bosque Aleatorio *(Random Forest, RF)*:** Es un método utilizado en el aprendizaje automático o aprendizaje máquina que se utiliza para tareas de clasificación y de regresión. Se realiza obteniendo varios árboles de decisión, el cual es una estructura que hace una serie de preguntas sobre los datos para llegar a una conclusión. Cada árbol se construye con una muestra de los datos de entrenamiento, así que cada árbol será ligeramente diferente de los demás, ya que la muestra utilizada para la creación de cada árbol será diferente. El resultado final se obtiene de la predicción que obtenga más votos o que se repita más. En la Figura 1 se muestra un ejemplo en el que se busca determinar si un niño podrá salir a jugar utilizando el algoritmo de bosque aleatorio. Se observa que se utiliza árboles de decisión distintos que han sido creados a partir de porciones de las muestras de los datos de entrada. El fin es el de determinar si un niño podrá

salir a jugar un día determinado, si la mayoría de los árboles llegan a la conclusión ‘‘jugar’’ (*play*), el resultado será ese. [8]

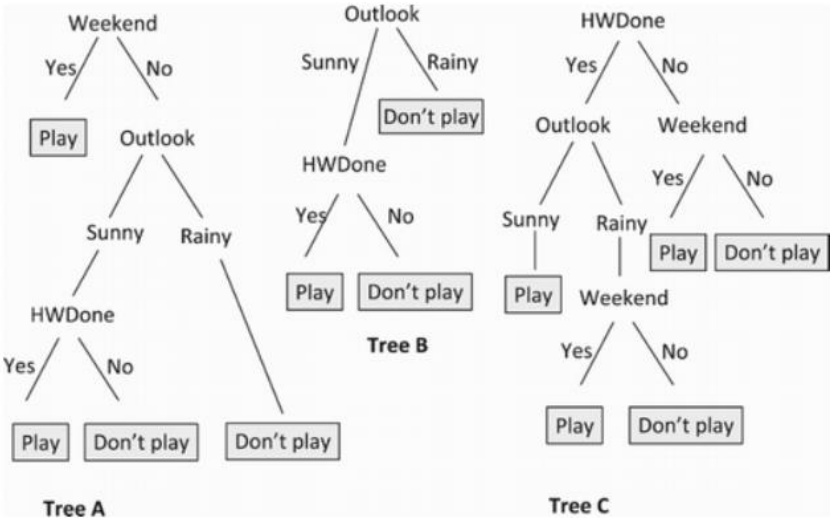


Figura 1. Ejemplo de árboles de decisión del algoritmo de bosques aleatorios. [8]

Máquinas de Vectores de Soporte (*Support Vector Machines, SVM*): Es un modelo lineal utilizado en el aprendizaje de máquina para tareas de clasificación principalmente, pero también se puede utilizar para las de regresión. Se basa en crear una línea o hiperplano que separe los datos de entrada en distintas clases. Un hiperplano es un espacio que ha sido creado a partir de otro plano con un numero de dimensiones ‘‘n’’, por lo que será un subplano de ‘‘n-1’’ dimensiones provenientes del espacio inicial que ha sido dividido en dos partes desconectadas. El punto más cercano desde cada conjunto al hiperplano se denomina vector de soporte (*support vector*) y la suma de los vectores de soporte define el margen, el objetivo del algoritmo es maximizar el margen, es decir, encontrar el hiperplano en el cual el margen es el mayor posible. En la Figura 2 se observa el hiperplano optimo que divide el conjunto de datos rojo y el conjunto de datos azul, el margen maximizado y los vectores de soporte de cada conjunto de datos. [9]

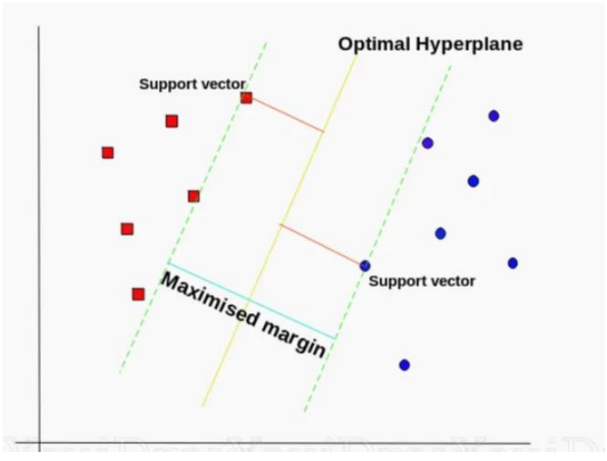


Figura 2. Hiperplano óptimo entre los datos azules y rojos usando el algoritmo SMV [9]

2.3. Redes neuronales

Las redes neuronales son un subconjunto del aprendizaje de máquina y su funcionamiento trata de imitar el de las neuronas biológicas en el cerebro humano. Están formadas por nodos, también llamados neuronas, ubicados en distintas capas. Existe una capa de entrada, diversas capas ocultas y finalmente una capa de salida. Cada nodo tiene asociado unos pesos y un **umbral de influencia**, y sólo si la salida de un nodo individual es mayor al umbral, se activará dicho nodo y enviará datos o características a todos los nodos de la siguiente capa. El funcionamiento de las redes neuronales se basa en introducir datos para que la red llegue a aprender y mejorar su precisión con el tiempo. Estas redes son una gran herramienta ya que tareas como reconocimiento de voz o de imagen tarda para una red neuronal unos minutos y para una persona experta en el tema, unas horas. [10] Se muestra en la Figura 3 un ejemplo de los nodos de cada capa, que al comunicarse con los nodos de la capa siguiente forman una red.

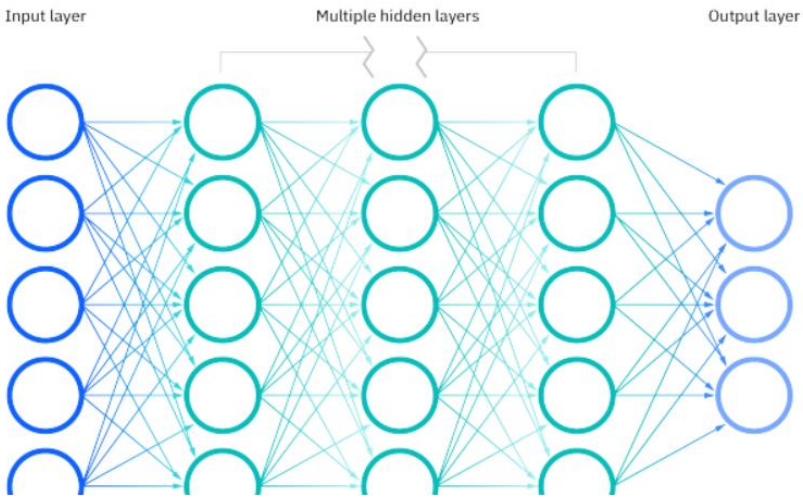


Figura 3. Nodos en capas de una red neuronal. [10]

En cada neurona se realiza una suma ponderada de los datos de entrada que recibe cada nodo y posteriormente a la salida z del nodo se le aplica una función de activación. En la ecuación (1) se muestran los parámetros utilizados al calcular la salida de cada nodo, x_i es el conjunto de datos de entrada del nodo, w_i el peso correspondiente a dicha entrada y b el umbral, *bias* o sesgo, z será la salida del nodo.

$$z = (\sum_{i=1}^{n} w_i \cdot x_i) + b \tag{1}$$

Los parámetros w_i y b son ajustados por la misma red al momento del entrenamiento. En cuanto a los pesos, es la forma que tiene la red de darle más importancia a ciertas características que ha encontrado entre el conjunto de datos de entrada, es decir, la característica más relevante para el resultado buscado tendrá un peso mayor a los demás. Por otro lado, el valor de *bias* no guarda relación con una característica, en lugar de ello, es el valor que tiene la red para ajustar las salidas lo mejor posible al *ground truth* que es la salida deseada de la red

Finalmente, a la salida del nodo se le aplica la función de activación σ, entonces el resultado final del nodo y se calcula como se muestra en la ecuación (2).

$$y = \sigma(z) \tag{2}$$

En la Figura 4 se observa un esquema del proceso realizado en el nodo para la obtención de su salida. De izquierda a derecha se muestra la suma ponderada de las entradas, pesos y el umbral o sesgo; posteriormente la aplicación de la función de activación $f(z)$ y finalmente la salida o *output* del nodo.

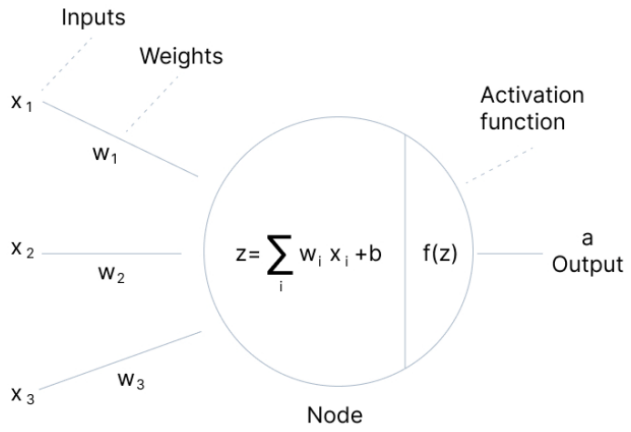


Figura 4. Esquema del proceso de generación de la salida de un nodo. [11]

Existen dos tipos de propagación de información dentro de la red neuronal: [11]

Propagación hacia adelante: Es el proceso básico de una red neuronal, la información se mueve en una sola dirección desde la capa de entrada, a través de las capas ocultas, hasta la capa de salida. La información siempre se mueve hacia adelante, no hay bucles en la red y los pesos que toma la red al principio serán aleatorios. En conclusión, se realiza la propagación hacia adelante para obtener las salidas de la red.

Retropropagación: Esta es una técnica utilizada en el entrenamiento de las redes neuronales. Posterior a la propagación hacia adelante, la red calcula el error entre la salida obtenida y la salida esperada, y se envía dicho error hacia atrás, a través de la red, con el objetivo de ajustar los pesos y sesgos de la red para minimizar el error. Este proceso se repite varias veces con el fin de mejorar la precisión en la red.

El objetivo de utilizar funciones de activación reside en varios puntos que se explican a continuación.

2.3.1. Funciones de activación

Las funciones de activación desempeñan un papel crucial en el funcionamiento de las redes neuronales. Su responsabilidad es decidir si la información de salida de una neurona se activará, en otras palabras, si se transmitirá a la siguiente capa de la red. Esto es esencialmente lo que determina la salida de una neurona basándose en un conjunto de entradas dado. [12]

Los objetivos que se busca cumplir al aplicar una función de activación son los siguientes:

No linealidad: Al introducir no linealidad al modelo a través de las funciones de activación, la red neuronal podrá aprender datos más complejos y realizar tareas que un modelo lineal no es capaz de efectuar. Como la composición de dos funciones lineales genera otra función lineal, todas las capas se comportarían de la misma manera, la red sería un modelo de regresión lineal y las tareas complejas precisan la no linealidad.

Normalización: A fin de evitar el desbordamiento numérico y mantener los valores de la red bajo control, se pueden utilizar funciones de activación que limitan la salida a un rango específico, como lo hacen la Sigmoidea o la Tangente Hiperbólica, las cuales se describirán más adelante.

Derivabilidad: Al multiplicarle una función diferenciable, es decir, derivable en todo punto de su dominio, a la salida de cada neurona, se garantiza la salida sea diferenciable también. Es importante la diferenciabilidad de las salidas de cada neurona por los algoritmos de optimización utilizados para el entrenamiento de la red como el del descenso de gradiente, que se explica más adelante.

A continuación, se describen algunas de las funciones de activación utilizadas en redes neuronales [11]:

Función paso binaria (*Binary Step Function*): Se basa en un valor de umbral ‘‘n’’ que típicamente es ‘‘0’’ para dar el resultado del nodo, el nodo será activado sólo si la salida es mayor o igual al valor umbral y en caso de ser menor que el valor umbral el nodo no se activará, es decir, no enviará su salida a la siguiente capa de la red. Esta función también es conocida como *binary step* por su nombre en inglés. Se observa su gráfico en la Figura 5.

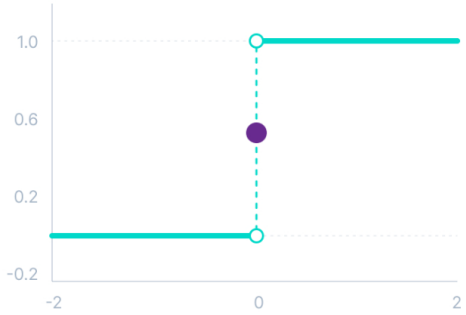


Figura 5. Gráfico de la función paso binaria. [11]

La función paso binaria se define como se muestra en la ecuación (3), y aunque es una función fácil de implementar, en la actualidad es poco utilizada. Su uso en las redes neuronales no es frecuente ya que es una función no diferenciable y al no proveer salidas multivalor no se puede usar para tareas de clasificación multiclase, tampoco es posible usar la retropropagación ya que la derivada de la función es siempre 0.

$$\begin{cases} f(x) = 0 & si \quad x < 0 \\ f(x) = 1 & si \quad x \geq 0 \end{cases} \tag{3}$$

Función Lineal: Es una función que no varía el valor de la salida de los nodos, es a menudo utilizada para la capa de salida de una red neuronal de regresión, donde los valores esperados son números dentro de un rango cualquiera, y no se requiere ajustar la salida. No se utiliza a menudo ya que no es posible ejecutar la retropropagación porque la derivada de la función es una constante sin relación con la entrada del nodo. Se observa su gráfico en la Figura 6.

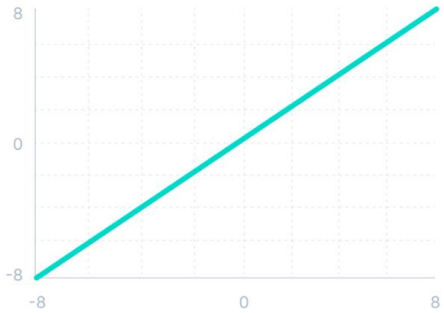


Figura 6. Gráfico de la función lineal. [11]

Matemáticamente la función lineal se define como se muestra en la ecuación (4), siendo x la entrada a la función y también su salida:

$$f(x) = x \tag{4}$$

Las funciones de activación descritas a continuación son no lineales, por lo tanto, la salida de los nodos será derivable y será posible utilizar la retropropagación para ajustar los pesos y umbrales con el fin de obtener una mejor predicción.

Función Sigmoidal: La función sigmoidal (por su nombre en inglés, ***sigmoid***) es una función cuya curva forma una ‘‘S’’ alargada. Esencialmente, la función toma cualquier valor de salida de la neurona y lo comprime en un rango entre 0 y 1. Los valores más grandes y positivos se mapean a 1 mientras que los valores menores de signo negativo se mapean a 0, lo que suele ser útil para tareas de clasificación en la última capa de un modelo de red neuronal, porque las probabilidades de salida sumarían entre todas 1. El gráfico de esta función se muestra en la Figura 7.

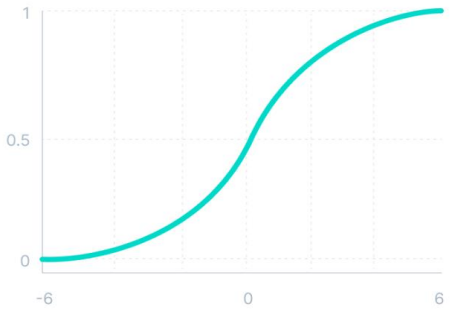


Figura 7. Gráfico de la función sigmoidal. [11]

Matemáticamente la función sigmoidea se define como se muestra en la ecuación **(**2), donde x es la entrada a la función:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

Función ReLU: Esta es la función de activación mayormente utilizada en las redes neuronales. Es una función derivable y permite la retropropagación, ya que no es una función lineal en todo su dominio. Como se observa en la Figura 8, las neuronas sólo se desactivarán si la salida es menor que 0.

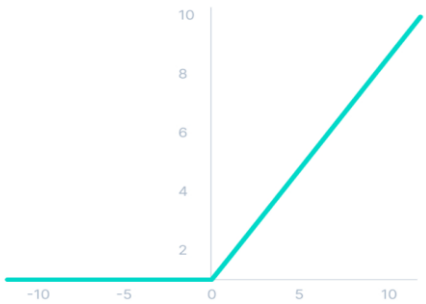


Figura 8. Gráfico de la función ReLU. [11]

Las ventajas de la utilización de esta función de activación es la eficiencia computacional que trae consigo no activar todas las neuronas, en relación con la eficiencia computacional de otras funciones como la sigmoidal. Además, esta función acelera la convergencia del descenso del gradiente hacia el mínimo global de la función de pérdida, que será descrita en breve.

$$f(x) = \max(0, x) \tag{6}$$

Función Tangente hiperbólica (Tanh): Es una función similar a la sigmoidal, por su forma de ‘‘S’’, sin embargo, el rango de salida en este caso es de -1 a 1. Se observa en la Figura 9, que los mayores valores positivos son mapeados a 1, y mientras menor sea la entrada es decir, más negativa, más cercana estará la salida al valor -1. Esta función es usualmente utilizada en las capas ocultas de la red. Una de las ventajas de esta función es que la media de la capa intermedia será 0 o cercana a ello, por lo que los datos estarán centrados en dicho valor y ello hace que el aprendizaje de la red será más sencillo.

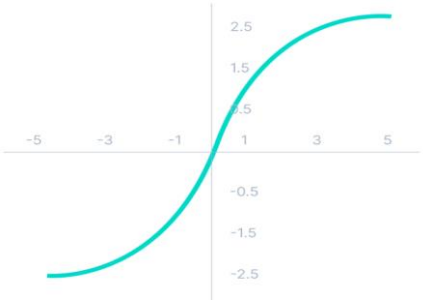


Figura 9. Gráfico de la función tangente hiperbólica. [11]

En la ecuación (7) se observa la representación matemática de la función tangente hiperbólica donde x es la entrada a la función:

$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} - e^{-x})} \tag{7}$$

Función *Softmax*: La función *Softmax* transforma cada entrada en una probabilidad positiva, y dado que la suma de todas las probabilidades generadas es 1, es la mejor opción para problemas de clasificación multiclase. La interpretación de cada valor de la salida de la función es la probabilidad de que la entrada pertenezca a la clase correspondiente. Esta función es utilizada cuando se necesita poder tomar un vector de números reales y mapearlos a un vector de números reales entre 0 y 1 que sumen 1. Se observa el gráfico de la función en la Figura 10.

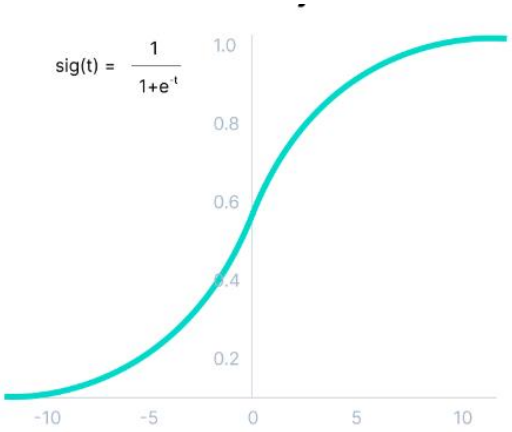


Figura 10. Gráfico de la función *softmax*. [11]

En la ecuación (8) se observa la representación matemática de la función *softmax* donde e es la base del logaritmo natural, x_i el iésimo elemento del vector de entrada x y x_j cada elemento del vector de entrada:

$$f(x_i) = \frac{(e^{x_i})}{\sum_j^n (e^{x_j})} \tag{8}$$

Se ha explicado el proceso de entrenamiento de una red neuronal y distintos elementos que son necesarios para el aprendizaje de una red neuronal. El proceso consiguiente al de entrenamiento es el de validación, el cual también precisa de un conjunto de datos de entrada etiquetados (conjunto de validación) y distinto al de entrenamiento que se le introduce a la red una vez entrenada. El conjunto de datos de entrada utilizado en el entrenamiento de la red ha de ser significativamente grande, para obtener buenos resultados, por el contrario, el set de validación suele ser mucho menor que el de entrenamiento.

Como se ha comentado anteriormente, al realizar el entrenamiento, cada nodo elige distintos valores para los parámetros de peso y umbral según la entrada que le llega, y al realizar la retropropagación, se envía información sobre los errores que se han cometido al elegir dichos parámetros. Estos errores son calculados por una función de coste, entre la salida de cada nodo y el valor real o *ground truth* del set de entrenamiento.

2.3.2. Función de pérdida de calidad y optimización

La encargada del cálculo de los errores asociados a la salida de cada nodo respecto a la salida esperada es la **función de pérdida de calidad** (*loss function***)**, hay diversos tipos de dichas funciones y todas utilizan una entre diversas métricas de precisión tal como: el error cuadrático medio (*Mean Square Error*, MSE), la raíz del error cuadrático medio (*Root Mean Squared Error*, RMSE), el error absoluto medio (*Mean Absolute Error*, MAE), o el error porcentual absoluto medio (*Mean Absolute Percentage Error*, MAPE). El gráfico de la función de pérdida permite hacerse una idea mientras se realiza el entrenamiento de la red de la precisión que va alcanzando con cada iteración, el valor óptimo siempre se encontrará en el mínimo absoluto dibujados por el gráfico.

