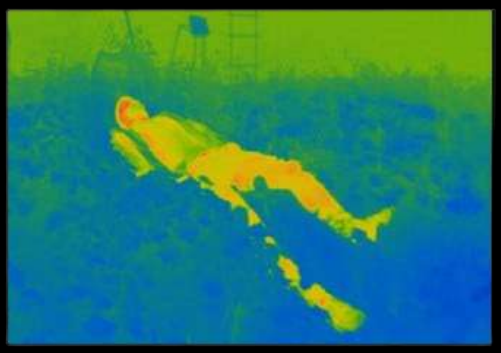
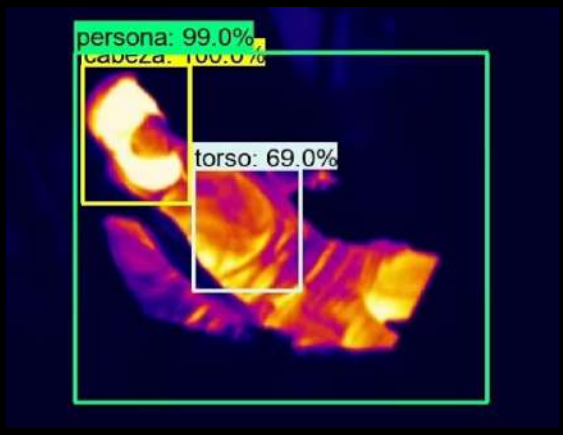
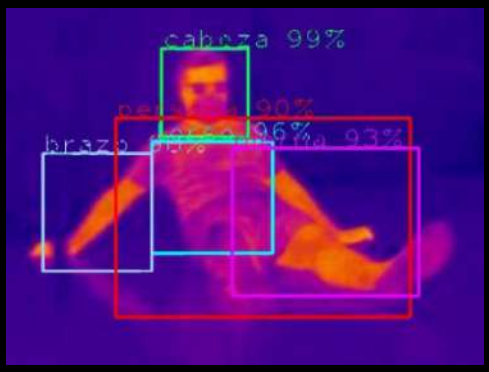




Detección de Víctimas Mediante Redes Neuronales e Imágenes Térmicas



Guillermo Prieto Sánchez

Agradecimientos

Gracias a Christyan Mario Cruz Ulloa, cotutor de este TFG, por guiarme por esta aventura, por su apoyo a lo largo de todo el proyecto y por su implicación, con sus ideas, propuestas y momentos de ayuda cuando más lo he necesitado.

Gracias a Antonio Barrientos Cruz, tutor de este TFG, por inspirarme, por darme la oportunidad de poder realizar este proyecto, por siempre estar disponible para dar consejos y alguna corrección, y por todo el apoyo moral que me ha dado durante todo este tiempo.

Gracias a mi familia, mis padres y mi hermana, por todo el apoyo que me han dado día a día desde siempre. Sin ellos no estaría aquí.

RESUMEN

La aplicación de redes neuronales convolucionales a la visión artificial supuso un avance para los algoritmos de detección. Actualmente se están desarrollando múltiples aplicaciones de dichos algoritmos de detección junto con empleo de cámaras térmicas para tareas de vigilancia o reconocimiento facial.

El presente trabajo tiene como objetivo implementar modelos que utilicen estos algoritmos que emplean redes neuronales para desarrollar la tarea de detección de víctimas en catástrofes o situaciones hostiles mediante el uso de imágenes térmicas.

Para obtener estos modelos, se ha configurado un entorno de programación en Python, se han generado diversos datasets en los que aparecen principalmente personas simulando ser víctimas, se han etiquetado y se han entrenado distintos modelos de redes neuronales convolucionales. Los principales resultados muestran una alta eficacia de los modelos entrenados en la detección de víctimas durante la validación.

En este trabajo se presentan diferentes pruebas, recreando escenarios de desastre para evaluar la eficacia de los modelos obtenidos en distintas condiciones lumínicas y ambientales.

Como principal conclusión extraída a partir de este desarrollo se puede establecer que la detección de víctimas gracias al procesamiento de imágenes térmicas mediante redes neuronales es una alternativa viable y robusta en tareas de búsqueda y rescate.

ABSTRACT

The application of convolutional neural networks to computer vision represented a breakthrough for detection algorithms. Currently, multiple applications of these detection algorithms are being developed together with thermal cameras for surveillance or facial recognition tasks.

The present work aims to implement models using these algorithms employing neural networks to develop the task of victim detection in catastrophes or hostile situations by using thermal images.

To obtain these models, a Python programming environment has been set up, several datasets have been generated in which mainly people simulating to be victims appear, labeled and different convolutional neural network models have been trained. The main results show a high efficiency of the trained models in victim detection during validation.

In this work, different tests are presented, recreating disaster scenarios to evaluate the effectiveness of the models obtained in different lighting and environmental conditions.

The main conclusion drawn from this development is that the detection of victims through the processing of thermal images using neural networks is a viable and robust alternative in search and rescue tasks.