En la Figura 11, se observa un ejemplo de función de pérdida parabólica calculado con el MSE en el que el error ha llegado al valor más aceptable siendo este el mínimo absoluto de la parábola. Posteriormente ha comenzado a incrementar el error, al suceder esto, se puede asumir que la red se ha sobreentrenado, en este punto la red ya no está aprendiendo, si no memorizando las entradas y etiquetas proporcionadas. Entonces el objetivo a cumplir en la fase de entrenamiento es ajustar los parámetros de peso y umbral de cada neurona, hasta alcanzar el valor mínimo de la función de pérdida. [14]

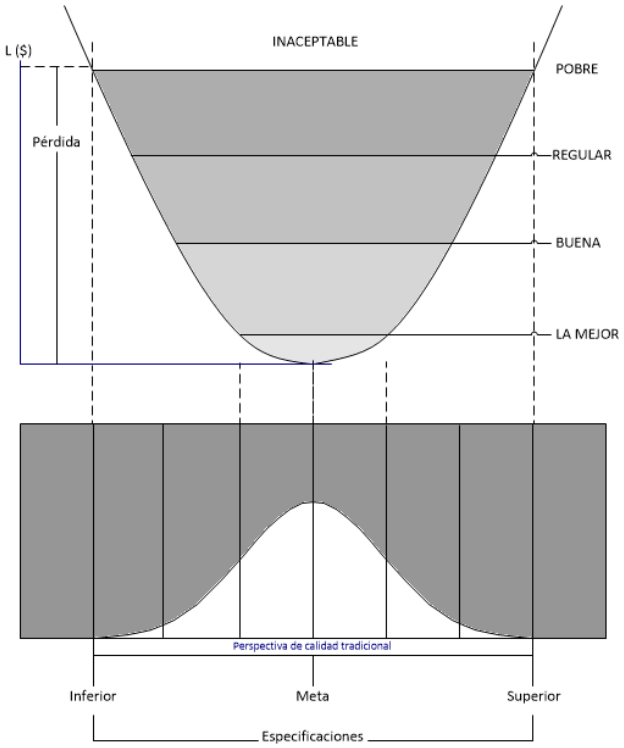


Figura 11. Gráfico de la función de pérdida de calidad parabólica. [14]

En la ecuación (9) se observa la expresión de la función de pérdida de calidad del MSE, siendo y_i el resultado obtenido a la salida de la red, gt_i el *ground truth* y N el número de elementos en el conjunto de entrenamiento. Esta función es una parábola, como la que se observa en la Figura 11, por lo tanto, sólo tendrá un mínimo y será más sencillo para la red aproximarse a él. En el caso de funciones con varios mínimos locales, este procedimiento puede ser más complicado para la red.

$$MSE(y, gt) = \frac{1}{N} \cdot \left(\sum_{i=1}^{N} (y_i - gt_i)^2 \right) \tag{9}$$

Para alcanzar el mínimo absoluto representado por una función de pérdida se utilizan **algoritmos de optimización,** los cuales calculan la derivada parcial de la función de pérdida para así determinar si se debe aumentar o disminuir los pesos elegidos para cada entrada de manera que se llegue a alcanzar el mínimo absoluto de dicha función. De esta manera, un algoritmo de optimización sabe la dirección en la que se ha de mover, sin embargo, para saber cuánto o el valor al cual se ha de modificar los pesos, el algoritmo tiene en cuenta el hiperparámetro llamado tasa de aprendizaje (*learning rate*). Multiplicando el gradiente obtenido por la tasa de aprendizaje se elige un nuevo peso en cada iteración.

La tasa de aprendizaje es vital para la correcta elección del nuevo peso en cada iteración ya que si es muy alta los cambios en los parámetros serán muy grandes y es posible que el peso elegido se encuentre

por encima del mínimo absoluto. Por lo tanto, el valor óptimo o mínimo puede que nunca se llegase a alcanzar. En el caso contrario, si la tasa de aprendizaje es muy pequeña, converger a una solución tardará más tiempo y habrá posibilidad de que no al encontrar un mínimo local, no se pueda salir de él.

Algunos algoritmos de optimización usados en las redes neuronales son los siguientes:

Gradiente Descendiente Estocástico (*Stochastic Gradient Descent, SGD*): Este es un método utilizado para encontrar los valores mínimos de una función de pérdida en las redes neuronales y en general también es utilizado en optimización matemática. El proceso comienza durante el entrenamiento de la red con la asignación aleatoria de los parámetros de peso y umbral de influencia, posteriormente del conjunto de base de datos de entrada selecciona un subconjunto aleatorio en cada paso para evaluación. Seguidamente se calcula la derivada de la función de pérdida. La derivada mide la pendiente de la curva de la función, por lo tanto, al crecer la función, su derivada también lo hará y cuando la función llegue a su punto mínimo, la derivada valdrá 0. Así, se seguirá recorriendo la función de pérdida en el sentido del vector gradiente y ajustando los parámetros de pesos hasta llegar al punto con derivada de valor 0, o, hasta alcanzar un número máximo de iteraciones. [15]

En la Figura 12, se observa una función de pérdida con dos mínimos. Se visualiza que el algoritmo de optimización realiza tres iteraciones de cálculo de gradiente y elección de nuevos pesos hasta encontrar el mínimo absoluto de la función. En la práctica se ha de probar con diversos valores de tasa de aprendizaje hasta conseguir los mejores resultados de validación.

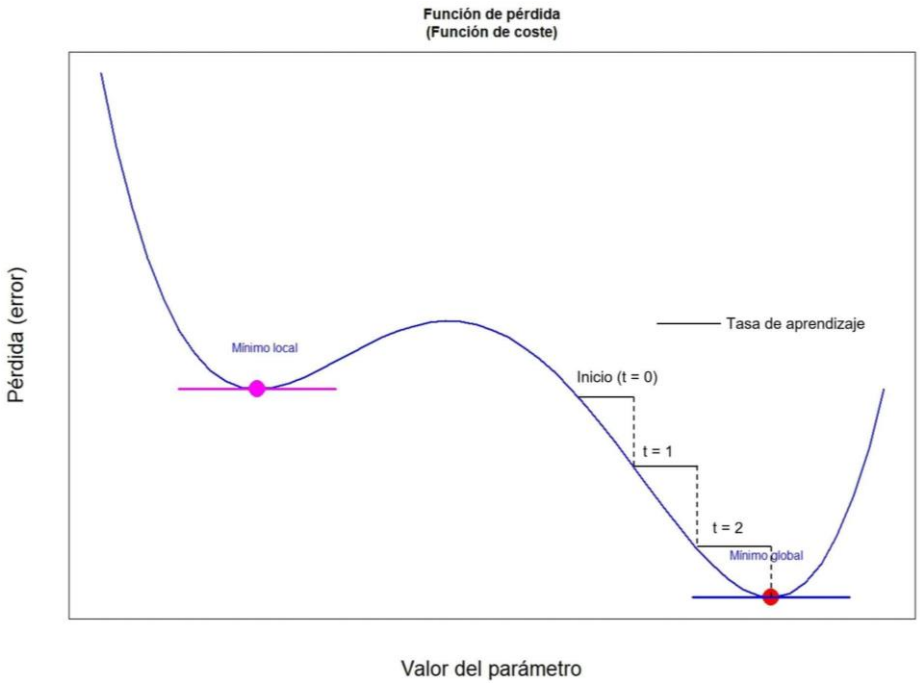


Figura 12. Gráfico de función de coste con mínimo local y global. [16]

Estimación Adaptativa del Momento (*Adaptive Moment Estimation*, ADAM): Es un algoritmo que realiza dos cálculos, el del momento de primer orden y el de segundo orden para una elección adaptativa de la tasa de aprendizaje. ADAM mantiene un promedio en

> movimiento de los gradientes calculados para cada parámetro de entrada que se conoce como ‘‘primer momento’’, este promedio brinda una idea de la dirección general en la que los gradientes han estado moviéndose lo cual contribuye a la estabilización de los pesos elegidos. Por otro lado, mantiene un promedio en movimiento del cuadrado de los gradientes llamado ‘‘segundo momento’’, con este promedio se sabe si habrá que reducir la tasa de aprendizaje en caso de que el valor de los gradientes sea muy grande, o aumentarla en el caso contrario. De esta manera, ADAM podrá tomar pasos más pequeños si los gradientes son grandes para prevenir inestabilidad en los parámetros de peso, y pasos más grandes si las magnitudes de los gradientes son pequeñas, para acelerar el aprendizaje. [17]

Si bien el funcionamiento descrito anteriormente es una generalización de las redes neuronales, existe un subconjunto de estas redes llamado aprendizaje profundo (*deep learning*), dentro del cual se encuentra un tipo de redes neuronales que cuentan con un mayor número de capas ocultas que realizan tareas específicas a las entradas utilizadas durante el entrenamiento de la red. Actualmente la clasificación y reconocimiento de imágenes o *computer vision* se realiza mayormente a través de estas redes derivadas del aprendizaje profundo, las redes convolucionales.

2.4 Redes neuronales convolucionales

Las redes neuronales convolucionales (***Convolutional Neural Network,* CNN**) son especialmente utilizadas para tareas de reconocimiento de imágenes. Las imágenes son matrices de valores de color distribuidas en una dimensión de altura y ancho definidas, por lo tanto, los objetos y formas presentes en las imágenes son también una matriz subconjunto con valores de color. A partir de filtros y agrupando capas, las CNN son capaces de dar sentido a los datos de una imagen. Como en las redes neuronales existen tres tipos de capas involucradas en el funcionamiento de las CNN, se encuentran en el siguiente orden, la capa convolucional, la capa de agrupación o *pooling* y la capa completamente conectada (FC por su nombre en inglés *fully connected*). La capa completamente conectada suele considerarse como la capa de salida si es la última, ya que puede haber varias de estas capas ocultas antes de la de salida. Por otro lado, de las capas convolucionales y de las de agrupación pueden existir varias adicionales. [18]

El funcionamiento de una **capa convolucional** consiste en la aplicación de convoluciones, siendo estas productos y sumas, entre la capa de entrada y uno o varios kernel, su conjunto se llama filtros. Si la imagen es a color, contará con tres capas una para cada color primario: rojo, verde y azul (RGB *Red Green Blue*) por lo tanto será necesario que el filtro tenga tres dimensiones también. En la Figura 13, se observa que la entrada es seccionada en matrices más pequeñas, este proceso se realiza recorriendo todas las posibles posiciones de la imagen y en cada una se obtiene un valor de salida. Finalmente genera una matriz de capa convolucionada, que contendrá características extraídas de la imagen. [19]

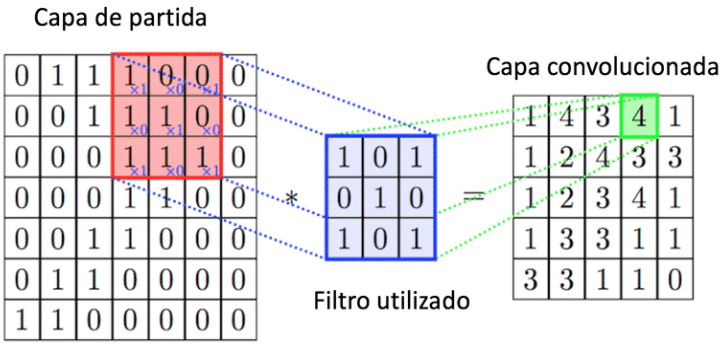


Figura 13. Ejemplo de procedimiento de convolución. [19]

Al desplazar el kernel o filtro utilizado por la imagen de entrada se obtiene una imagen filtrada, sin embargo, en realidad se utilizan variados filtros con el fin de obtener variadas imágenes que esbocen distintas características de la imagen original. De esta manera, la red podrá en el futuro distinguir entre objetos presentes en la imagen. Finalmente, se le aplica una función de activación a las imágenes convolucionadas, por ejemplo, la ReLU la cual se ha descrito anteriormente, con el fin de obtener mapas de detección de características, tal y como se muestra en la Figura 14.

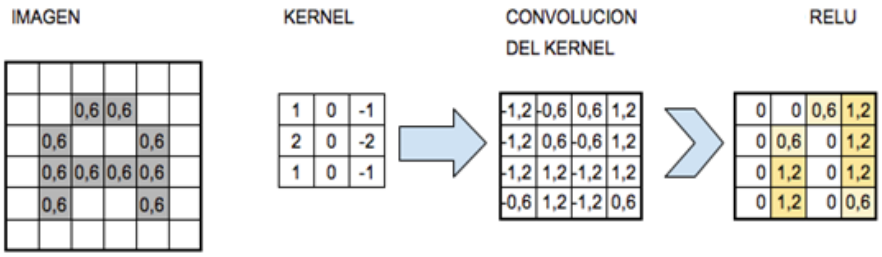


Figura 14. Ejemplo de procedimiento de convolución. [20]

Posterior a la obtención de los mapas de características debido a que se tendrán diversos de éstos, el costo computacional de utilizar todos los datos y el aumento de estos si se realiza otra convolución implicaría un gran procesamiento. Es por eso, que en la **capa de *pooling*** se realiza un submuestreo de cada mapa de características obtenido gracias a la convolución. El submuestreo se realiza con la finalidad de conservar las características de mayor importancia que ha detectado cada filtro. En resumen, al realizar una convolución de la imagen de entrada se crea una imagen de menor dimensión, pero con una profundidad correspondiente al número de filtros que se hayan elegido para la convolución. Este procedimiento es lo que le da el nombre al aprendizaje profundo, la profundidad se refiere a las imágenes obtenidas de cada filtro utilizado, que han sido generadas de una única imagen.

Como ejemplo se observa la Figura 15, en la cual la matriz de 4×4 píxeles se reduce a una matriz 2×2 la cual es una matriz de cuatro píxeles. Existen diversas técnicas para realizar el submuestreo y elegir el valor que prevalece, una de estas es la llamada *maximum pooling* o submuestreo máximo, el cual se basa en tomar el valor máximo obtenido de cada matriz de 2×2 píxeles (o del tamaño indicado para

el submuestreo). Esto se realiza a lo largo de la imagen de arriba abajo y de izquierda a derecha hasta completar toda la imagen.

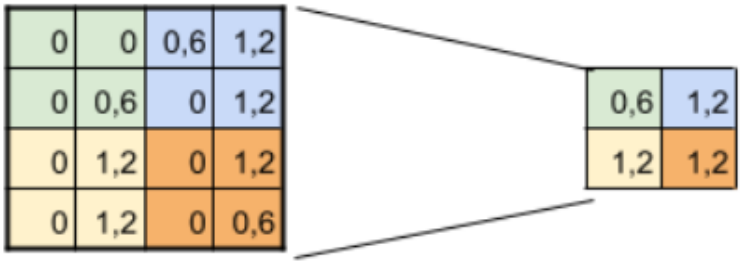


Figura 15. Ejemplo de procedimiento de submuestreo de 2x2. [20]

Usualmente las redes neuronales convolucionales realizan varias iteraciones de convolución y de submuestreo antes de generar la capa *fully connected.* La capa FC se genera tomando la salida obtenida del último submuestreo de dimensión (alto, ancho, numero de mapas) y es ‘‘aplanada’’, es decir, deja de ser profunda y la dimensión número de mapas es igual a 1. Posteriormente, se le aplica una función de activación para crear la capa de salida final.

Se observa en la Figura 16, el proceso por el que pasa la imagen de entrada dentro de una CNN, siendo *softmax* una función de activación ejemplo que genera una salida según las clases definidas, y perro, gato y caballo son los resultados de la red o clases definidas. En general, las capas ocultas de convolución y pooling se encargan de procesar los datos de manera que la red sea capaz de resolver las tareas planteadas. Así mismo, mientras más capas o filtros se apliquen a la imagen más características y por tanto precisión obtendrá la red neuronal.

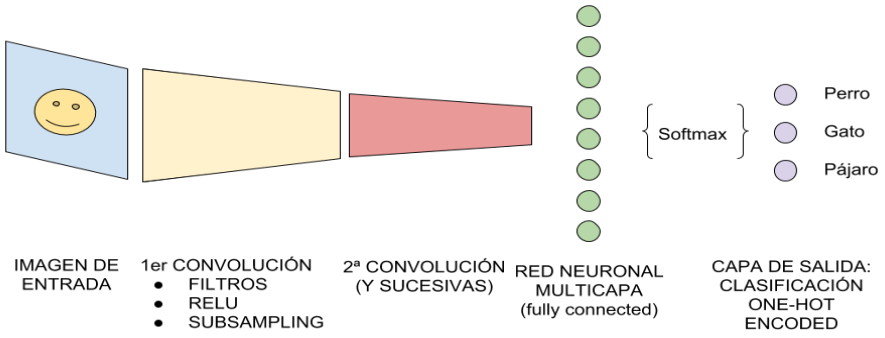


Figura 16. Arquitectura de una CNN [20]

La arquitectura descrita anteriormente es una generalización de la utilizada en redes neuronales convolucionales, sin embargo, existen distintos componentes que varían entre una red a otra que las hacen distintas. Dichas arquitecturas se describen a continuación.

2.4.1. Arquitecturas de las redes neuronales convolucionales

Hasta ahora se ha puntualizado que las redes neuronales convolucionales son una clase dentro de las redes neuronales profundas (*deep neural networks*) que se utiliza comúnmente para el análisis de imágenes en tareas de segmentación, clasificación o detección de objetos. A continuación, se describen algunas arquitecturas utilizadas en el campo de procesamiento de imágenes que han mejorado la precisión de las redes neuronales convolucionales más generales.

VGGNet: Son redes con una arquitectura simple en comparación con otras, sin embargo, suele tener mejor precision que arquitecturas más complejas. De esta arquitectura hay 6 versiones disponibles y las más populares son la VGG-16 y la VGG-19, sus siglas vienen del nombre del grupo de geometría de Oxford en inglés *Visual Geometry Group*. En las arquitecturas VGG, las capas convolucionales se apilan una encima de la otra, aumentando sucesivamente tras cada capa el número de filtros utilizados. Otra característica importante es que el tamaño de los filtros es de 3×3 píxeles. Este tamaño de filtro ha sido elegido mediante pruebas de que el área que cubre dos filtros de 3×3 píxeles es mayor a la que cubriría un filtro de 5×5 píxeles, siendo la primera opción computacionalmente más eficientes ya que requiere menos operaciones de multiplicación. [21][24]

A medida que las redes son más profundas, la VGG será susceptible al problema del gradiente desvanecido, también conocido como *vanishing gradients* el cual sucede cuando los gradientes de las primeras capas de la red neuronal se vuelven muy pequeños y pueden tender a 0. Esto sucede en las redes profundas debido a las multiplicaciones repetidas de gradientes a través de las capas de retropropagación, como resultado los pesos y sesgos variarán muy poco durante el entrenamiento y el resultado es que la red no logre aprender correctamente o necesite más tiempo para aprender.

Cabe resaltar que este tipo de CNN funcionan muy bien para crear nuevos modelos a partir de modelos iniciales pre-entrenados, utilizando la técnica de transferencia de aprendizaje o *transfer learning* y también han demostrado ser efectivas para tareas de clasificación de imágenes una vez ajustados los parámetros. A continuación, se describen los procesos llevados a cabo en la arquitectura VGG-16.

> **VGG-16:** Como se mencionó anteriormente, se utilizan filtros de 3×3 píxeles que, aunque son considerados pequeños, aseguran un mejor rendimiento computacional. Además, estos filtros permiten la extracción de detalles en una escala más granular, comparativamente superior a lo que sería posible con filtros de mayor tamaño..
>
> Cada filtro se mueve a través de la imagen o el mapa de características de entrada utilizando un *stride* o paso de 1. El *stride* se refiere a cuántos píxeles se mueve el filtro a través de la entrada en cada paso. Un *stride* de 1 significa que el filtro se mueve píxel por píxel, lo que permite una exploración completa y detallada de la entrada.
>
> Para asegurarse de que el filtro pueda moverse a través de toda la imagen sin perder información en los bordes, se utiliza una técnica conocida como *padding* o relleno. En este caso, se agrega un relleno de tamaño 1 alrededor del borde de la imagen o del mapa de características, permitiendo que el filtro se aplique completamente hasta el borde de la imagen original.
>
> Después de cada capa convolucional, se aplica la antes mencionada técnica de *max pooling* o submuestreo máximo con un tamaño de 2 y un paso de 2, con el fin de disminuir las dimensiones espaciales de la entrada, ancho y alto tomando el valor máximo dentro de una ventana deslizante, la descrita anteriormente de 2×2 píxeles. Esta técnica permite reducir la

cantidad de parámetros (pesos y sesgo) y computación necesarios, al mismo tiempo que ayuda a la red a identificar las características más importantes.

A pesar de que la agrupación máxima reduce el tamaño espacial de la representación, el número de filtros y por lo tanto la profundidad de la representación aumenta con cada capa. Esto significa que, aunque el tamaño espacial de la representación se reduce, la red puede aprender características cada vez más complejas a medida que se avanza en las capas. De esta manera, la red puede identificar y aprender características de alto nivel a pesar de la disminución del tamaño espacial. [22] Además, esta arquitectura se puede encontrar de libre acceso y utilizar pre-entrenada con un conjunto de datos de nombre ImageNet, el cual contiene 14 millones de imágenes, cada una agrupada en categorías.

En la Figura 17, se observa una representación de la VGG-16, la dimensión original de la imagen es de 224×224 píxeles, con tres dimensiones de profundidad por el RGB. Al implementar *max pooling* con paso de 2 y alto de 2 la ventana deslizante será una matriz de 2×2 píxeles. Finalmente, el resultado del max pooling genera una imagen con la mitad de las dimensiones iniciales de largo y ancho pero mayor profundidad debida a los filtros aplicados en las convoluciones. El número de filtros aplicados en cada convolución va en aumento y por eso antes de alcanzar la capa fully connected la profundidad de la imagen es de 512.

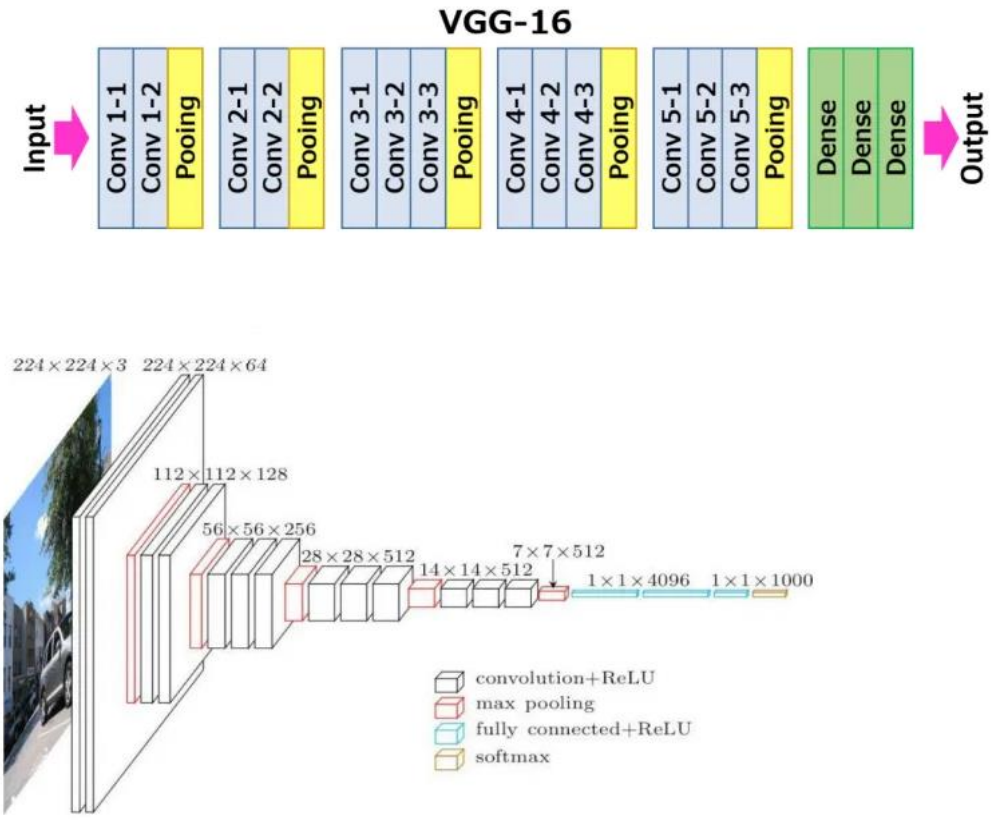


Figura 17. Ejemplo de la arquitectura VGG-16. [23]

Inception: Es una arquitectura de red neuronal propuesta por Szegedy, se destaca por su eficiencia en velocidad y tamaño, teniendo menos parámetros. [25] Se basa en la idea de una "microarquitectura", donde cada capa oculta tiene una representación de nivel superior de la imagen y se puede elegir entre varias operaciones, como agrupamiento o diferentes tipos de capas. Esta arquitectura utiliza varios

kernels o filtros en lugar de solo uno, lo que permite detectar características tanto pequeñas como de alto nivel. En el diseño de Inception, se sigue un agrupamiento promedio con convoluciones de varios tamaños que luego se concatenan. Los parámetros de estos *kernels* pueden ser aprendidos a partir de los datos.Un aspecto único de Inception es el uso de convoluciones de 1x1 para reducir la dimensionalidad de las características y, por lo tanto, disminuir los cálculos necesarios. Esto resulta en un menor uso de la RAM durante la inferencia, lo que mejora la eficiencia.[21]

A diferencia de otras arquitecturas como VGG, en Inception las operaciones suceden en paralelo, lo que contribuye a su eficiencia. Además, para manejar el gran volumen de salida de estos paralelismos, se introducen filtros de 1×1píxeles para la reducción de la dimensionalidad. En la Figura 18 se observa un ejemplo de la arquitectura de *Inception.*

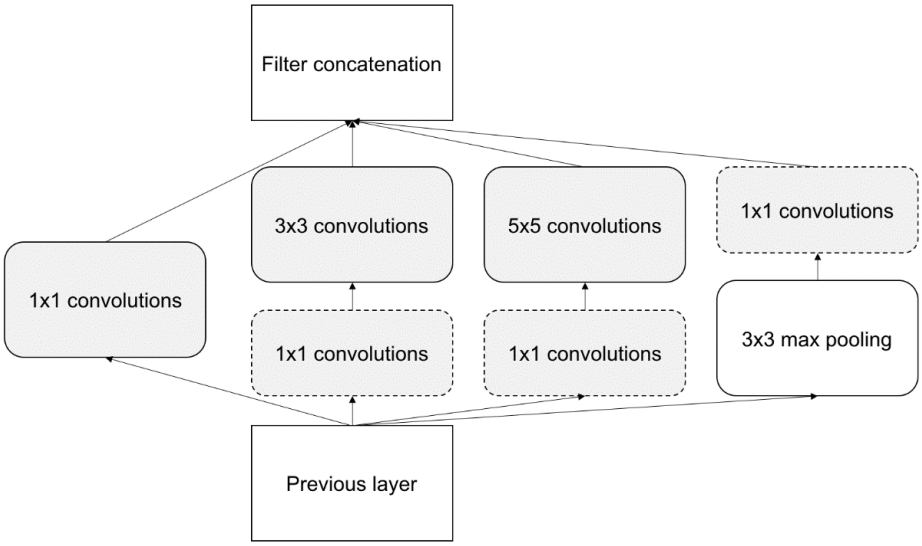


Figura 18. Ejemplo de la arquitectura *Inception*. [22]

ResNet-50: Es una red neuronal propuesta en [26] que ha desmotrado que las redes más profundas, es decir, con más capas, pueden ser entrenadas efectivamente. En muchas redes neuronales, cuanto más profunda es la red, más se satura la precisión. Esto no sucede porque la red esté sobreajustada ni a la presencia de un gran número de parámetros, se debe a que los gradientes tienen dificultades para propagarse cuando son muchas las capas, por el antes mencionado problema del desvanecimiento del gradiente.

La solución propuesta por ResNet a este problema es la introducción de bloques residuales. En estos bloques, la entrada de una capa se suma a la salida de una capa posterior a esto se le llama un atajo o *skip connection*, lo que permite que los gradientes se propaguen directamente a través de la red. Estos atajos permiten que el error se retropropague a las capas anteriores de manera más efectiva, permitiendo entrenar redes más profundas.

Por lo tanto, en lugar de cada capa aprender una nueva representación, cada capa en una ResNet está aprendiendo la resta o residuo entre su entrada y una representación objetivo ideal, lo que es un objetivo de aprendizaje más manejable y evita el problema de desvanecimiento del gradiente.[21] En la Figura 19 se muestra la retroalimentación que se produce en las capas de una ResNet-50, donde x es la entrada y ‘‘*weight layer*’’ o capas de pesos son las capas pertenecientes a la red. El resultado de la segunda capa será la suma de su salida más la salida tras haberle aplicado la función de activación ReLU.

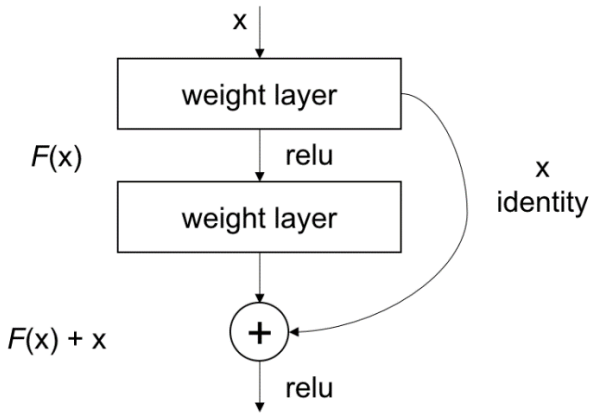


Figura 19. Ejemplo de la arquitectura ResNet-50. [22]

U-NET: Es una arquitectura de red que utiliza extensivamente la ampliación de datos o *data augmentation* con el fin de aprovechar al máximo las muestras y etiquetas existentes. Esto la hace una red única, ya que usualmente para el buen aprendizaje de una red se necesita un conjunto de datos de entrenamiento de gran tamaño. La arquitectura cuenta con una vía de contracción o codificador para recoger el contexto de la imagen y una vía de expansión de decodificador simétrica para permitir una localización píxel a píxel del objeto. Su nombre U-NET viene dado por la forma que describe su arquitectura, el lado izquierdo de la U es la codificación y el derecho la decodificación. Se observa en la Figura 20 la arquitectura U-NET. [34]

1. Camino de contracción (Codificador): Consiste en dos convoluciones de 3×3 píxeles cada una, seguidas por una función de activación ReLU, y luego un *max pooling* de 2×2 píxeles con paso 2 para la reducción de la dimensionalidad. Cada paso hacia abajo en el camino de contracción dobla el número de características mientras reduce a la mitad la resolución espacial de la imagen.

2. Camino de expansión (Decodificador): En esta etapa, la imagen se reconstruye para devolverla a su tamaño original. Cada paso del camino de expansión consta de una convolución transpuesta de 2×2 píxeles, seguida por una concatenación o *copy and crop* con el mapa de características correspondiente al camino de contracción. La red intenta reconstruir o decodificar la imagen original a partir de la versión simplificada o codificada que tiene. Para hacerlo, utiliza las características aprendidas durante la parte de codificación y vuelve a ampliar la imagen hasta su tamaño original. Finalmente, la red intenta determinar a qué categoría pertenece cada píxel de la imagen realizando una convolución con un filtro 1×1 píxeles y de profundidad igual al número de clases que se considerarán.

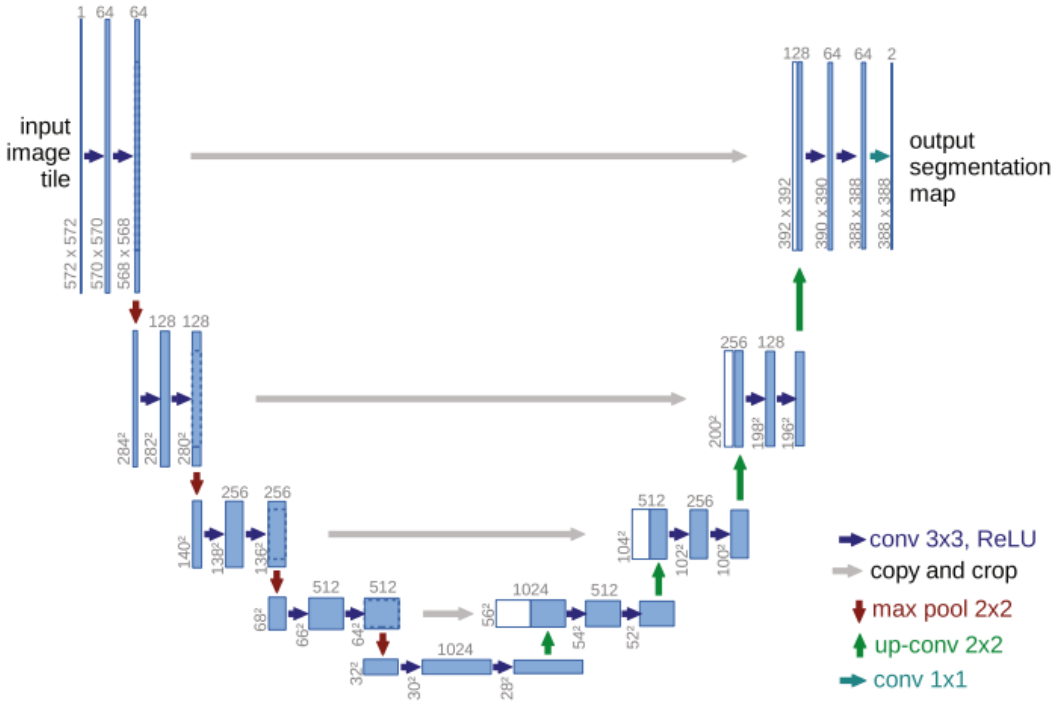


Figura 20. Esquema de la arquitectura U-Net. [34]

2.4.2. Tareas de visión por computador

Las tareas de *computer vision*, o visión por computadora, se refieren a las diversas tareas que las computadoras pueden realizar para ganar una comprensión de imágenes digitales o video. Estas tareas buscan imitar la visión humana utilizando la inteligencia artificial y el aprendizaje automático. A continuación, se describen distintas tareas que se realizan con CNN.

1. **Clasificación de imágenes:** Esta tarea busca asignar una etiqueta o clase a una imagen completa. Un ejemplo sería determinar si una imagen de una radiografía contiene o no signos de una enfermedad específica, o de clasificar la especie de animal que se encuentra en una imagen, como se muestra en la Figura 21. Se muestra que el resultado de la red es gato o *cat*. Esta tarea a menudo utiliza redes neuronales convolucionales CNN para el aprendizaje supervisado. [27]

Figura 20. Ejemplo de clasificación de imágenes. [28]

2. **Detección de objetos:** Esta tarea se basa en localizar la posición de objetos en una imagen. Para cada objeto detectado, el modelo dibuja una caja delimitadora que indica la ubicación del objeto en la imagen y una etiqueta indicando la clase del objeto, como se muestra en la Figura 22. Algunos modelos que realizan esta tarea son los R-CNN, Fast R-CNN, Faster R-CNN y YOLO. [29]

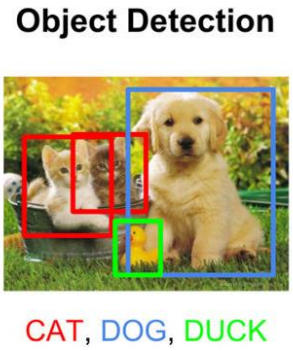


Figura 21. Ejemplo de detección de objetos. [28]

3. **Segmentación semántica:** En la segmentación semántica en lugar de simplemente detectar objetos y proporcionar cajas delimitadoras, el objetivo es clasificar cada píxel en la imagen, es decir, proporcionar una etiqueta de clase para cada píxel. Esto permite la separación detallada de los objetos y el fondo. Un ejemplo visual de los resultados se muestra en la Figura 23. [30]

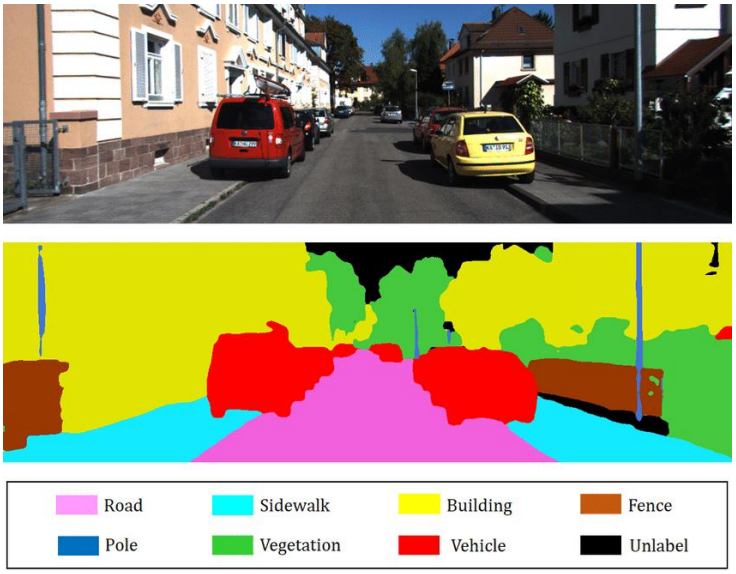


Figura 22. Ejemplo de segmentación semántica. [30]

2.4.2. Modelos de visión por computador

Se menciona a continuación cuatro modelos utilizados comúnmente en las redes neuronales convolucionales en tareas específicas. [29]

1. **Regiones con redes neuronales convolucionales (R-CNN)**: Este modelo utiliza una técnica llamada búsqueda selectiva para proponer un conjunto de posibles regiones de la imagen que podrían contener objetos, estas regiones se llaman regiones propuestas. Luego, para cada región propuesta, se usa una red neuronal convolucional para clasificarla en una de las clases de objetos o como fondo. En la Figura 24 se muestra un ejemplo de las fases ejecutadas por el modelo.

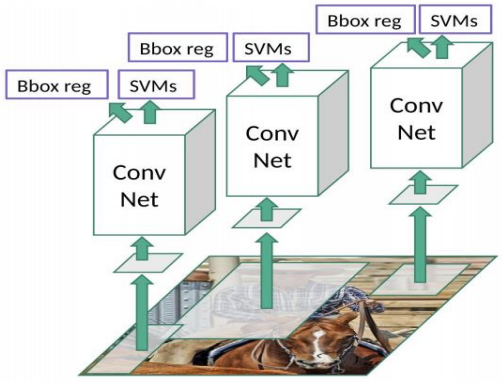


Figura 23. Funcionamiento del modelo R-CNN. [29]

1. **Regiones con redes neuronales convolucionales rápido (*Fast* R-CNN)**: Este es una mejora del R-CNN que es más rápido y eficiente. En lugar de aplicar la CNN a cada región propuesta individualmente, *Fast* R-CNN aplica la CNN a toda la imagen una vez y luego extrae características para cada región propuesta de la salida de la CNN. Esto reduce la cantidad de cálculos necesarios.

2. **Regiones con redes neuronales convolucionales más rápido (*Faster* R-CNN)**: Este modelo va un paso más allá al incluir una red llamada *Region Proposal Network* (RPN) que aprende a proponer las regiones en lugar de usar búsqueda selectiva. Esto hace que el proceso sea aún más rápido y permite que todo el sistema sea entrenado de manera conjunta. Su funcionamiento se observa en la Figura 25.

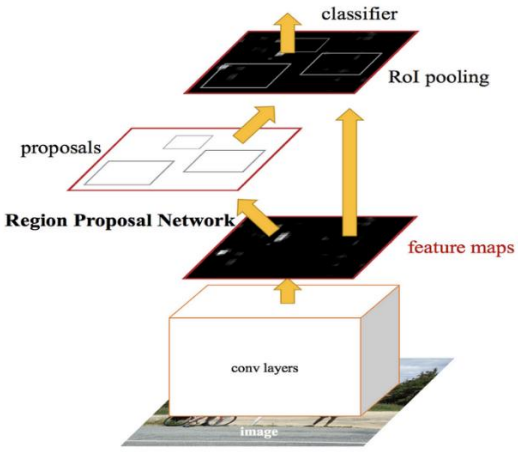


Figura 24. Funcionamiento del modelo *Faster* R-CNN. [29]

3. ***You Only Look Once* (YOLO)**: YOLO, en lugar de proponer regiones y luego clasificarlas, divide la imagen en una cuadrícula y predice para cada celda de la cuadrícula si contiene el centro de un objeto y, de ser así, qué clase de objeto es y cuál es la caja delimitadora del objeto.

Este enfoque es muy rápido y puede funcionar en tiempo real, pero suele ser menos preciso con objetos pequeños o cuando varios objetos están muy cerca el uno del otro. En la Figura 26 se muestra la cuadricula de cada objeto identificado con un diferente color y posteriormente el recuadro que delimita el objeto.

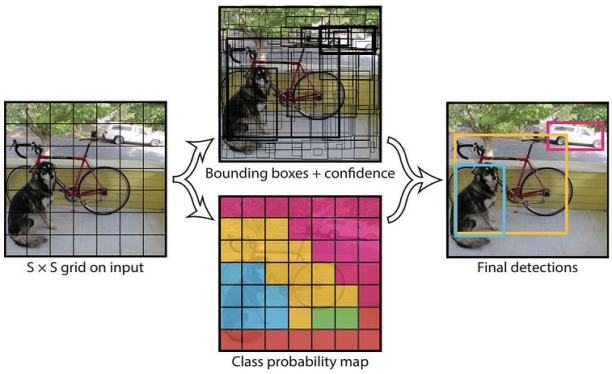


Figura 25. Funcionamiento del modelo YOLO. [29]

4. **Regiones con redes neuronales convolucionales con máscara (*Mask* R-CNN)**: Este modelo es una extensión del Faster R-CNN que agrega una rama para predecir una máscara que indica la forma del objeto, es decir, para cada píxel de la región propuesta, la red predice si el píxel pertenece al objeto. Esta característica hace que Mask R-CNN sea útil no solo para la detección de objetos, es decir, determinar qué objetos están presentes en una imagen y dónde se encuentran, sino también para la segmentación de instancias determinar la forma exacta de cada objeto en la imagen. La habilidad de *Mask* R-CNN para manejar tanto la detección de objetos como la segmentación de instancias ha hecho que sea una herramienta muy poderosa y versátil en la visión por computadora. [32] En la Figura 27 se muestra la máscara de color realizada en cada objeto identificado y sus cajas delimitadoras.

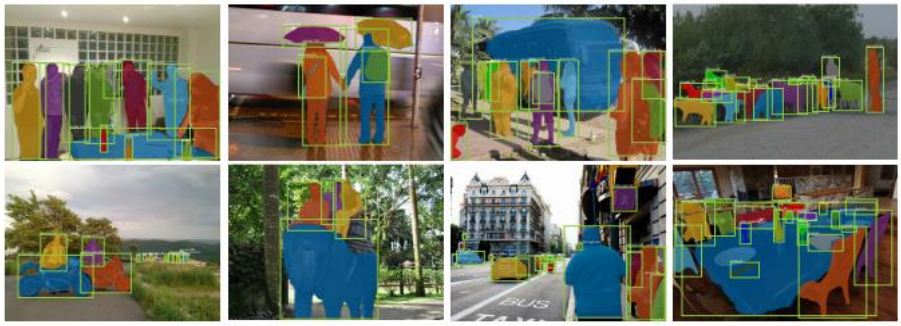


Figura 26. Funcionamiento del modelo *Mask* R-CNN. [32]

2.4.3. Conteo de objetos usando CNN

Existen varias técnicas de aprendizaje automático que pueden ser utilizadas para contar objetos en imágenes. Dependiendo de la tarea a realizar, algunas técnicas pueden ser más adecuadas que otras.