Resumen Ejecutivo

Los grandes avances tecnológicos de los últimos años han supuesto una revolución en campos como la Vision Artificial, gracias a la aplicación y desarrollo de nuevas y mejoradas redes neuronales a los modelos de visión. De la misma manera ocurre con instrumentos de medición como sensores o cámaras térmicas, donde se ha mejorado sustancialmente la calidad de las imágenes térmicas usando sensores más específicos. La figura 1 muestra la cámara infrarroja empleada en este proyecto.

Figura 1: Cámara infrarroja Optris PI 640 [1]

Por otra parte, el auge del Deep Leaning que se ha dado en los últimos años, debido a su accesibilidad por ser software libre, ha creado una gran comunidad desarrolladora que ha logrado dar múltiples pasos adelante en el sector, que han permitido generar algoritmos y aplicaciones robustas para diversas aplicaciones.

El presente proyecto tiene como objetivo principal identificar víctimas mediante redes neuronales e imágenes térmicas en entornos de desastre recreados a partir de diferentes materiales como pueden ser escombros.

Para ello se han entrenado diferentes modelos pre-entrenados de redes neuronales convolucionales como son: Faster R-CNN, SSD y YOLOv3, para adaptarlos a la tarea de detección de víctimas.

Para la realización de este proyecto han sido necesarios dos entornos virtuales de programación en Python en los que se han instalado múltiples librerías de Deep Learning y Visión Artificial, como Tensorflow, Pytorch y OpenCV.

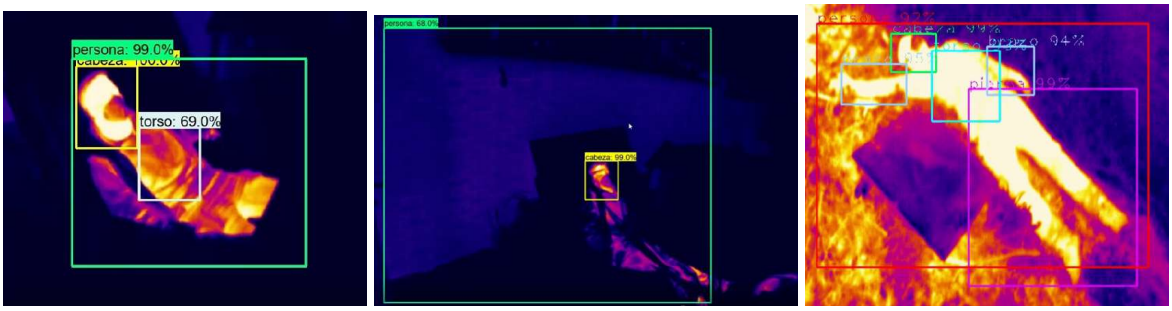
Para entrenar las redes neuronales que sean capaces de detectar víctimas se requiere una gran cantidad de datos de entrenamiento en forma de imágenes. Por ello, se han generado datasets, utilizando la cámara infrarroja, en los que se mostraban personas simulando ser víctimas en distintos escenarios. Estas imágenes se han etiquetado utilizando las etiquetas 'persona', 'cabeza', 'brazo', 'pierna' y 'torso' mediante el software LabelImg. Un ejemplo de etiquetado se representa en la figura 2.

Figura 2: Ejemplo de etiquetado en LabelImg. Fuente:Autor

Posteriormente se han entrenado las redes neuronales, disminuyendo las pérdidas de la red según vaya avanzando el proceso, hasta llegar a un valor crítico de en torno al 5 - 15 % de pérdida. Si se entrena por debajo de este límite sucederá un fenómeno conocido como overfitting, en el cual la red empezará a tener mayor pérdida al evaluar datos con los que no ha sido entrenada la red.

Seguidamente los modelos se evalúan, obteniendo el valor de una serie de métricas que miden la eficacia y precisión de la red.

Se han logrado obtener modelos que emplean redes neuronales para detectar víctimas. Entre ellos el que destaca es la red YOLOv3, por su alta precisión, velocidad y baja pérdida. La figura 3 muestra un ejemplo de detección con cada red.



(a) Faster R-CNN (b) SSD (c) YOLO

Figura 3: Ejemplo de detecciones Faster R-CNN, SSD y YOLOv3. Fuente: Autor

Se han desarrollado una serie de pruebas para evaluar la eficacia de los modelos entrenados.

La primera de ellas fue generar datasets a diferentes horas del día, de manera que haya un dataset con imágenes con mayor contraste térmico, otro con menor contraste, como se ve en la figura 4, y uno que contenga los datos de ambos datasets.