1. **Detección de objetos:** Como se menciona anteriormente, la detección de objetos es una técnica que puede ser utilizada para contar objetos. Modelos como R-CNN, Fast R-CNN, Faster R-CNN y YOLO pueden ser entrenados para detectar y localizar objetos de interés en una imagen. Una vez que los objetos son detectados, se realiza un conteo de estos. [31]

2. **Segmentación de instancias:** La segmentación de instancias no sólo realiza la detección de objetos, también identifica la forma exacta de cada objeto. Como se ha mencionado anteriormente, Mask R-CNN es un ejemplo de un modelo de segmentación de instancias. Al igual que en el ejemplo anterior, una vez que los objetos son segmentados, pueden ser contados. [32]
3. **Estimación de densidad:** Algunos problemas de conteo de objetos, especialmente aquellos con muchos objetos pequeños y cercanos entre sí, pueden ser abordados como problemas de estimación de densidad. La idea es crear un mapa de densidad que representa la densidad de objetos en diferentes partes de la imagen. Los modelos de aprendizaje profundo como las redes convolucionales pueden ser entrenados para predecir estos mapas de densidad a partir de imágenes del cual posteriormente se obtiene una estimación del número de objetos. [33] En la Figura 28 se observa la imagen de validación o *test image* introducida en la red, el mapa de densidad que se corresponde a su etiqueta *o ground truth* y la estimación realizada por la red a partir de la cual se realiza el conteo de los objetos presentes en la imagen.

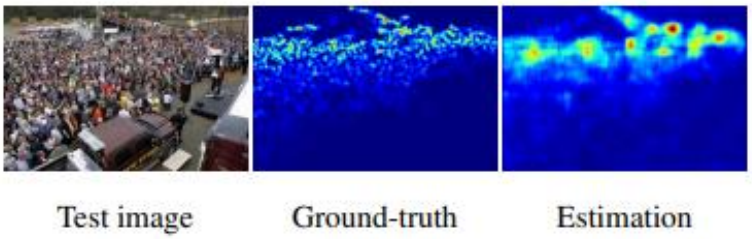


Figura 27. De izquierda a derecha: Imagen de entrada, Mapa de densidad con etiqueta, Mapa de densidad resultante de una red que realiza estimación de densidad. [33]

La elección de la técnica a utilizar dependerá del tipo específico de problema de conteo de objetos que se quiera resolver, así como de las características de las imágenes que se estén utilizando. Por ejemplo, la detección de objetos puede ser más adecuada para imágenes con pocos objetos grandes y claramente delimitados, mientras que la estimación de densidad puede ser más adecuada para imágenes con objetos pequeños.

2.4.4. Medidas de error

Con el fin de evaluar la calidad del modelo, se emplean diversas medidas de error que permiten caracterizar las discrepancias entre las predicciones del modelo y los valores reales. [41] A continuación, se describen las siguientes medidas de error utilizadas en este estudio:

En las ecuaciones (10), (11), (12), (13) y (14) se presentan los parámetros correspondientes a las medidas de error utilizadas en este estudio. En estas ecuaciones, n representa el número total de conteos realizados, y_i denota los valores individuales de etiqueta o referencia (*ground truth*) y $\hat{y}_i$ representa los valores predichos por el modelo para cada instancia i .

Error Absoluto Medio (*Mean Absolute Error*, MAE): Es una medida de la diferencia promedio entre los valores reales y los valores predichos por un modelo. Se calcula como la media de los errores

absolutos. Su valor es relativo al tipo de datos, en este caso su valor en las tablas indica el error al estimar el número de peces en una imagen. La ecuación (10) describe el cálculo del MAE.

$$MAE = \frac{1}{n} \cdot \sum_{i}^{n} (|(y_i - \hat{y}_i)|) \tag{10}$$

Error de Porcentaje Medio Absoluto (*Mean Absolute Percentage Error* , MAPE): Es una medida de la precisión de un modelo y representa el porcentaje promedio de diferencia entre los valores reales y los valores predichos. Se calcula como se muestra en la ecuación (11). Su valor es adimensional.

$$MAPE = \frac{1}{n} \cdot \sum_{i}^{n} \left| \frac{(y_i - \hat{y}_i)}{y_i} \right| \tag{11}$$

Error Cuadrático Medio (*Mean Squared Error,* MSE): Es una medida estadística que se utiliza frecuentemente en aprendizaje automático y en estadística para cuantificar el error entre los valores estimados o predichos por un modelo y los valores reales o verdaderos. En este caso su valor en las tablas indica el error al estimar el número de peces en una imagen. Se calcula como se muestra en la ecuación (12). [43]

$$MSE = \frac{1}{n} \cdot \sum_{i}^{n} (|(y_i - \hat{y}_i)^2|) \tag{12}$$

Raíz del Error Cuadrático Medio (*Root Mean Squared Error,* RMSE): Es una medida de la diferencia entre los valores reales y los valores predichos por un modelo, al cuadrado. En este caso su valor en las tablas indica el error al estimar el número de peces en una imagen. Se calcula como se muestra en la ecuación (13).

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{i}^{n} (|(y_i - \hat{y}_i)^2|)} \tag{13}$$

Error Absoluto: El error absoluto es una medida utilizada para cuantificar la diferencia absoluta entre el valor observado y el valor predicho en un contexto de predicción o estimación. Se calcula tomando la diferencia absoluta entre el valor observado y el valor predicho para cada. Su cálculo se describe en la ecuación (14).

$$Error\ Absoluto = |(y_i - \hat{y}_i)| \tag{14}$$

3. Especificaciones y restricciones de diseño

El objetivo principal de este proyecto es el desarrollo de un sistema basado en redes neuronales que sea capaz de detectar y realizar un conteo de los rodaballos presentes en imágenes RGB sin zoom y a diferentes distancias del tanque a la cámara. Se desea implementar una red neuronal preentrenada para el reconocimiento de personas que ha sido probada en imágenes con zoom previamente. [3]

3.1. Especificaciones de diseño

- El sistema diseñado utiliza redes neuronales convolucionales con técnicas de estimación de densidad.

-El sistema ha sido entrenado únicamente para detección a través de imágenes sin zoom de rodaballos en estado larvario presentes en tanques de cría de peces.

- El sistema estima el número de peces presentes en cada imagen.

- El sistema posee un entrenamiento previo para detección de objetos, así que se hace uso de la técnica de *transfer learning*.

3.2. Restricciones de diseño

− El sistema recibe como entrada imágenes de dimensión de 1024×768 píxeles, provenientes de imágenes originales con dimensión de 2560×1920 píxeles.

− La base de datos de imágenes estará limitada a 19 imágenes de entrenamiento y 4 imágenes de validación. Las imágenes serán las proporcionadas por el Grupo de Aplicaciones Multimedia y Acústica.

− El etiquetado de las imágenes no ha sido realizado por profesionales, por lo tanto, las imágenes con gran nivel de oclusión pueden contener etiquetas erróneas.

− El sistema se desarrollará en lenguaje de programación Python 3.

− El proyecto se desarrolla utilizando librerías de código abierto.

− El sistema debe contar con la capacidad para estimar el número de rodaballos presentes en una imagen.

− El sistema debe ser capaz de realizar el proceso de conteo de peces de manera autónoma, sin necesidad de asistencia humana.

4. Descripción de la solución propuesta

4.1. Selección de modelo a utilizar

Este proyecto busca continuar el desarrollo realizado previamente sobre un conjunto de datos de imágenes RGB de rodaballos en estado larvario con zoom que representaban imágenes parciales del tanque. [3] El modelo utilizado en dicho proyecto ha sido creado para el conteo de personas en imágenes, sin embargo, se ha demostrado su buen funcionamiento para el conteo de peces en la investigación antes mencionada.

A fin de mejorar la precisión en el conteo de las larvas de rodaballo presentes en una imagen de un tanque de piscifactoría se realiza este proyecto, orientado al conteo de imágenes sin zoom y, por tanto, con objetos de menor tamaño.

La red SS-DCNet utiliza el aprendizaje supervisado, por lo que se requiere de características y etiquetas para el entrenamiento y la validación del modelo. A su vez, el conjunto de datos utilizado es reducido, por lo que se busca probar la eficiencia del modelo SS-DCNet que utiliza una arquitectura U-NET, para un conjunto reducido de imágenes de entrenamiento.

4.1.1 Análisis de la SS-DCNet

El conteo visual busca estimar el número de objetos en una imagen, este conteo en teoría puede llegar a ser infinito, pero en la práctica los datos en una imagen son limitados, por lo tanto, las soluciones posibles se encuentran en un conjunto cerrado entre cero y un número ‘‘n’’ [0, n]. Usualmente se utilizan técnicas de regresión para llevar a cabo estos conteos, pero pueden tener poca precisión al encontrarse con escenas que no se han visto antes, especialmente si el conteo se encuentra fuera del rango o conjunto cerrado observado.

La SS-DCNet aprovecha una propiedad del conteo, ser espacialmente descomponible, y divide la región de objetos hasta que los conteos de las subregiones estén dentro del rango o conjunto cerrado de conteo observado. Esta técnica le da su nombre a la red, División y Conquista Espacial (*Spatial Divide and Conquer*) y a través de ella logra generalizarse para manejar escenarios de conjunto abierto, habiendo aprendido de situaciones o imágenes con un conteo de conjunto cerrado.[35] En la Figura 29, se observa la imagen original dividida en cuatro secciones y posteriormente la esquina superior izquierda se ha dividido en cuatro trozos más, para asegurar que cada imagen obtenga un conteo dentro del conjunto cerrado [0, $C_{máx}$].

Figura 28. Divisiones espaciales realizadas en una imagen. [35]

En la Figura 30, se observa el proceso ejecutado a la imagen de origen: La imagen original es aumentada de tamaño con el fin de introducirla a la red para inferencia de la cantidad de objetos en áreas locales. La línea discontinua naranja se utiliza para establecer una conexión entre la sub-imagen (en azul), el mapa de características local (en naranja) y finalmente el conteo local de objetos (los puntos naranjas). En el lado del decodificador, se parte de la imagen cuyas características han sido extraídas y se decodifica con la información del mapa de características obtenido en el paso anterior (cuadrado naranja) y a partir de ambas informaciones se formula un conteo. En resumen, la imagen de entrada se divide espacialmente y se aumenta su tamaño para determinar el conteo de objetos en áreas más pequeñas. Mientras que, en el mapa de características, la técnica divide y conquistarás se utiliza para aumentar su tamaño, decodificarlo y dividirlo con el fin de optimizar los cálculos necesarios.

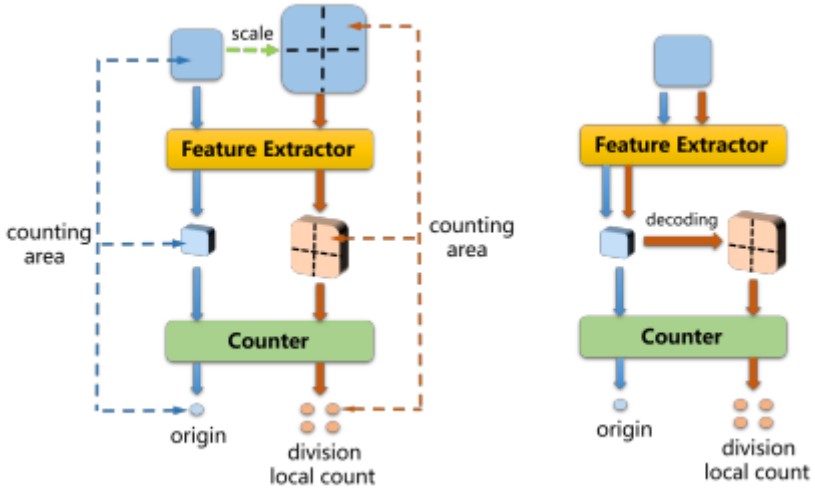


Figura 30. Funcionamiento del codificador (izquierda) y el decodificador (derecha) U-NET. [35]

La red adopta todas las capas convolucionales de la VGG16 y así se simplifican los primeros dos bloques convolucionales, a los que simplemente se hace referencia como "Conv" en la Figura 31 (izquierda). Para dividir y aumentar el mapa de características se utiliza un decodificador similar al de la arquitectura U-NET, mostrado en la Figura 30 (derecha), luego, se generan tres componentes importantes: el mapa de dimensiones aumentadas o *upsampling* ‘‘U_{is}’’, los conteos de división ‘‘C_{is}’’

y las máscaras de división W_{is} Estos son generados por tres componentes de la red: un aumentador de dimensión o *upsampler*, un contador de conjunto cerrado y un decisor de división de imagen. Estos componentes trabajan juntos para decidir cómo dividir la imagen y contar los objetos dentro de estas divisiones.

Una vez que se obtienen estos componentes, se produce un proceso de fusión mostrado en la Figura 31 (derecha). El conteo de la división anterior C_{i-1} se aumenta con el mapa de *upsampling* correspondiente U_i luego se fusiona con el conteo de la división actual C_i utilizando la máscara de división correspondiente W_i. Este proceso da como resultado el conteo de la división i-ésima DIV_i.

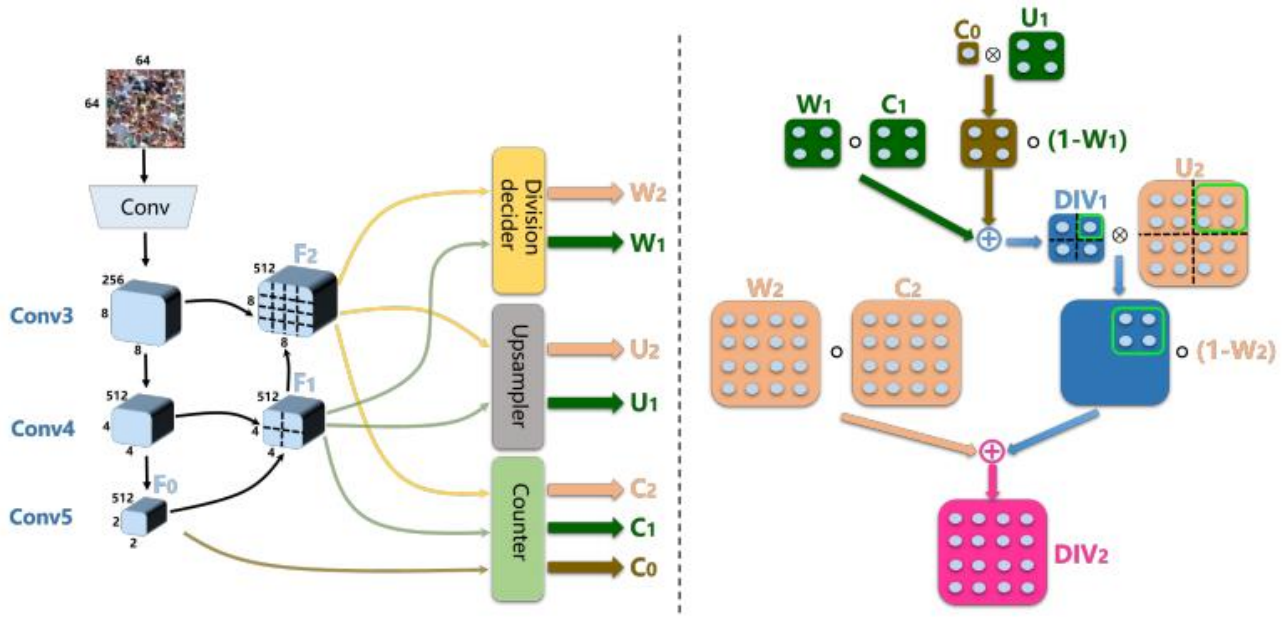


Figura 31. Arquitectura de la SS-DCNet (izquierda) y proceso de división espacial divide y conquistarás (derecha). [35]

La primera división, DIV_i, se calcula utilizando la ecuación (15). En esta usando el producto Hadamard se combina la máscara de división W1, el conteo de la división previa C_0, y el conteo de la división actual C_1. El término M_1 representa una matriz de 1 de la dimensión de W_1 y avg es un operador de redistribución de promedio, que divide igualmente un valor de conteo en una región 2×2 píxeles.

$$DIV_1 = (M_1 - W_1) \circ avg(C_0) + W_1 \circ C_1 \quad (15)$$

La red SS-DCNet también puede realizar múltiples divisiones espaciales al decodificar aún más y dividir el mapa de características, el número máximo de divisiones es 4 en la arquitectura VGG16, pero los experimentos mostraron que una división de dos etapas es suficiente para garantizar un rendimiento satisfactorio. $DIV2$ es el resultado del conteo de objetos después de la segunda división, se calcula de manera recursiva con respecto a $DIV1$, utilizando la Ecuación (16).

$$DIV_2 = (M_1 - W_i) \circ avg(DIV_{i-1}) + W_i \circ C_i \quad (16)$$

Las clases consideradas por el clasificador son generadas con un valor de paso o *step* de 0.5, y comienzan desde cero hasta el valor $C_{máx}$, conteo máximo en espacios de 64×64 píxeles dentro de la imagen. Si un conteo es superior al parámetro $C_{máx}$ de la red, ese espacio de 64×64 píxeles será dividido en subregiones hasta que el conteo de cada subregión esté dentro del conjunto cerrado

[0,$C_{máx}$]. Para los valores dentro del rango [0, 0.5] el paso es de 0.05 esta partición de denomina partición doblemente lineal. [36]

Las funciones de pérdida utilizadas en este modelo son las siguientes:

Pérdida de la fusión (*Merging loss*): Se calcula el error generado en la última división efectuada DIV_i, mediante la función de error medio absoluto.

Pérdida del contador (*Counter loss*): Se calcula el error mediante una función de entropía cruzada y el error total es la suma de los errores de conteo C_i.

Pérdida por división (*Division loss*): Se calcula como se muestra en la ecuación (17) y mide el error generado al asignar a un subsegmento de imagen el valor $C_{máx}$ cuando el valor de conteo real o *ground truth* supera este parámetro, es decir cuando $C_{0,gt} > C_{máx}$.

$$L_{DIV} = -M_1 \cdot (C_{máx}) \cdot \log(\max(W_1)) \quad (17)$$

Pérdida por sobremuestreo (*Upsampling loss*): Se calcula un mapa de submuestreo U_i, que es utilizado para redistribuir ciertos recuentos a una resolución más alta en la red SS-DCNet. Para guiar esta operación de submuestreo y minimizar los errores, se necesita una función de pérdida de submuestreo. El mapa de submuestreo se calcula a partir de la distribución real de los recuentos locales. Y se utiliza una función de pérdida L1 para cada mapa de submuestreo, sumándolas todas para obtener la pérdida total de submuestreo.

Finalmente, la función de pérdida total (*Total loss*) es la suma ponderada de todas las pérdidas calculadas anteriormente.

La SS-DCNet es funcional para conjuntos de imágenes de entrenamiento reducidos ya que utiliza técnicas de **aumento de datos** o *data augmentation* para la creación de nuevas imágenes de entrenamiento a partir de las originales. En particular se generan 9 subimágenes de resolución 256×192 píxeles que representa $\frac{1}{4}$ de la resolución original de la imagen. Las primeras 4 subimágenes generadas corresponden a las cuatro esquinas de la imagen, mientras que las restantes 5 se seleccionan de forma aleatoria. Además, se aplican técnicas de escala y volteado aleatorio (*random horizontal flip)* como parte del proceso de aumento de datos.

A su vez, a cada imagen del conjunto de validación y de entrenamiento se le aplica una **normalización**. Al normalizar las imágenes, se logra una escala consistente en los valores de los píxeles, lo que facilita el entrenamiento y mejora la convergencia del modelo. Además, ayuda a evitar que ciertos píxeles dominen la contribución al cálculo de gradientes durante el proceso de entrenamiento, lo que puede conducir a un mejor rendimiento general del modelo.[39] En este caso se realiza restando a cada píxel la media de los pixeles de toda la imagen y dividiendo entre 255.

Además, el **algoritmo de optimización** utilizado es el SGD combinado con un parámetro de control de velocidad de actualización de pesos (*momentum*) y con una técnica de regularización utilizada en conjunción con algoritmos de optimización. El parámetro ***momentum*** es una medida de la persistencia del impulso en la dirección de actualización de los pesos. Ayuda a acelerar el proceso de convergencia y a superar posibles obstáculos locales en la superficie de pérdida. La técnica es el ***weight decay*** o decaimiento de pesos, se aplica durante el proceso de entrenamiento de un modelo de aprendizaje automático para controlar el sobreajuste (*overfitting*) y mejorar su capacidad de generalización. [39]

4.1.2. Mapas de densidad para el conteo de objetos

El conteo de objetos se solía realizar a través de la detección y localización de instancias individuales de objetos, lo cual conlleva un proceso de aprendizaje largo para la red empleada. Existe una técnica que se basa en estimar la densidad de elementos en una imagen. Esta técnica permite calcular una integral sobre cualquier área seleccionada de la imagen, lo que produce el recuento de elementos presentes en esa región específica. Esta es una forma eficaz de cuantificar y localizar la densidad de elementos en diferentes áreas de una imagen. La estimación del conteo de objetos a través del uso de mapas de densidad es introducida en [38].

El procedimiento de cálculo de los mapas de densidad es el siguiente: se tiene un conjunto de imágenes de entrenamiento y cada grupo de píxeles de estas imágenes está asociado a unas características. Además, cada imagen de entrenamiento ha sido etiquetada con un software desarrollado en el grupo GAMMA. El etiquetado consiste en marcar de forma manual los centros de los rodaballos con un punto de dos dimensiones que representa un objeto, en este caso un rodaballo. Para lograr contar con precisión la cantidad de objetos en una imagen, se utiliza una función de densidad, que asigna valores reales a los píxeles de la imagen y cuya suma en una región de la imagen nos da el recuento de objetos en esa región. En lugar de detectar y localizar objetos individualmente, se estima una función de densidad en cada píxel.

Para aprender de la función de densidad, se utiliza el aprendizaje supervisado. Se ajusta un modelo lineal que transforma las características de cada píxel en una estimación de la función de densidad. En las CNN, se utiliza una función de pérdida para medir la discrepancia entre la función de densidad estimada y la función de densidad real basada en las etiquetas. Esto tiene como objetivo encontrar los mejores parámetros para el modelo lineal que minimicen el error. En resumen, el enfoque propuesto se basa en aprender una función de densidad mediante un modelo lineal que se ajusta a los datos de entrenamiento y utiliza características de los píxeles para estimar la cantidad de objetos en una imagen.

La función de densidad de define como se muestra en la ecuación (18) donde p representa un píxel de la imagen de entrenamiento, P son las etiquetas en 2D de la imagen, σ representa la desviación estándar de la distribución gaussiana y controla el grado de dispersión de los valores alrededor del punto central y $M_{1_{2x2}}$ es una matriz de 1 de dimensión 2×2 píxeles.

$$F_i(p) = \sum_{P \in Pi} \mathrm{N}\left(p; P; \sigma^2 M_{1_{2x2}}\right) \tag{18}$$

La función definida en la ecuación (19) representa el filtro o región gaussiana normalizada utilizada para generar la función de densidad.

$$\mathrm{N}\left(p; P; \sigma^2 M_{1_{2x2}}\right) = \frac{1}{2\pi\sigma^2} e^{\frac{-((px-Px)^2+(py-Py)^2)}{2\sigma^2}} \tag{19}$$

La función calcula la probabilidad de que el punto p pertenezca a la distribución gaussiana, y este cálculo se basa en la distancia entre las coordenadas x e y de p y P. Cuanto más cerca esté p de P y menor sea la varianza σ^2, mayor será la probabilidad. En resumen, el kernel gaussiano 2D normalizado se utiliza para calcular la probabilidad de que un punto en una imagen pertenezca a una distribución gaussiana centrada en otro punto. La selección del valor de la desviación estándar puede realizarse de manera dinámica a través de geometría adaptativa, el sistema se basan en la correlación entre los

tamaños de las cabezas y su respectiva distancia. Por lo tanto, la desviación estándar (σ) se calcula dinámicamente para cada punto etiquetado, tomando como referencia la media de la distancia entre el punto en cuestión y sus vecinos más cercanos, y multiplicándola por un coeficiente reductor.

La implementación actual no tiene en cuenta las variaciones de tamaño que surgen de la conversión de un espacio tridimensional a dos dimensiones, por lo tanto, en este proyecto no se utiliza este procedimiento y el valor de la desviación estándar es fijo. Esta decisión se tomó porque el conjunto de datos específico utilizado en este proyecto no satisface las condiciones para las que se diseñaron las estrategias de geometría adaptativa mencionadas, que involucra cabezas de personas. Además, en esta implementacion de asume que todos los rodaballos se encuentran en el mismo plano y no interfiere en gran medida a la profundidad que estos se encuentren pues la mayoría se encuentran en la superficie.

4.1.3. Implementación de SS-DCNet

Este proyecto es una continuación del desarrollado en [3], en el cual se utiliza de base la implementación de la red SS-DCNet realizada por Dmitry Burdeiny; que ha sido entrenada con el conjunto de imágenes, ImageNet. Este modelo puede ser reentrenado con nuevas imágenes, utilizando la técnica de *transfer learning*. [37]

Los componentes del desarrollo se describen a continuación: El **archivo de configuración** *config_train_val_test.yaml* contiene parámetros asociados a la validación de la red como: $\boldsymbol{C_{máx}}$, y sigma. El archivo **label_count_utils** genera el conjunto cerrado de resultados de conteo del clasificador que va de 0 al parámetro denotado $\boldsymbol{C_{máx}}$: [0, $\boldsymbol{C_{máx}}$]. El fichero **dataset** contiene una clase que recibe los datos de validación y entrenamiento y los transforma en la estructura necesaria para el módulo de entrenamiento y el de inferencia. El archivo **Loss** contiene las funciones para el cálculo de los errores antes mencionados. La implementación de la red convolucional y el clasificador SS-DCNet está contenida en el fichero **SDCNet**. Con el script **Generador de Mapa de densidad,** se obtienen los mapas de densidad del conjunto de imágenes de validación y entrenamiento en formato *.npz* para su posterior uso en los módulos de inferencia y entrenamiento. A partir de los datos generados con el generador de mapa de densidad se entrena la red usando el módulo **Entrenamiento**. El conteo de rodaballos en nuevas imágenes se implementa en el script **Inferencia**. El *learning rate* inicial es de valor 0,0001 y es dividido por un factor de 10 con cada iteración durante el entrenamiento.

Se ha desarrollado dos algoritmos destinados a la estimación de los parámetros del modelo, Cmáx y sigma, basándose en las características inherentes de la imagen. El algoritmo "**obtenerCmax**" elabora un histograma que ilustra los percentiles de la cantidad de rodaballos contenidos en un cuadro de 64x64 píxeles, además del número de veces que esta situación se presenta, procedimiento que se realiza para cada una de las imágenes. Por otro lado, el algoritmo "**obtenerSigma**" genera los mapas de densidad de las imágenes proporcionadas con el objetivo de analizar la dispersión resultante de la variación del parámetro sigma, permitiendo así la selección de un valor óptimo para este parámetro.

Finalmente, se utiliza un código fuente desarrollado previamente durante las prácticas externas realizadas en el CITSEM. En este código se utiliza la SS-DCNet para el conteo de objetos a partir de imágenes con zoom. Las imágenes con zoom son obtenidas a partir de la aplicación de preprocesamiento a imágenes sin zoom. Específicamente se divide la imagen original en 4

subimágenes de resolución igual a ¼ de la imagen original. Posteriormente se aplica interpolación bicúbica para lograr una resolución de 1024×768 píxeles.

4.2. Conjuntos de entrenamiento y validación para el modelo de detección de objetos

Durante el desarrollo de este proyecto se utiliza un conjunto de imágenes reducido ya que las características del modelo de red neuronal utilizado se pueden ajustar usando pocos datos de reentrenamiento. El conjunto de entrenamiento, está formado por 19 imágenes mientras que el conjunto de validación, nombrado como 'conjunto de validación 1' lo conforman 4 imágenes. Todas las imágenes utilizadas han sido capturadas sin zoom. Para comprobar cómo afectan los diferentes niveles de zoom al rendimiento de la red neuronal se generó un segundo conjunto de validación con 10 imágenes. Estas difieren de las del conjunto anterior ya que la distancia de la cámara al tanque es mayor y los objetos se aprecian con un menor tamaño, a este conjunto se le ha llamado 'conjunto de validación 2'. Todas las imágenes han sido obtenidas a partir de videos proporcionados por el CITSEM y como parte de este proyecto han sido etiquetadas las imágenes del conjunto de entrenamiento y de validación 1. Las imágenes pertenecientes al conjunto de validación 2 han sido etiquetadas previamente en prácticas externas realizadas en el CITSEM. Las imágenes de todos los conjuntos de datos a utilizar son RGB y de resolución 2560×1920 píxeles.

Para la realización de las etiquetas se ha utilizado una aplicación de etiquetado desarrollada por el CITSEM codificada en Python. El formato en el que se generan las etiquetas de los conjuntos de validación y entrenamiento es el siguiente: El valor de un contador que empieza desde el número 0 aumenta con cada selección con el ratón hasta el número de centroides de rodaballo etiquetados y sus coordenadas ''x'' e ''y'' en píxeles en un archivo ''.csv''. Se han modificado los módulos de generación de mapas de densidad, y el de inferencia para que no sea necesario ejecutarlos con argumentos en línea de comando y para recibir el nuevo formato de etiquetado o *ground truth.* En la Figura 32, se muestra una imagen que ha sido etiquetada con la aplicación mencionada, cada punto azul es el centroide de un rodaballo y por tanto sus coordenadas son guardadas junto con un contador que lleva la información de cuántos rodaballos se han seleccionado.

Figura 32. Imagen del conjunto de entrenamiento etiquetada.

Por otro lado, se ha desarrollado un script encargado de la generación de conjuntos de datos, el cual realiza una reducción en la resolución de las imágenes, empleando un factor de división de 2.5, la

resolución resultante es de 1024×768 píxeles. Posteriormente, este mismo script también se encarga de ajustar las correspondientes etiquetas. Este proceso tiene como objetivo principal minimizar la cantidad de cuadros de 64×64 píxeles que el sistema produce durante el conteo de objetos, lo que no solo acelera el proceso, sino que también ayuda a evitar la sobre-segmentación. Este script asegura la coherencia en el nombramiento de los archivos, modificando tanto las imágenes como los archivos ‘‘.csv’’ para que compartan el mismo nombre. Sin embargo, se añade un prefijo ‘‘GT_’’ a los archivos ‘‘.csv’’, facilitando su identificación como el *ground truth* correspondiente a cada imagen en cuestión.

Además, se ha adaptado el módulo generador de mapas de densidad para que extraiga la información de las etiquetas requerida para la construcción de los mapas de densidad a partir de un archivo con extensión ‘‘.csv’’. Dicho archivo alberga un número entero ‘‘n’’ en la primera columna, que representa la cantidad de objetos, seguido por dos números en las columnas sucesivas que especifican la posición ‘‘x’’ y ‘‘y’’, respectivamente. En contraposición, la estructura previa requería de un archivo de extensión ‘‘.mat’’ que contenía la misma información. Se han realizado mejoras significativas al código de inferencia. Se ha incorporado el cálculo de errores y la función de generar mapas de calor para cada estimación. Además, se ha introducido el uso de un archivo de configuración para definir algunos parámetros en lugar de utilizar la línea de comandos. Esta última modificación también se ha aplicado al código generador de mapas de densidad, lo que ha mejorado la flexibilidad y facilidad de uso de ambos componentes del sistema.

A continuación, en la Figura 33, se presentan algunas imágenes del conjunto de entrenamiento con distintas densidades de peces. Las imágenes están ordenadas de izquierda a derecha, de menor a mayor densidad de peces.

Figura 33. Imágenes con distinta densidad de peces del conjunto de entrenamiento.

En la Figura 34, se muestran tres imágenes del conjunto de validación 1. El nombre de las imágenes sigue la siguiente estructura ‘‘L03T01-20230419110544-20230419111144’’ donde ‘‘L03’’ se refiere al Larvario 3, ‘‘T01’’ se refiere al Tanque 1, ‘‘2023.04.19 11:05:44’’ fecha y hora de inicio del vídeo, ‘‘2023.04.19 11:11:44’’ fecha y hora final del vídeo. Las imágenes están ordenadas de izquierda a derecha, de menor a mayor densidad de peces.

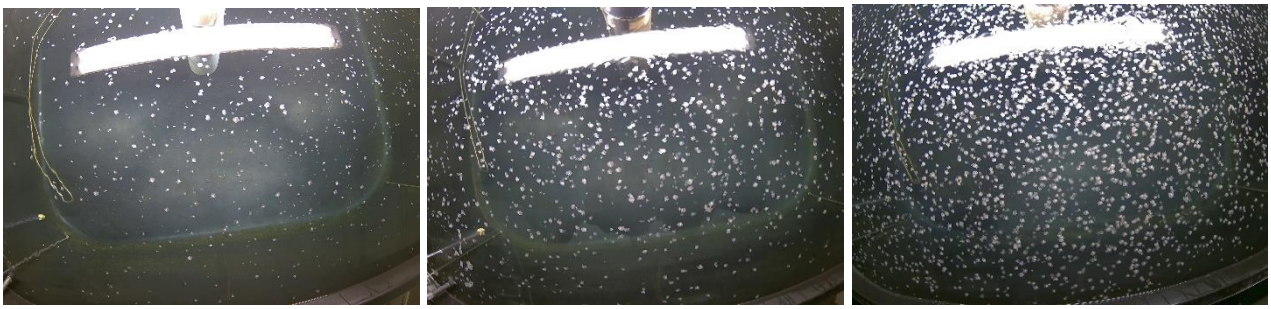


Figura 34. Imágenes con distinta densidad de peces del conjunto de validación 1.

A continuación, en la Figura 35, se muestran tres imágenes del conjunto de validación 2. Las imágenes están ordenadas de izquierda a derecha, de menor a mayor densidad de peces. Es importante destacar que en la imagen de la izquierda es la ‘‘L04T02-20220106053533-20220106054033’’, en esta imagen se observan los peces con un menor tamaño en comparación con las imágenes del conjunto de entrenamiento. Esto será interesante a la hora de analizar los resultados.

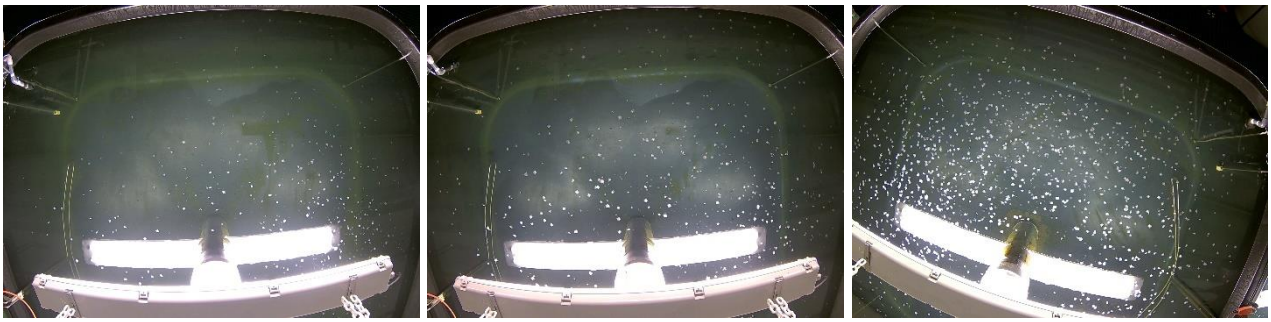


Figura 35. Imágenes con distinta densidad de peces del conjunto de validación 2.

4.3. Parámetros de la SS-DCNet

4.2.1. Desviación estándar

En este desarrollo se utiliza un valor de desviación estándar, σ o sigma, fijo en la creación de mapas de densidad, este valor determina la escala o alcance de la distribución del mapa de densidad. En otras palabras, afectará la suavidad y la difusión de la densidad alrededor de cada punto u objeto en la imagen.

Si se elige una desviación estándar pequeña, la distribución de la densidad será más puntual y concentrada alrededor de cada punto, lo que resultará en una mayor precisión y resolución en la estimación del conteo de objetos. Sin embargo, esta distribución puntual puede ser más sensible al ruido y a las variaciones locales en la imagen, lo que podría llevar a una mayor sensibilidad a los falsos positivos o negativos.

Por otro lado, si se elige una desviación estándar más grande, la distribución de la densidad será más suave y difusa alrededor de cada punto, abarcando un área más amplia. Esto puede ayudar a mitigar el efecto del ruido y las variaciones locales, y hacer que la estimación del conteo sea más robusta. Sin

embargo, una desviación estándar demasiado grande puede difuminar demasiado la densidad y conducir a una pérdida de detalles finos en la estimación. [40]

En la Figura 36, se muestra los mapas de densidad obtenidos del conjunto de imágenes de entrenamiento, usando tres valores distintos de sigma. Se puede ver que al aumentar de valor el alcance de cada distribución aumenta.

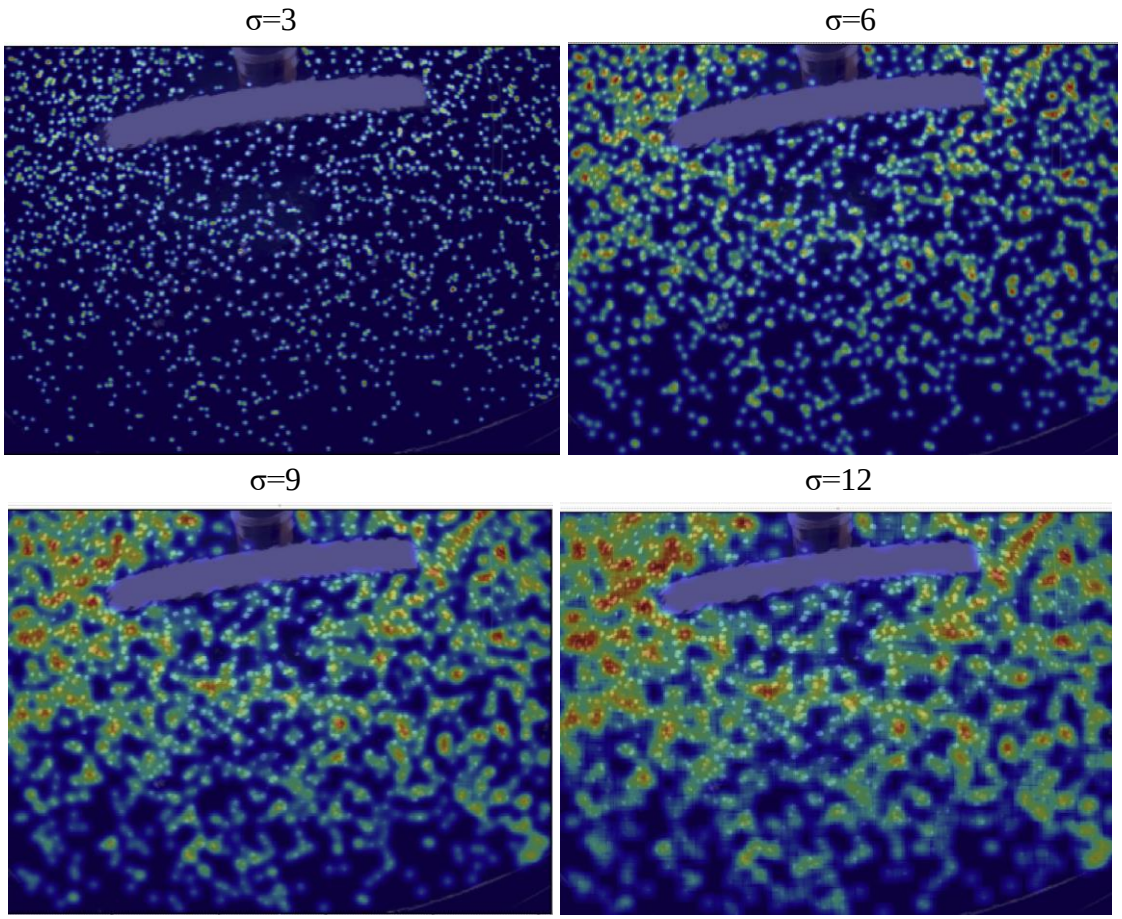


Figura 36. Mapa de densidad obtenido de los siguientes valores de sigma (σ).

4.2.2. Dimensión del clasificador

En el clasificador se generan las clases consideradas como conjunto de solucion con un paso o *step* de 0.5 y abarcan un rango desde cero hasta un valor máximo denominado $C_{máx}$, el cual representa el conteo máximo permitido en espacios de 64×64 píxeles dentro de la imagen. Si se obtiene un conteo que supera el valor de $C_{máx}$ establecido por la red, el espacio de 64×64 píxeles se divide en subregiones más pequeñas hasta que el conteo de cada una de estas subregiones esté dentro del rango cerrado [0, $C_{máx}$]. Este proceso garantiza que los conteos obtenidos estén dentro de los límites establecidos por el modelo de clasificación. Si $C_{máx}$ es demasiado grande, el modelo puede tener dificultades para distinguir entre diferentes cantidades de objetos, lo que resulta en predicciones imprecisas. Por otro lado, si $C_{máx}$ es muy pequeño, el modelo puede tener una capacidad limitada para contar correctamente los objetos, lo que resulta en una pérdida de información importante debido a la incapacidad para capturar la variabilidad en la cantidad de objetos en las imágenes.

Con el fin de determinar el valor máximo del clasificador a utilizar en el modelo se realiza el gráfico mostrado en la Figura 37. Este refleja las instancias en las que se ha observado una cantidad específica de objetos dentro de un espacio de 64×64 píxeles en las imágenes del conjunto de datos.

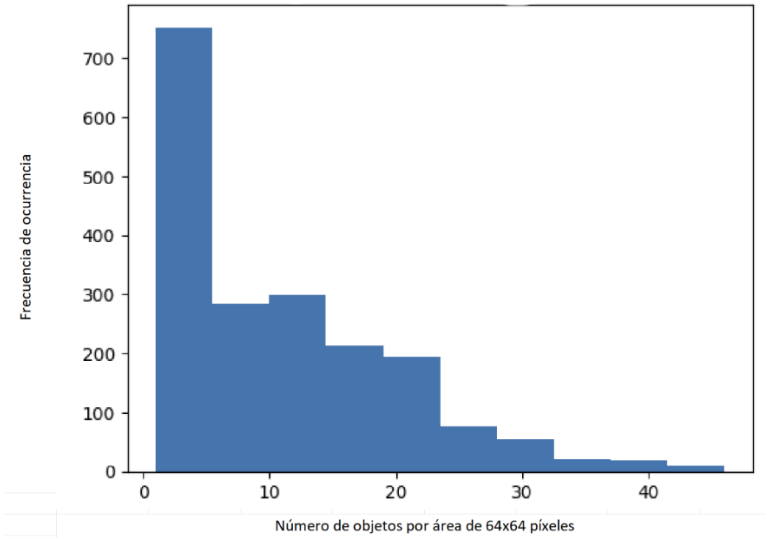


Figura 37. Número de objetos por área de 64×64 píxeles vs Frecuencia de ocurrencia.

La Tabla 1 presenta la distribución de la cantidad de objetos observados en un espacio de 64×64 píxeles, mostrando las instancias correspondientes a través del percentil.

Tabla 1. Percentiles de número de objetos en espacio de 64x64 píxeles obtenidos del conjunto de datos.

Número de Objetos	Percentil [%]
28	95
18	75
9	50
3	25

Se llevarán a cabo diversas pruebas con el objetivo de determinar el valor de $C_{máx}$ produce mejores resultados.

5. Resultados

5.1. Resultados obtenidos variando la desviación estándar

Los resultados mostrados en la Tabla 3, reflejan los valores de error obtenidos del conteo con modelos entrenados con el valor de $C_{máx}$ fijo a 18 y distintos valores para la desviación estándar. Los datos indican que los errores mínimos se logran utilizando una desviación estándar igual a 9.

Tabla 3. Valores de error obtenidos para distintos valores de desviación estándar.

Desviación estándar	MAE	MSE	MAPE [%]
2	195,72	232,37	12,19
5	185,15	225,97	10,86
9	135,81	174,4	7,83
10	147,94	188,73	8,37
11	136,59	172,96	8,15

En la Figura 38, se observan distintos mapas de calor obtenidos del conteo de objetos en la misma imagen del conjunto de validación 1 al utilizar distintos valores de desviación estándar. El mejor resultado obtenido, según el cálculo de los errores mostrado en la Tabla 3, es el que se observa en la Figura 38 Centro Izquierda, el cual se corresponde con un valor de desviación estándar 9. En la figura con valor de σ = 2, se puede apreciar mayor existencia de píxeles o puntos amarillos respecto al mejor resultado obtenido. Esto sugiere que en estos puntos se está cometiendo una sobreestimación en el conteo de peces. Lo mismo sucede en las figuras de valores σ = 5 y 10. Por el contrario, en la figura de valor de sigma σ = 11 se observa menos cantidad de píxeles amarillos, por lo tanto, se puede suponer que al utilizar dicho valor de sigma ocurre una subestimación en el conteo de los peces.

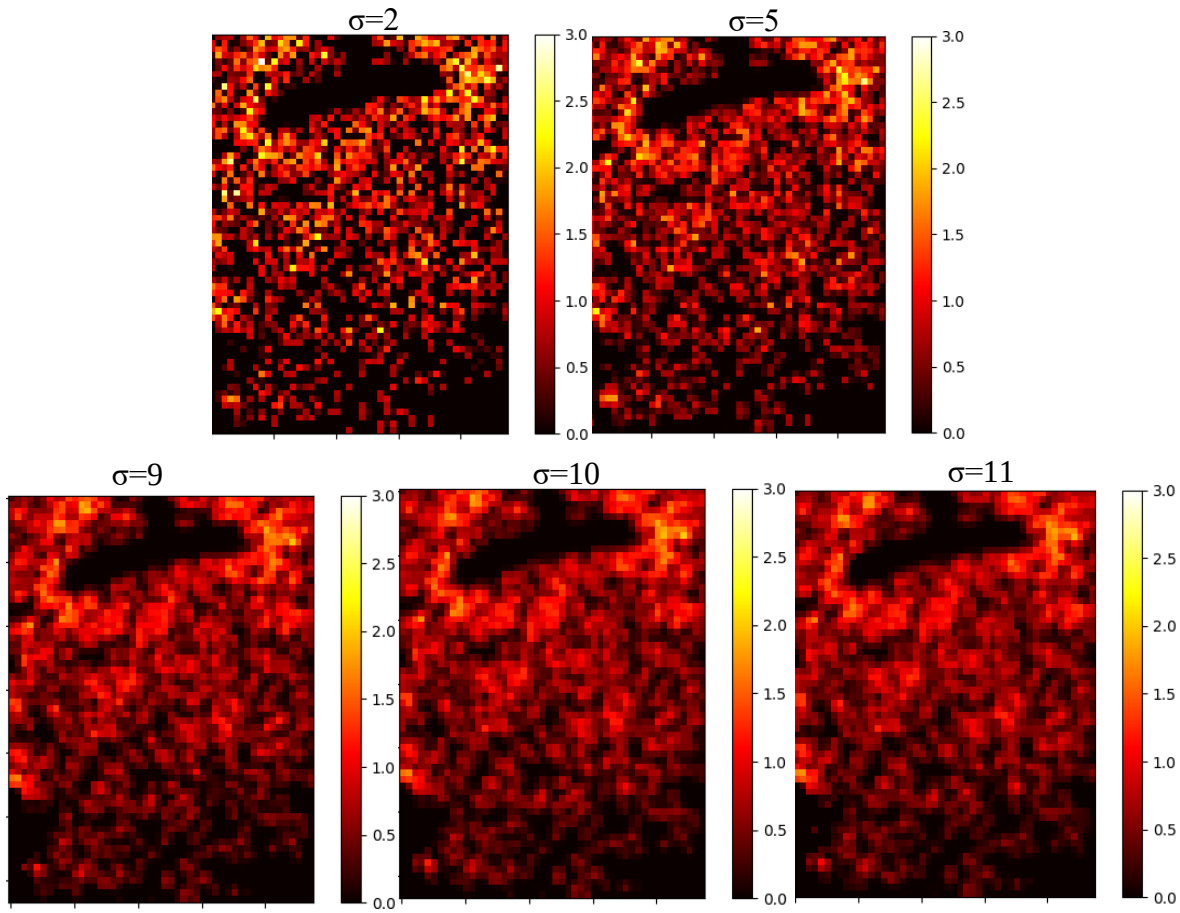


Figura 38. Mapa de calor obtenido de los siguientes valores de desviación estándar (σ).

5.2. Resultados obtenidos variando la dimensión del clasificador $C_{máx}$

En la Tabla 4, se observa que los errores de conteo de los modelos entrenados con una desviación estándar (σ) fija de 9 varían dependiendo del valor de la dimensión del clasificador $C_{máx}$. Se observa que los errores más bajos se obtienen cuando se utiliza un $C_{máx}$ de 18

Tabla 4. Valores de error obtenidos para distintos valores $C_{máx}$.

$C_{máx}$	MAE	MSE	MAPE [%]
17	206,53	256,55	11,85
18	135,81	174,40	7,83
20	144,14	187,21	8,00
28	161,64	202,32	9,45

En la Figura 39, se observan distintos mapas de calor obtenidos del conteo de objetos en la misma imagen del conjunto de validación 1 al utilizar distintos valores de $C_{máx}$ y el mismo valor de desviación estándar 9. La escala de colores presentada en los mapas de calor representa la cantidad de rodaballos identificados en una seccion de 16×16 píxeles de la imagen. Aquí, el color negro señala la ausencia de

rodaballos, mientras que el blanco indica la detección de tres rodaballos en un solo píxel. Los píxeles con tonos amarillos sugieren una detección de rodaballos entre 2,3 y 2,9. Así, una mayor densidad de rodaballos en una sección específica se reflejará mediante tonalidades más claras, como el amarillo y blanco. El mejor resultado obtenido, según el cálculo de los errores mostrado en la Tabla 4, es el que se observa en la figura que se corresponde con un valor de $C_{máx}$=18. En la figura de valor de $C_{máx}$=17, se puede apreciar mayor existencia de píxeles o puntos amarillos respecto al mejor resultado obtenido. Esto sugiere que en estos puntos se está cometiendo una sobreestimación en el conteo de peces. Lo mismo sucede al utilizar un $C_{máx}$= 28. Por el contrario, en la figura de $C_{máx}$=20 se observa menos cantidad de píxeles amarillos, pero también mayor cantidad de píxeles rojos por lo tanto no se aprecia claramente la diferencia de utilizar un $C_{máx}$ de 20 y uno de 18, ya que el conteo no varía en gran medida.

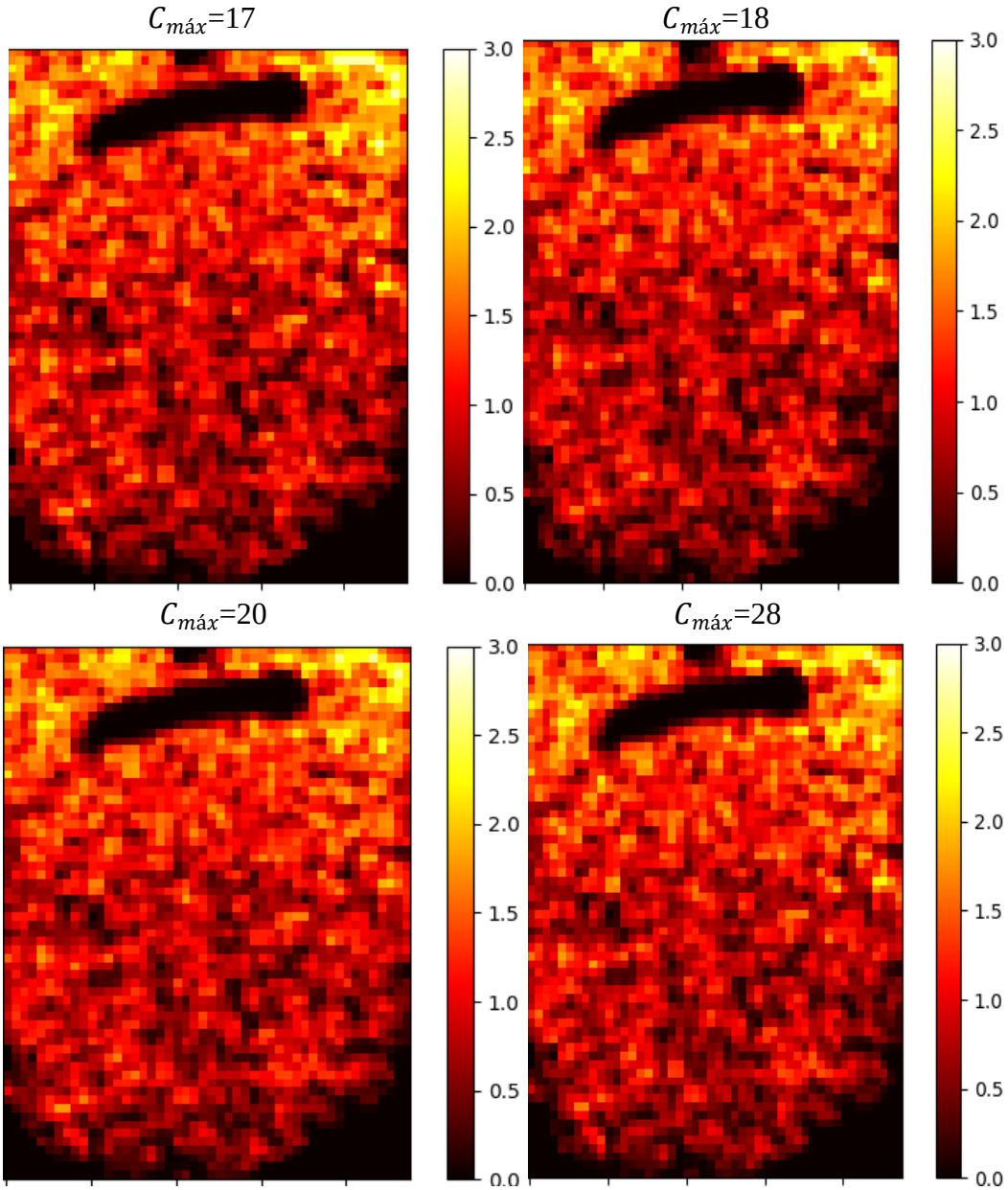


Figura 39. Mapa de calor obtenido de los siguientes valores $C_{máx}$.

5.3. Procedimiento de entrenamiento

Una vez determinados los valores óptimos para la desviación estándar 9 y la dimensión del clasificador 18, se procede a realizar un nuevo entrenamiento. La arquitectura utilizada, VGG-16, se encuentra pre-entrenada en el conjunto de imágenes ImageNet y se utiliza como base para este proyecto. Además, la SS-DCNet ofrece la ventaja de poder ser re-entrenada utilizando un modelo previamente entrenado, aprovechando el *checkpoint* generado en etapas anteriores.

Para garantizar que el modelo no está alcanzando un punto de sobreajuste durante el re-entrenamiento, se considera el gráfico de la función de pérdida total. En la Figura 40 se puede observar que, a pesar de las fluctuaciones en los errores, la tendencia general de la curva es descendente. Esto indica que el modelo continúa aprendiendo en este punto y no ha mostrado signos de sobreajuste al finalizar el reentrenamiento. Aunque el valor ideal de la función de pérdida es cero, lo que indicaría que las predicciones del modelo se ajustan perfectamente a los valores reales, alcanzar un valor de cero en la práctica es raro e inclusive indeseable. Por lo tanto, en lugar de buscar un valor específico para la función de pérdida, lo más común es tratar de minimizarla tanto como sea posible y mirar cómo evoluciona durante el entrenamiento.

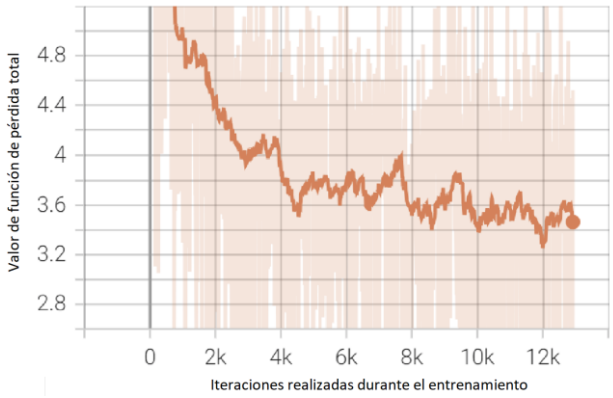


Figura 40. Función de pérdida total en función de las iteraciones realizadas en el entrenamiento

Las Figuras 41, 42 y 43 presentan los errores calculados para un conjunto de validación aleatorio que la red neuronal utiliza durante el proceso de entrenamiento ilustrado en la Figura 40. El valor del MAE se calcula durante el proceso de entrenamiento y se registra en función de la época. Como se muestra en la Figura 41, se observa una disminución consistente del MAE a partir de la época 150, alcanzando un valor mínimo de 43,00 peces al cabo de 200 épocas. Este patrón sugiere que el modelo ha aprendido eficazmente a minimizar el error durante este período de entrenamiento. El *chekpoint* asociado a este momento específico del entrenamiento puede ofrecer los resultados de estimación más precisos, ya que el modelo ha logrado reducir significativamente el error. Dado que este valor es el más bajo alcanzado durante el entrenamiento, se selecciona este modelo como el más óptimo.

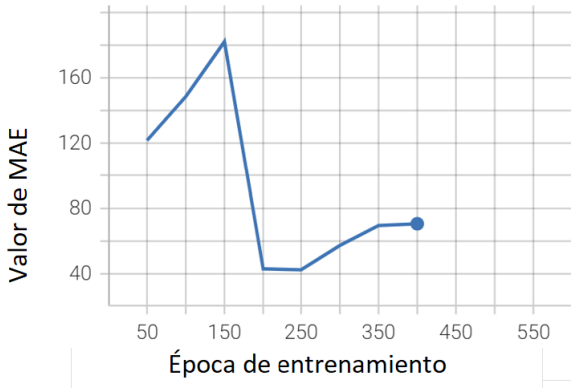


Figura 41. Valor de MAE en función del número de época entrenada.

En la Figura 42, se representa el gráfico de error RMSE, en el cual se puede apreciar una disminución constante de los errores a lo largo de las épocas de entrenamiento. Estos resultados indican que el modelo está mejorando su desempeño y logrando reducir el RMSE alcanzando un valor de 58,91 peces.

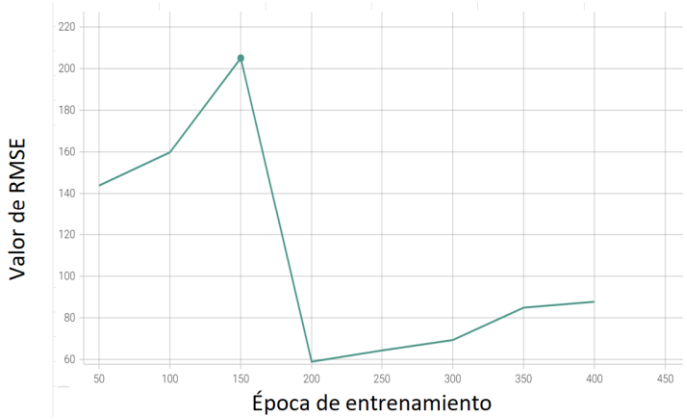


Figura 42. Valor de RMSE en función del número de época entrenada.

Las fluctuaciones del MAPE son mostradas en la Figura 43.El valor del MAPE también disminuye a partir de la época 150, alcanzando finalmente un valor de 1,68%. Esto indica que el modelo continúa mejorando su capacidad de predecir de manera precisa al reentrenarse.

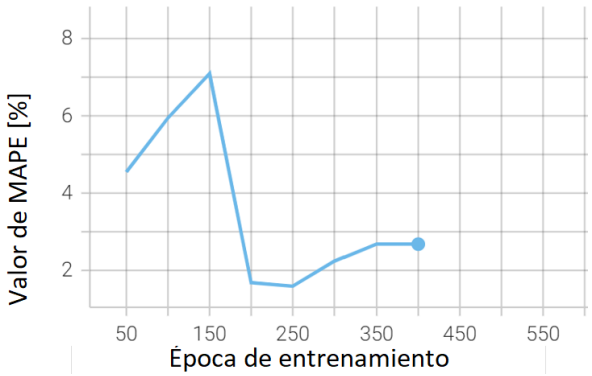


Figura 43. Valor de MAPE en función del número de época entrenada.

5.4. Análisis comparativo de los resultados de los distintos modelos SS-DCNet utilizando imágenes sin zoom.

A continuación, se analiza la influencia de los parámetros de la red, específicamente la desviación estándar y el valor de $C_{máx}$, en la SS-DCNet. En el modelo generado en el proyecto [3] que llamaremos ‘‘modelo con zoom’’, las imágenes utilizadas para el entrenamiento presentaban peces de mayor tamaño por ser capturadas con zoom, lo que resultó en la elección de un valor de $C_{máx}$ de 5 en el clasificador y un valor de 12 para la desviación estándar. En contraste, en este proyecto se utilizaron imágenes sin zoom, lo cual requiere un valor de $C_{máx}$ significativamente mayor y un valor menor de desviación estándar para evitar que se pierda información de los peces más pequeños de las imágenes sin zoom.

Se exhiben en la Tabla 7, los resultados de errores obtenidos para el conjunto de validación 1, utilizando el modelo generado en este proyecto ‘‘modelo sin zoom’’ y el ‘‘modelo con zoom’’. Por otro lado, el modelo sin zoom obtiene el mejor resultado para el conjunto de validación 1 con el valor de desviación estándar igual a 9 y un valor de $C_{máx}$ de 18. Es importante destacar que el conjunto de validación 1 y el conjunto de entrenamiento se obtienen mediante la captura de imágenes a la misma distancia de la cámara al tanque. Por lo tanto, los tamaños de los peces visualizados en estas imágenes son similares:

Tabla 7. Resultados obtenidos de diferentes modelos para el conjunto de validación 1.

Conjunto de validación 1			
Modelo	**RMSE**	**MAE**	**MAPE [%]**
Modelo sin zoom	72,93	55,50	3,48
Modelo con zoom	361,17	308,97	22,64

La utilización de una desviación estándar de 12 en el modelo con zoom ha resultado en un difuminado excesivo de la densidad y la pérdida de detalles finos en la estimación. Esto ha llevado a una subestimación de los valores en cada imagen y ha contribuido a errores en la predicción. Anteriormente se comentaba que la desviación estándar de 11 también genera una subestimación de la cantidad de peces en la imagen, por lo que es de esperarse que el valor 12 también lo haga.

En la Tabla 8 se muestra los resultados de estimación de peces obtenidos con el ‘modelo con zoom’ para cada imagen del conjunto de validación 1, y se comprueba que los valores estimados son menores a los valores reales etiquetados. El mayor error ocurre en la imagen con mayor densidad de peces, que suele presentar oclusión entre los objetos. Por otro lado, se muestran los resultados obtenidos con el ‘modelo sin zoom’ y se observa que se encuentra sobreestimando en todos los casos.

Tabla 8. Resultados de estimación obtenidos con el modelo sin zoom para el conjunto de validación 1.

Imagen	Modelo sin zoom	Modelo con zoom	Valor real
L05T12-20230410123447-20230410124047	714,42	423,10	709,00
L03T01-20230419110544-20230419111144	1368,44	1131,69	1348,00
L03T02-20230419111659-20230419112259	1581,29	1337,26	1455,00
L03T05-20230419121137-20230419121737	2431,86	1746,06	2362,00

En la Tabla 9 se evidencia que tanto el modelo sin zoom como el modelo con zoom presentan los errores más altos en la imagen con la mayor densidad de peces. Este resultado indica que, en general,

las imágenes con una mayor densidad de peces, que suelen incluir la oclusión de objetos, resultan en estimaciones de conteo poco precisas. Esta observación pone de manifiesto la dificultad que representa contar de manera precisa los peces en imágenes con una alta densidad y presencia de obstrucciones. Estos factores pueden afectar la capacidad del modelo para detectar y contar adecuadamente los peces individuales.

Tabla 9. Error absoluto obtenido de ambos modelos para el conjunto de validación 1.

Imagen	Error absoluto modelo sin zoom	Error absoluto modelo con zoom
L05T12-20230410123447-20230410124047	5,42	285,90
L03T01-20230419110544-20230419111144	20,44	216,31
L03T02-20230419111659-20230419112259	126,29	117,74
L03T05-20230419121137-20230419121737	69,86	615,94

En la Tabla 10 se muestran los resultados obtenidos para el conjunto de validación 2 en los modelos de prueba. Aunque las imágenes del conjunto de validación 2 se capturaron sin zoom, la distancia de la cámara a la superficie del tanque es mayor que en el conjunto de validación 1, lo que implica que el tamaño aparente de los peces en estas imágenes sea menor. El resultado del 'modelo sin zoom' revela el error generado al utilizar un modelo entrenado con imágenes que presentan peces de un tamaño ligeramente mayor al utilizado para la validación, es decir diferentes. Por otro lado, el error mostrado en el 'modelo con zoom' revela el error provocado al utilizar un modelo entrenado con imágenes que presentan peces de un tamaño significativamente mayor al utilizado para la validación.

Tabla 10. Resultados obtenidos de ambos modelos para el conjunto de validación 2.

Conjunto de validación 2			
Modelo	**RMSE**	**MAE**	**MAPE [%]**
Modelo sin zoom	196,00	130,88	16,82
Modelo con zoom	526,28	482,38	55,39

En la Tabla 11 se presentan los resultados al aplicar ambos modelos en la estimación del número de peces en cuatro imágenes del conjunto de validación 2, con densidades variadas de peces. En todas las imágenes, se observa un error de subestimación, el cual es mayor para el modelo con zoom, lo que es de esperarse ya que los parámetros $C_{máx}$y desviación estándar en ese modelo se eligieron para objetos de un mayor tamaño.

Tabla 11. Resultados de estimación para cuatro imágenes del conjunto de validación 2 usando cada modelo.

Imagen	Modelo sin zoom	Modelo con zoom	Valor real
L04T02-20220106053533-20220106054033	330,99	168,60	421,00
L04T02-20220109000501-20220109001002	462,70	242,28	532,00
L04T08-20220105153136-20220105153636	940,28	472,57	993,00
L04T08-20220108220211-20220108220711	1225,45	781,49	1294,00

En la Tabla 12, se puede apreciar que el mayor error absoluto se produce en el modelo sin zoom al evaluar imágenes con menor densidad de peces y peces de menor tamaño, como se ilustra en la Figura 35. Este error puede ser atribuido al hecho de que en el conjunto de entrenamiento no se han incluido imágenes con objetos de tamaño tan reducido. Por lo tanto, el modelo carece de experiencia en la

detección y conteo preciso de peces de menor tamaño, lo que resulta en un mayor error absoluto en estas imágenes.

Por otro lado, para el modelo con zoom, se mantiene la observación de que los errores más altos se encuentran en las imágenes con una mayor densidad de peces. Esto indica que el modelo también enfrenta dificultades al estimar con precisión los conteos en situaciones de alta densidad, donde la oclusión y la superposición de peces pueden complicar la tarea de detección y conteo.

Tabla 12. Error absoluto obtenido de ambos modelos para el conjunto de validación 2.

Imagen	Error absoluto modelo sin zoom	Error absoluto modelo con zoom
L04T02-20220106053533-20220106054033	90,01	252,40
L04T02-20220109000501-20220109001002	69,30	289,72
L04T08-20220105153136-20220105153636	52,72	520,43
L04T08-20220108220211-20220108220711	68,55	512,51

Estos resultados muestran la importancia de tener conjuntos de entrenamiento diversificados y representativos, que abarquen una amplia gama de densidades y tamaños de peces. De esta manera, el modelo puede aprender patrones y características relevantes para una estimación precisa, incluso en escenarios desafiantes con diferentes condiciones de densidad y tamaño de peces.

Es interesante destacar que el conjunto de entrenamiento del modelo sin zoom está limitado de 19 imágenes. A pesar de esto, los resultados para el conjunto de validación 1, que presenta similitudes con las imágenes de entrenamiento debido a que estas han sido tomadas con la misma distancia de cámara a tanque, han mostrado un mejor rendimiento en comparación con el conjunto de validación 2. Este último conjunto contiene imágenes sin zoom pero con peces de menor tamaño.

Esto sugiere que el modelo sin zoom ha logrado generalizar adecuadamente a las imágenes de validación 1 debido a la similitud en las condiciones de zoom. Sin embargo, el rendimiento del modelo se ve afectado en el conjunto de validación 2, donde los peces son de menor tamaño. Este punto subraya la necesidad crucial de poseer un conjunto de entrenamiento variado y representativo. Debería incluir imágenes de peces de diferentes tamaños y bajo diferentes niveles de zoom. Esto mejorará la habilidad del modelo para generalizar y rendir efectivamente en una variedad más amplia de escenarios.

5.5. Evaluación comparativa de los resultados de los distintos modelos SS-DCNet utilizando imágenes con zoom.

Las imágenes con zoom empleadas en este proyecto han sido obtenidas tras la aplicación de un preprocesamiento. Con el fin de obtener imágenes de la misma resolución que se ha utilizado previamente como entrada de la red en el proyecto de referencia, es necesario dividir las imágenes originales de 2560×1920 píxeles en cuatro imágenes más pequeñas. Posteriormente, las imágenes divididas generadas son interpoladas con el fin de escalarlas a 2560×1920 píxeles nuevamente y así simular imágenes originales sin zoom. El algoritmo utilizado es la interpolación bicúbica. Posteriormente se obtiene la estimación del número de peces para cada una de las 4 imágenes pequeñas y se suman para obtener el total de peces en una imagen completa. El proceso descrito anteriormente forma parte del código fuente desarrollado en prácticas externas en el CITSEM. Se observa en la Figura 44 un ejemplo de imagen original del conjunto de validación 1.

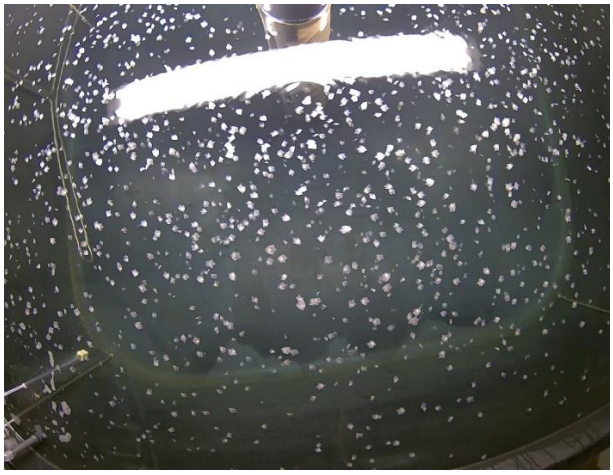


Figura 44: Imagen original perteneciente al conjunto de validación 1.

En la Figura 45 se observa el resultado obtenido al realizar la interpolación bicúbica para cada uno de los cuadrantes de la imagen original.

Figura 45: Imagen dividida en cuatro cuadrantes.

A continuación, se muestran los resultados de evaluación del 'modelo con zoom' y del 'modelo sin zoom'. Además, se evalúan ambos modelos utilizando el conjunto de validación 1 con zoom, el cual consiste en imágenes originalmente sin zoom que han sido preprocesadas para simular el efecto de zoom.

Se puede observar que los resultados obtenidos al aplicar el 'modelo con zoom' presentan errores significativos. Esto puede atribuirse al hecho de que el modelo no ha sido entrenado con imágenes con el mismo nivel de zoom que las utilizadas en esta evaluación. Sin embargo, es interesante notar que el error obtenido con el modelo sin zoom sobre las imágenes interpoladas del conjunto de validación 1, es mayor que el obtenido con las imágenes originales de este conjunto. Este incremento en el error puede ser debido a las interpolaciones o preprocesamientos aplicados a las imágenes, junto con el hecho de que el modelo no ha sido entrenado para estimar sobre imágenes con zoom. Por otro lado, se observa que el error del 'modelo con zoom' es considerablemente mayor al utilizar imágenes con zoom simulado respecto a las que tienen zoom original. Las imágenes con zoom original y las imágenes con zoom simulado pueden tener diferencias en las características visuales y estructurales. El modelo

puede ser más adecuado para capturar y estimar correctamente en las imágenes sin zoom debido a la alineación de características y detalles presentes en este conjunto de imágenes. En resumen, para ambos modelos puede haber dificultades para generalizar y adaptarse a las imágenes con zoom simulado debido a la falta de imágenes de este tipo durante el entrenamiento. Es necesario un entrenamiento más extenso o una adaptación específica para manejar eficazmente este tipo de imágenes.

Tabla 13. Resultados obtenidos de diferentes modelos para el conjunto de validación 1 con zoom simulado.

Conjunto de validación 1 con zoom simulado			
Modelo	**RMSE**	**MAE**	**MAPE [%]**
Modelo sin zoom	202,24	165,18	14,63
Modelo con zoom	717,51	637,31	40,84

En la Tabla 14 se muestra los resultados de estimacion de peces obtenidos con el 'modelo con zoom' para cada imagen del conjunto de validación 1 con zoom simulado, y se comprueba que los valores estimados son mayores a los valores reales etiquetados, lo mismo sucede con el 'modelo sin zoom' y se observa que se encuentra sobreestimando en todos los casos.

Tabla 14. Resultados de estimación obtenidos para cuatro imágenes del conjunto de validación 1 con zoom simulado usando cada modelo.

Imagen	Modelo sin zoom	Modelo con zoom	Valor real
L05T12-20230410123447-20230410124047	990,46	903,17	709
L03T01-20230419110544-20230419111144	1381,10	1844,24	1348
L03T02-20230419111659-20230419112259	1390,02	2230,00	1455
L03T05-20230419121137-20230419121737	2080,82	3445,84	2362

Los errores absolutos obtenidos para cada imagen del conjunto de validación 1 con zoom simulado se encuentran en la Tabla 15. En ambos modelos, se observa que los errores son mayores en imágenes con alta densidad de peces. Esto puede ser atribuido a la presencia de oclusión entre los objetos en la imagen, lo cual dificulta la estimación precisa. Los modelos pueden sobreestimar la cantidad de peces en imágenes con alta densidad y oclusión, lo cual se refleja en los errores más altos.

Tabla 15. Error absoluto obtenido de ambos modelos para el conjunto de validación 1.

Imagen	Error absoluto modelo sin zoom	Error absoluto modelo con zoom
L05T12-20230410123447-20230410124047	281, 18	194,17
L03T01-20230419110544-20230419111144	33,10	496,24
L03T02-20230419111659-20230419112259	64,98	775,00
L03T05-20230419121137-20230419121737	281, 46	1083,84

En la Tabla 16 se muestran los resultados al aplicar ambos modelos sobre el conjunto de validación 2, el cual consiste en imágenes originalmente sin zoom que han sido preprocesadas para simular el efecto de zoom. Para el modelo sin zoom se puede observar que el error ha aumentado respecto al obtenido con conjunto de validación 1 original.

Por otro lado, el error mostrado por el modelo sin zoom revela las limitaciones al utilizar un modelo entrenado con imágenes sin zoom y evaluarlo en imágenes con zoom. De manera similar, el error del

modelo con zoom revela las dificultades al utilizar un modelo entrenado con imágenes con zoom y evaluarlo en imágenes que son significativamente diferentes. Estos resultados resaltan la importancia de entrenar y evaluar los modelos en conjuntos de datos que sean representativos de las condiciones reales de uso. Además, demuestran que los modelos pueden verse afectados por cambios en las condiciones de captura de las imágenes, como la presencia o ausencia de zoom.

Tabla 16. Resultados obtenidos de diferentes modelos para el conjunto de validación 2 con zoom simulado.

Conjunto de validación 2 con zoom simulado			
Modelo	**RMSE**	**MAE**	**MAPE**
Modelo sin zoom	159,36	146,97	16,51
Modelo con zoom	374,29	328,84	36,14

De acuerdo con los resultados de la estimación de peces presentados en la Tabla 17, se puede observar una tendencia a la sobreestimación en el conteo de peces en ambos modelos. Esto significa que los modelos tienden a estimar un número mayor de peces en comparación con el conteo real. Esto puede deberse a que los parámetros de la SS-DCNet no han sido elegidos específicamente para el tamaño de objetos presentes en las imágenes.

Tabla 17. Resultados de estimación obtenidos para cuatro imágenes del conjunto de validación 2 con zoom usando cada modelo.

Imagen	Modelo sin zoom	Modelo con zoom	Valor real
L04T02-20220106053533-20220106054033	501,66	644,68	421,00
L04T02-20220109000501-20220109001002	714,85	713,55	532,00
L04T08-20220105153136-20220105153636	1120,43	1419,12	993,00
L04T08-20220108220211-20220108220711	1535,57	1834,94	1294,00

Los errores absolutos correspondientes a cada imagen del conjunto de validación 2 con zoom simulado se presentan en la Tabla 18. Se observa que, en ambos modelos, los errores son mayores en las imágenes que contienen una mayor densidad de peces. Este fenómeno puede atribuirse a la presencia de oclusión entre los objetos en la imagen, lo cual dificulta la estimación precisa del conteo de peces. Estos resultados también resaltan la limitación de los modelos en la capacidad de lidiar con la oclusión y la necesidad de desarrollar enfoques más avanzados para abordar este desafío en el conteo de peces.

Tabla 18. Error absoluto obtenido de ambos modelos para el conjunto de validación 2.

Imagen	Error absoluto modelo sin zoom	Error absoluto modelo con zoom
L04T02-20220106053533-20220106054033	72,80	223,69
L04T02-20220109000501-20220109001002	176,96	181,55
L04T08-20220105153136-20220105153636	148,29	426,12
L04T08-20220108220211-20220108220711	214,94	540,94

6. Impacto del Proyecto

Las redes neuronales y el aprendizaje automático abren nuevas oportunidades para la sostenibilidad. Este proyecto presenta una alternativa no intrusiva a la práctica vital y necesaria del conteo de peces realizada en acuicultura. La introducción de un método de conteo de bajo coste como el descrito en el proyecto, que funciona sin la ayuda de un humano puede contribuir con el alcance de los siguientes Objetivos de Desarrollo Sostenible (ODS) planteados por las Naciones Unidas en la agenda 2030 [42]:

ODS 2 - **Hambre cero**: Al utilizar la inteligencia artificial para el conteo de peces, se puede mejorar la eficiencia de las piscifactorías, aumentar la producción de peces, lo cual podría ayudar a abordar problemas de inseguridad alimentaria y hambre.

ODS 8 - **Trabajo decente y crecimiento económico**: El proyecto podría generar empleos, tanto en el diseño y mantenimiento de los sistemas de conteo como en las piscifactorías que los utilicen. Además, al aumentar la eficiencia de las piscifactorías, se puede mejorar su rentabilidad y contribuir al crecimiento económico.

ODS 9 - **Industria, innovación e infraestructura**: Un sistema automatizado de conteo de peces es una innovación significativa en la industria acuícola que puede promover el desarrollo tecnológico y la infraestructura resiliente.

ODS 12 **- Producción y consumo responsables**: La tecnología de redes neuronales puede ayudar a gestionar mejor los recursos de las piscifactorías y minimizar el desperdicio, promoviendo patrones de consumo y producción más sostenibles.

ODS 14 - **Vida submarina**: Al usar métodos no invasivos se promueve el cultivo de peces en lugar de la pesca excesiva en los océanos, los sistemas de conteo de peces pueden ayudar a proteger la vida marina.

ODS 15 - **Vida de ecosistemas terrestres**: Al optimizar el uso de recursos acuáticos, se pueden disminuir las presiones sobre los ecosistemas terrestres que actualmente son los más utilizados en la agricultura para la alimentación de millones de personas.

Cabe resaltar que el uso de inteligencia artificial y tecnología en la acuicultura también podría contribuir al ODS 17 - **Alianzas para lograr los objetivos**, ya que se podrían requerir colaboraciones entre las siguientes disciplinas y sectores: tecnología, biología marina, la industria acuícola y las organizaciones de conservación.

7. Presupuesto

A continuación, se describe los recursos utilizados durante la realización del proyecto y su precio, dichos recursos se corresponden a lo material y a la mano de obra requerida durante el desarrollo del proyecto.

El presupuesto expuesto a continuación ha sido calculado teniendo en cuenta que la fase de investigación ha sido de 70 horas de trabajo durante dos meses y la fase de desarrollo de 250 horas durante 3 meses y medio, dando un total de 320 horas empleadas en la realización de este proyecto. Se muestra en la Tabla 19 el presupuesto general.

Tabla 19: Presupuesto General del Proyecto.

Activo	Precio unitario [€]	Unidades	Precio total [€]
Personal de investigación por hora	20	1	1400
Personal de desarrollo por hora	20	1	5000
Ordenador Razer Blade 16	3600	1	3600
Pantalla HUAWEI	120	1	120
Licencia de Windows 11	20	1	20
Presupuesto Total			10140

Para la realización de este proyecto se utiliza de base de datos imágenes provenientes del centro de investigación y herramientas de código abierto, la utilización de ambos no acarrea costos de ningún tipo y por ello no se incluyen en el presupuesto.

La amortización lineal de los equipos informáticos en un período de vida útil de 4 años se muestra en la Tabla 20.

Tabla 20: Amortización de los equipos informaticos durante el período de realización del proyecto.

Activo	Precio unitario [€]	Amortización al año [€]	Amortización para el proyecto [€]	Precio total al finalizar proyecto [€]
Ordenador Razer Blade 16	3600	900	262,5	3337.5
Pantalla HUAWEI	120	30	8,75	111,25

8. Conclusiones

Tradicionalmente, el conteo de peces se realiza de forma manual y visual, lo cual requiere la intervención humana y es laborioso. En este proyecto, se ha utilizado el enfoque de aprendizaje automático para abordar este desafío de manera eficiente y no intrusiva. Se ha implementado la red neuronal SS-DCNet, previamente entrenada para la detección de objetos con el conjunto de entrenamiento ImageNet, y se ha aplicado el método de *Transfer Learning* para adaptarla al conteo de larvas de rodaballo. En particular, las redes neuronales convolucionales han demostrado una notable capacidad para extraer información visual a partir de imágenes y vídeos. En este proyecto, se ha aprovechado el potencial de las redes neuronales convolucionales para abordar el desafío del conteo de rodaballos en imágenes RGB con diferentes niveles de zoom en tanques de piscifactorías.

Los resultados obtenidos respaldan la viabilidad y precisión del modelo SS-DCNet para el conteo de poblaciones de peces. El sistema desarrollado ha demostrado su capacidad autónoma para estimar el número de rodaballos en cada imagen, sin requerir intervención humana, con un error de 3,48% en el caso del modelo sin zoom aplicado sobre el conjunto de validación 1, y un error de 14,63% para el conjunto de validación 1 con zoom simulado. Estos resultados representan una mejora significativa en comparación con los métodos tradicionales, ya que reducen tanto el tiempo como la complejidad del proceso de conteo. Además, se destaca la capacidad del modelo para lograr buenos resultados incluso con un número limitado de imágenes de entrenamiento, incluido los casos de imágenes con diferentes niveles de zoom, aunque se reconoce que la precisión puede verse comprometida ligeramente en estos escenarios. Los resultados favorables en la estimación corroboran que los algoritmos diseñados para determinar los parámetros del modelo, basándose en las características de la imagen, han demostrado su eficacia.

El modelo entrenado en este proyecto ha demostrado su capacidad para generalizar durante el proceso de aprendizaje, incluso al enfrentarse a imágenes ligeramente diferentes al conjunto de entrenamiento. En el caso del conjunto de validación 2, se obtuvo un error de 16,4%, mientras que para el conjunto de validación 2 con zoom simulado se obtuvo un error de 16,51%.

La principal fuente de error identificada en el estudio ha sido la limitación del modelo en su capacidad para manejar situaciones de oclusión. Además, se observó una dependencia significativa en los parámetros de desviación estándar y $C_{máx}$ de la red SS-DCNet. Estos parámetros tuvieron un impacto significativo en el rendimiento del modelo con zoom cuando se trabajó con ambos conjuntos de validación, lo que resultó en un aumento considerable del error. Estos resultados resaltan la importancia de ajustar adecuadamente estos parámetros y considerar sus efectos en la precisión del modelo al trabajar con imágenes con zoom.

En conclusión, el desarrollo de técnicas de aprendizaje automático, como la implementada en este proyecto, ofrece soluciones eficientes y no intrusivas para el conteo de poblaciones de peces en la acuicultura. El modelo SS-DCNet ha demostrado ser una herramienta precisa y confiable en la estimación del número de rodaballos. Su aplicación puede tener un impacto positivo en la industria pesquera y en la preservación de los ecosistemas marinos, mejorando su eficiencia y sostenibilidad al facilitar la gestión adecuada de los recursos pesqueros.

En futuras líneas de trabajo, se recomienda expandir considerablemente los conjuntos de entrenamiento y validación, tanto para imágenes sin zoom como para imágenes con zoom. Esto permitirá mejorar el rendimiento de los modelos en ambos escenarios y obtener estimaciones más precisas. Sin embargo, es importante tener en cuenta que la precisión puede variar entre los diferentes

tipos de imágenes, en función de los parámetros seleccionados, como el valor de C_{max} y la desviación estándar. Por lo tanto, se sugiere explorar enfoques específicos para abordar cada escenario y optimizar el rendimiento en cada uno de ellos. Además, se recomienda realizar un etiquetado preciso de las imágenes, especialmente en casos de alta oclusión, para evitar etiquetas erróneas que puedan afectar la capacidad de los modelos para realizar conteos precisos. Esta atención cuidadosa a los conjuntos de datos y los parámetros de la red contribuirá a mejorar la calidad y la confiabilidad de las estimaciones de población de peces en futuros trabajos de investigación.

Referencias Bibliográficas

[1] Y. H. Y. D. Daoliang Li, «Nonintrusive methods for biomass estimation in aquaculture with emphasis on fish: a review,» *Rev. Aquac.*, vol. vol. 12, p. p. 1390–1411, 2019.

[2] Y. T. K. M. a. K. K. P. Sthapit, «Algorithm to Estimation Fish Population using Echosounder in Fish Farming Net,,» de *International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, Korea (South), 2019.

[3] J. M. G. A. A. González Fernández, «Conteo de objetos en imágenes RGB con redes convolucionales mediante división espacial. Proyecto fin de grado.,» Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, Madrid, 2022.

[4] M. G. a. R. G. Valliappa Lakshmanan, Practical Machine Learning for Computer Vision, O'Reilly Media, Inc., 2021.

[5] B. Mahesh, «Machine learning algorithms: A review,» *International Journal of Science and Research (IJSR)*, vol. 9, pp. 381-386, 2020.

[6] Y. Z. K. K. D. a. Q. W. H. U. Dike, «Unsupervised Learning Based On Artificial Neural Network: A Review,» de *IEEE International Conference on Cyborg and Bionic Systems (CBS)*, Shenzhen, China, 2018.

[7] J. M. Heras, «iartificial,» 29 09 2020. [En línea]. Available: https://www.iartificial.net/clasificacion-o-regresion/. [Último acceso: 13 06 2023].

[8] M. M. G. &. E. E. Khaled Fawagreh, «Random forests: from early developments to recent advancements,» *Systems Science & Control Engineering*, vol. 2:1, nº DOI: 10.1080/21642583.2014.956265, pp. 602-609, 2014.

[9] R. Pupale, «Towards Data Science,» Medium, 16 06 2018. [En línea]. Available: https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989. [Último acceso: 13 06 2023].

[10] «Machine Learning,» IBM Cloud Education, 2022. [En línea]. Available: https://www.ibm.com/es-es/topics/neural-networks. [Último acceso: 11 Junio 2023].

[11] P. Baheti, «Activation Functions in Neural Networks,» v7labs, 05 27 2021. [En línea]. Available: https://www.v7labs.com/blog/neural-networks-activation-functions. [Último acceso: 14 06 2023].

[12] D. M. M. R. Álvaro M. Montenegro D., «Fundamentos de IA y AP,» 2022. [En línea]. Available: https://aprendizajeprofundo.github.io/Libro-Fundamentos/Redes_Neuronales/Cuadernos/Activation_Functions.html. [Último acceso: 13 06 2023].

[13] P. S. Naru, «Understanding Activations & Optimization- Neural Networks,» IBM, 1 02 2023. [En línea]. Available: https://community.ibm.com/community/user/ai-datascience/blogs/pavan-saish-naru/2023/01/27/optimization-hyperparameters. [Último acceso: 14 06 2023].

[14] B. S. López, «Funcion de pérdida de calidad - Taguchi,» Ingenieria Industrial, 29 10 2019. [En línea]. Available: https://www.ingenieriaindustrialonline.com/gestion-de-calidad/funcion-de-perdida-de-calidad-taguchi/. [Último acceso: 2023 06 15].

[15] A. W. T. Y. J. M. T. O. Jonathon Price, «Stochastic gradient descent,» Cornell University, 2020. [En línea]. Available: https://optimization.cbe.cornell.edu/index.php?title=Stochastic_gradient_descent. [Último acceso: 15 06 2023].

[16] M. Molina, «Redes neuronales,» 30 05 2023. [En línea]. Available: https://www.cienciasinseso.com/redes-neuronales/. [Último acceso: 15 06 2023].

[17] T. B. Data, «Optimización de descenso de gradiente de Code Adam desde cero,» 12 01 2021. [En línea]. Available: https://topbigdata.es/optimizacion-de-descenso-de-gradiente-de-code-adam-desde-cero/. [Último acceso: 17 05 2023].

[18] I. C. Education, «Convolutional Neural Networks,» IBM, 2020. [En línea]. Available: https://www.ibm.com/topics/convolutional-neural-networks. [Último acceso: 15 06 2023].

[19] D. Calvo, «Red Neuronal Convolucional CNN,» Diego Calvo, 20 07 2017. [En línea]. Available: https://www.diegocalvo.es/red-neuronal-convolucional/. [Último acceso: 15 06 2023].

[20] N. A, «¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador,» Aprende Machine Learning, 29 11 2018. [En línea]. Available: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/. [Último acceso: 15 06 2023].

[21] S. K. Basaveswara, «CNN Architectures, a Deep-dive,» Towards Data Science, 27 08 2019. [En línea]. Available: https://towardsdatascience.com/cnn-architectures-a-deep-dive-a99441d18049. [Último acceso: 19 06 2023].

[22] R. Shanmugamani, Deep Learning for computer vision, O'reilly: Packt Publishing, 2018.

[23] T. Thaker, «VGG 16 Easiest Explanation,» Medium, 8 08 2021. [En línea]. Available: https://medium.com/nerd-for-tech/vgg-16-easiest-explanation-12453b599526. [Último acceso: 19 06 2023].

[24] S. K. y. Z. A, «Very deep convolutional networks for large-scale image recognition,» de *3rd International Conference on Learning Representations (ICLR)*, 2015.

[25] W. L. Y. J. P. S. S. R. D. A. D. E. V. V. A. R. Christian Szegedy, «Going deeper with convolutions,» *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA*, nº 10.1109/CVPR.2015.7298594., pp. 1-9, 2015.

[26] X. Z. S. R. J. S. Kaiming He, «Deep Residual Learning for Image Recognition,» *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA*, nº 10.1109/CVPR.2016.90, pp. 770-778, 2016.

[27] Stanford, «Convolutional Neural Networks for Visual Recognition,» Course Stanford, 2020. [En línea]. Available: https://cs231n.github.io/classification/. [Último acceso: 20 06 2023].

[28] «A Beginner's Guide to Object Detection,» Data camp, 2018. [En línea]. Available: https://www.datacamp.com/tutorial/object-detection-guide. [Último acceso: 20 06 2023].

[29] R. Gandhi, «R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms,» 9 07 2018. [En línea]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. [Último acceso: 19 06 2023].

[30] T. AI, «Semantic Segmentation: A Complete Guide,» 6 10 2021. [En línea]. Available: https://towardsai.net/p/l/machine-learning-7. [Último acceso: 19 06 2023].

[31] D. J. D. T. M. J. Girshick R, «Rich feature hierarchies for accurate object detection and semantic segmentation,» de *Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587)*, 2014.

[32] K. He, G. Gkioxari, P. Dollár y R. Girshick, «Mask R-CNN,» *IEEE International Conference on Computer Vision (ICCV)*, nº 10.1109/ICCV.2017.322, pp. 2980-2988, 2017.

[33] D. Z. S. C. S. G. Y. M. Yingying Zhang, «Single-Image Crowd Counting via Multi-Column Convolutional Neural Network,» *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA*, nº 10.1109/CVPR.2016.70., pp. 589-597, 2016.

[34] P. F. y. T. B. Olaf Ronneberger, «U-Net: Convolutional Networks for Biomedical Image Segmentation,» *Medical Image Computing and Computer-Assisted Intervention – MICCAI*, vol. 9351, nº 10.1007/978-3-319-24574-4_28, p. 234–241, 2015.

[35] H. L. C. L. L. L. C. S. Z. C. Haipeng Xiong, «From Open Set to Closed Set: Supervised Spatial Divide-and-Conquer for Object Counting,» *Cornell University*, nº 10.48550/arXiv.2001.01886, 2020.

[36] H. L. C. L. L. L. C. S. Z. C. Haipeng Xiong, «From Open Set to Closed Set: Counting Objects by Spatial Divide-and-Conquer,» *IEEE/CVF International Conference on Computer Vision (ICCV)*, nº 10.1109/ICCV.2019.00845, pp. 8361-8370, 2019.

[37] D. Burdeiny, «Unofficial Pytorch implementation of S-DCNet and SS-DCNet,» Github, 17 05 2022. [En línea]. Available: https://github.com/dmburd/S-DCNet.

[38] A. Z. Victor Lempitsky, «Learning To Count Objects in Images,» de *Advances in Neural Information Processing Systems 23 (NIPS)*, 2010.

[39] Y. B. a. A. C. Ian Goodfellow, «Deep learning,» *The MIT Press*, vol. 19, nº 10.1007/s10710-017-9314-z, pp. 305-307, 2018.

[40] C. Zhang, H. Li, X. Wang y X. Yang, «Cross-scene crowd counting via deep convolutional neural networks,» *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, nº 10.1109/CVPR.2015.7298684., pp. 833-841, 2015.

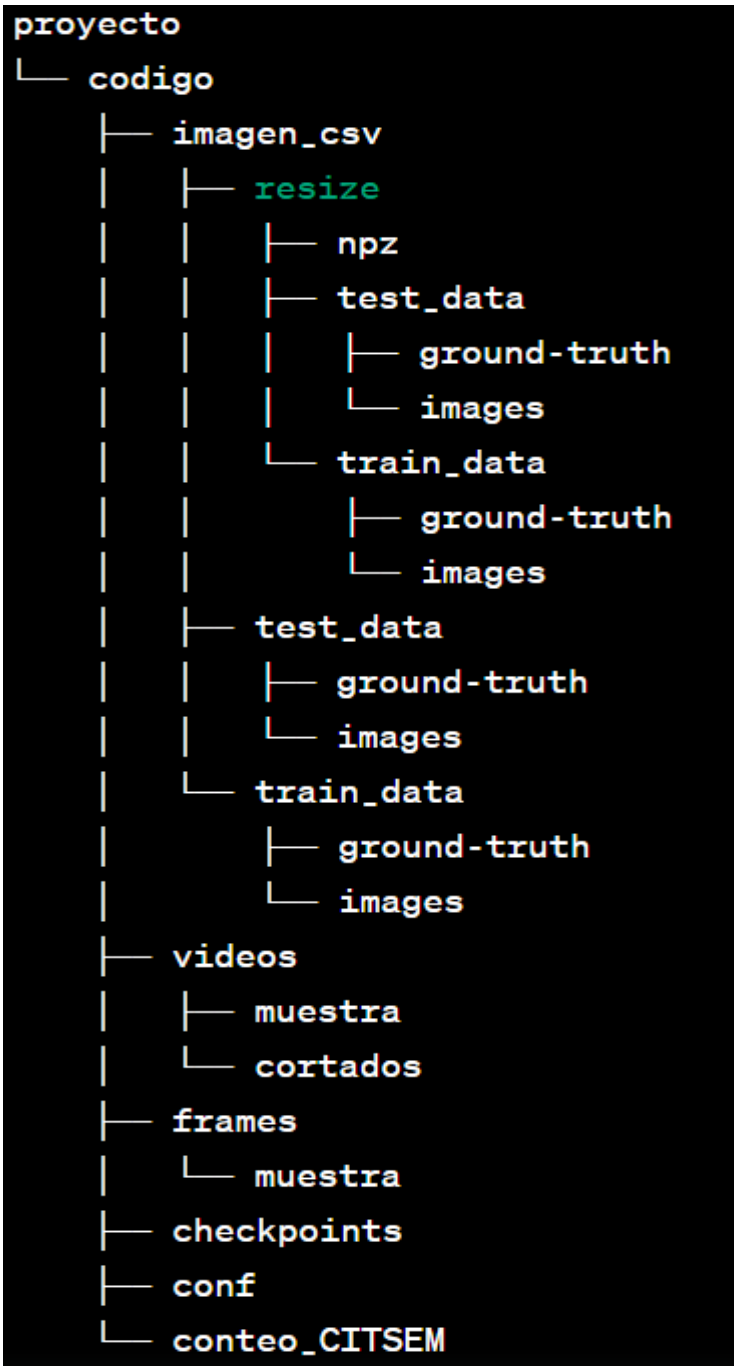
[41] N. Vandeput, «Forecast KPIs: RMSE, MAE, MAPE & Bias,» Towards Data Science, 5 07 2019. [En línea]. Available: https://towardsdatascience.com/forecast-kpi-rmse-mae-mape-bias-cdc5703d242d. [Último acceso: 23 06 2023].

[42] N. Unidas, «La Asamblea General adopta la Agenda 2030 para el Desarrollo Sostenible,» UN, 25 08 2015. [En línea]. Available: https://www.un.org/sustainabledevelopment/es/2015/09/la-asamblea-general-adopta-la-agenda-2030-para-el-desarrollo-sostenible/. [Último acceso: 23 06 2023].

10. Anexos

10.1. Manual de Usuario.

En el Anexo Figura 1 se muestra un ejemplo de la estructura de directorios utilizada en este proyecto:



Anexo Figura 1: Estructura de directorios del proyecto.

Dentro del directorio ‘‘conf’’ debe estar el archivo de configuración ‘‘config_train_val_test.yaml’’.
Dentro del directorio ‘‘checkpoints’’ se debe encontrar el archivo de extensión ‘‘.pth’’ generado con ‘‘train_SSDCNet.py’’
Dentro de los directorios ‘‘imagen_csv/test_data/’’ e ‘‘imagen_csv/train_data/’’ se deben encontrar en ‘‘images’’ las imágenes de entrenamiento y validación respectivamente y dentro de ‘‘ground-truth’’ las etiquetas realizadas en formato ‘‘.csv’’ de entrenamiento y validación respectivamente.

Los siguientes scripts no precisan de argumentos enviados mediante línea de comandos a la hora de la ejecución, a menos que se indique lo contrario:

Obtención de fotogramas de un vídeo:

El script **‘‘obtenedorDeFrames.py’’** toma vídeos, los corta en una sección determinada modificable y guarda capturas del vídeo cortado con el nombre del vídeo y número de fotograma.

- ✓ **frecuencia** = Indica cada cuántos fotogramas se guardará una captura.
- ✓ **cut_start** = Tiempo inicial del corte en segundos.
- ✓ **cut_end** = Tiempo final del corte en segundos.
- ✓ **input_folder** = Ruta del directorio con los archivos ‘‘.mp4’’ que se cortaran, ejemplo: ‘‘./videos/muestra/’’ .
- ✓ **output_folder** = Ruta del directorio de destino con los archivos ‘‘.mp4’’ que han sido cortados, ejemplo: ‘‘./videos/cortados/’’.
- ✓ **path** = Ruta del directorio de destino con los fotogramas ‘‘.jpg’’ que se obtenido. Ejemplo: ‘‘./frames/muestra/’’

Generación del conjunto de datos de entrenamiento y validación:

El script de **“createDataset.py”** renombra los datos que serán utilizados tanto para entrenamiento como para validación y les aplica un factor de reducción de 2.5. Precisa de las definiciones de ruta que se han de indicar dentro del script.

- ✓ **dir_csv_train** = Ruta del directorio con los archivos ‘‘.csv’’ que contienen las etiquetas. Ejemplo: ‘‘./imagen_csv/test_data/ground-truth/’’.
- ✓ **dir_image_train** = Ruta del directorio con los archivos ‘‘.jpg’’ que contienen las imágenes etiquetadas. Ejemplo: ‘‘./imagen_csv/test_data/images/’’.
- ✓ **dir_csv_resize_train** = Ruta de destino del directorio con los archivos ‘‘.csv’’ las etiquetas reducidas. Ejemplo: ‘‘./imagen_csv/resize/test_data/images/’’.
- ✓ **dir_image_resize_train** = Ruta de destino del directorio con los archivos ‘‘.jpg’’ que contienen las imágenes reducidas. Ejemplo: ‘‘./imagen_csv/resize/test_data/images/’’.

Generación de mapas de densidad

El script **“gen_density_maps_SSDCNet.py”** genera dos archivos que contienen los datos de entrenamiento y los datos de validación con los nombres ‘‘density_maps_S9_train.npz’’ y ‘‘density_maps_S9_test.npz’’. El ‘‘S9’’ indica el valor de sigma con el que han sido creados.

Previo a la ejecución del script se ha de definir en el archivo de configuración ‘‘config_train_val_test.yaml’’ los siguientes parámetros:

- ✓ **dataset_rootdir**: Ruta donde se encuentran los archivos ‘‘.jpg’’ y ‘‘.csv’’ que contienen las imágenes y etiquetas reducidas. Ejemplo: ‘‘./imagen_csv/resize/’’. En esta ruta también se depositan los archivos de entrenamiento y validación de extensión ‘‘.npz’’. Luego de su creación, en este proyecto se han movido dentro del directorio ‘‘npz’’.
- ✓ **sigma** = Valor entero de la desviación estándar utilizada para la generación de los mapas de densidad.

- ✓ **sigma_fixed**: Si se define como "True" el valor de sigma será fijo y el definido en el archivo de configuración. En caso contrario, si es "False" utiliza una estrategia de geometría adaptativa.
- ✓ **num_proc**: número de proceso en paralelo para ejecutar el algoritmo de generación.

Los siguientes parámetros se utilizan sólo si se elige el funcionamiento de geometría adaptativa:

- ✓ **knn**: Define el número de vecinos para calcular la distancia media.
- ✓ **max_knn_avg_dist**: Se guardará en esta variable el valor máximo de la distancia media.
- ✓ **sigma_coef**: Valor del coeficiente por el que se multiplica la distancia media, para el cálculo de la desviación estándar (sigma) adaptativo.
- ✓ **sqr_side**: El filtro del kernel gaussiano será del tamaño del cuadrado [-sqr_side/2, +sqr_side/2] en píxeles.

Entrenamiento de la red:

El script para entrenar el modelo es **"train_SSDCNet.py"**. Este script genera el checkpoint o archivo que contiene la información aprendida durante el entrenamiento de la red. Por otro lado, genera diversos archivos visualizables con Tensorboard que describen la evolución del entrenamiento.

Para ejecutar Tensorboard en la terminal o ''cmd'' Desde el directorio raíz del proyecto se ejecuta ''tensorboard -- logdir=. ''.

Este script recibe los siguientes argumentos a través de la línea de comandos:

- ✓ **dataset_rootdir** = Ruta del directorio raíz donde se encuentran las imágenes de entrenamiento reducidas en dimensión. Ejemplo: ''./imagen_csv/resize/''.
- ✓ **densmaps_gt_npz** = Ruta del archivo de extensión ''.npz'' con los mapas de densidad de entrenamiento. Ejemplo: ''./imagen_csv/resize/npz/density_maps_*.npz''.
- ✓ **num_intervals** = El número máximo de peces por sección de 64x64 pixeles, también llamado $C_{máx}$.

Para su ejecución introducir en la línea de comandos los siguiente:

```
Python .\train_SSDCNet.py --dataset_rootdir=./imagen_csv/resize/ --
densmaps_gt_npz=./imagen_csv/resize/npz/density_maps_*.npz --num_intervals 18
```

El archivo de configuración "config_train_val_test.yaml" contiene parámetros asociados a la validación de la red en las secciones ''train'' y ''validation''.

Estimación de objetos en las imágenes:

El script **''inference_with_plot.py''** realiza el conteo de objetos en la imagen partiendo del archivo de extensión ''.npz'' y del checkpoint generado en el script anterior. A su vez, devuelve el nombre de cada imagen y el conteo asociada a ella y los siguientes valores de error asociados a los resultados del conjunto de validación: MSE, RMSE, MAPE. Se han de definir las siguientes rutas para el funcionamiento del script.

- ✓ **directory** = Ruta donde se encuentran las imágenes de validación originales, sin reducción de dimensiones. Ejemplo: ''./imagen_csv/test_data/images''.

- ✓ **dir_conteo_CITSEM** = Ruta del directorio de destino de la estimación de cada imagen de validación. Ejemplo: ‘‘./conteo_CITSEM/’’.
- ✓ **cfg.test.trained.imgs_for_inference** = Ruta donde se encuentra el checkpoint que se utilizará para la inferencia. Ejemplo: ‘‘./checkpoints/epoch_0400.pth’’
- ✓ **real_counts** = Es un arreglo que contiene los conteos etiquetados reales necesarios para el cálculo de errores, en orden alfanuméricamente ascendiente.

En el archivo de configuración “config_train_val_test.yaml” se ha de definir el siguiente parámetro:

- ✓ **num_intervals**= El número máximo de peces por sección de 64x64 pixeles, también llamado $C_{máx}$.

Obtención del valor del parámetro Cmáx:

El script **‘‘obtenerCmax.py’’** analiza todas las imágenes de un directorio cuadro por cuadro, donde cada cuadro tiene un tamaño de 64×64 píxeles. Si encuentra que un cuadro contiene un rodaballo, registra la cantidad de rodaballos en ese cuadro. Una vez que ha analizado todos los cuadros en la imagen, guarda la información en un nuevo archivo .csv. Este archivo contendrá información sobre cuántos peces se encontraron en cada cuadro de la imagen. Finalmente, el código genera histogramas que muestran la distribución del número de turbot en los cuadros de las imágenes. Estos histogramas se guardan como imágenes ‘‘.png’’ y adicionalmente, imprime los percentiles (25, 50, 75, 95) de la distribución, que son medidas estadísticas que proporcionan información sobre la distribución de los datos. Se han de definir las siguientes rutas para el funcionamiento del script:

- ✓ **dir =** Ruta del directorio raíz donde se encuentran las imágenes reducidas en dimensión y su archivo de etiquetas correspondiente de extensión ‘‘.csv’’. Ejemplo: "./imagen_csv/resize/"
- ✓ **dir_csv_cmax =** Ruta del directorio raíz donde se guardaran los archivos de extension ‘‘csv.’’ con los conteos de rodaballos y los archivos ‘‘.png’’ con los histogramas de Cmáx. Ejemplo: "./csv_cmax/"

Obtención del valor del parámetro desviación estándar o sigma:

El script **‘‘obtenerSigma.py’’** genera mapas de densidad a partir de un conjunto de datos de imágenes y archivos ‘‘.csv’’ asociados. Los archivos contienen las coordenadas de los centroides de los rodaballos presentes en cada imagen. Crea los mapas de densidad de cada imagen en el directorio raíz según el valor de sigma, en un rango de 2 al 15. Los mapas de densidad para cada una de la imágenes y un sigma en particular se genera en un directorio de nombre **‘‘visualized_predictions_sigma_n’’** donde n es el valor de sigma utilizado. Se ha de definir la siguiente ruta para el funcionamiento del script:

- ✓ **dir =** Ruta del directorio raíz donde se encuentran las imágenes reducidas en dimensión y su archivo de etiquetas correspondiente de extensión ‘‘.csv’’. Ejemplo: "./imagen_csv/resize/"