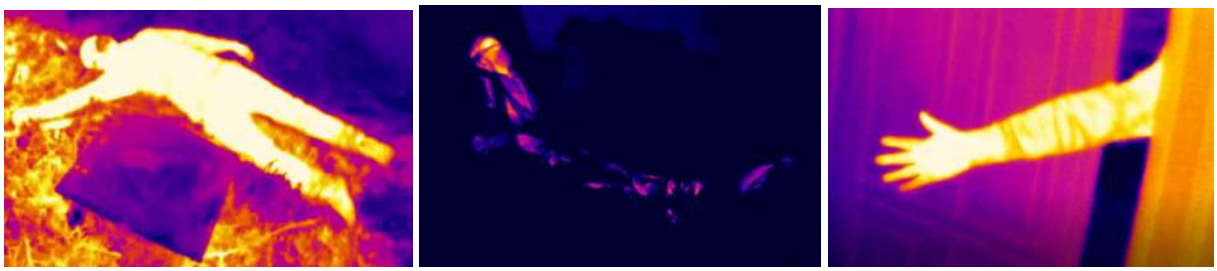


Figura 4: Ejemplos de imágenes variadas recogidas para los datasets. Fuente:Autor

La segunda prueba pretende estudiar la detección de distintas partes del cuerpo. Los resultados en la red Yolov3 indicaron que las clases 'persona','cabeza' y 'pierna' tienen mayor precisión a las de 'brazo' y 'torso', aunque todas están dentro de márgenes aceptables. Los valores de las precisiones de clase aparecen en 5

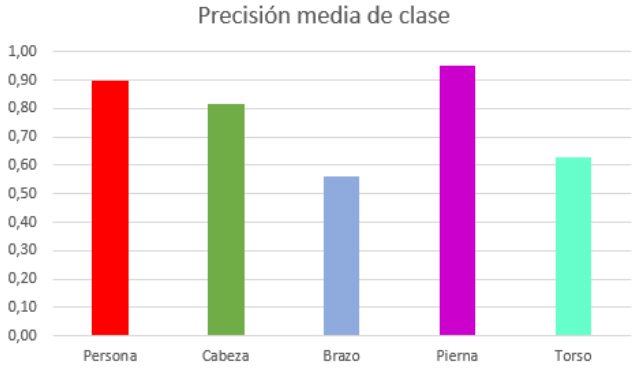


Figura 5: Precisión media de clase para YOLOv3. Fuente: Autor

La tercera prueba pretende diferenciar transeúntes de víctimas y se logró estableciendo una relación entre el largo y ancho del cuadro delimitador, que las distingue para cierto valor, como se ve en la figura 6.

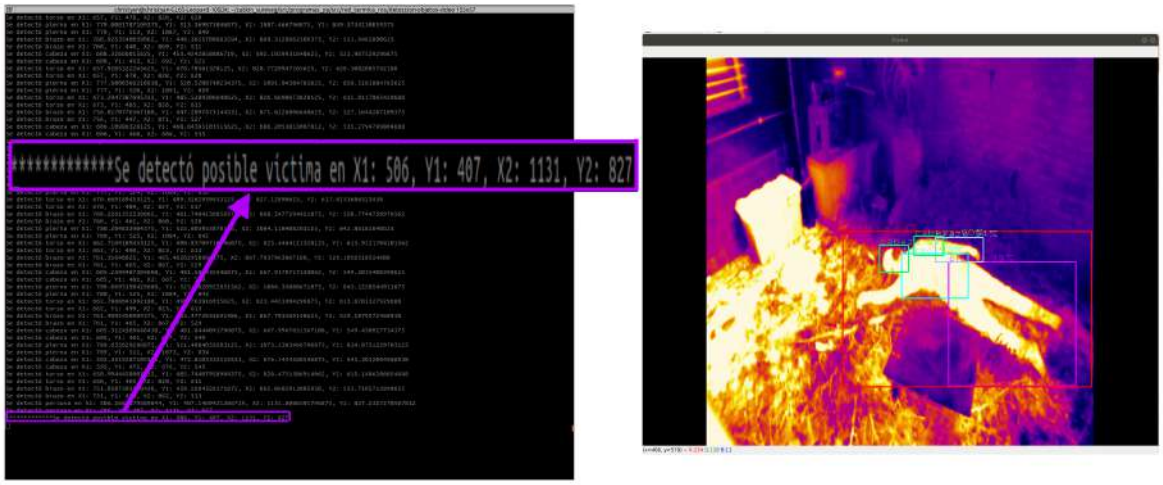
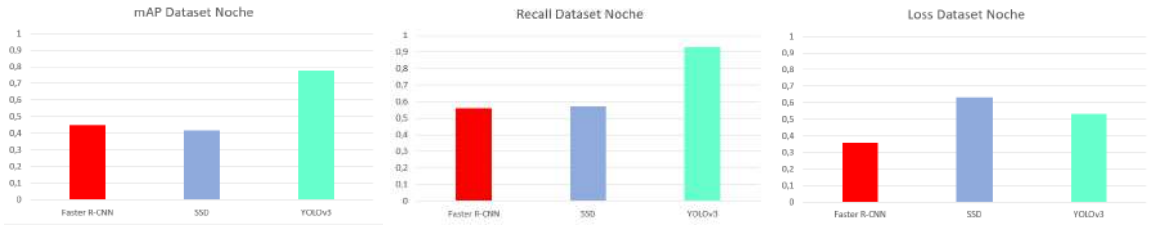


Figura 6: Ejemplo de alerta por pantalla de una posible víctima. Fuente: Autor

La cuarta prueba consiste en probar la red YOLOv3 en vídeos grabados por un robot móvil en espacios de interiores, que se muestan en la figura 7.

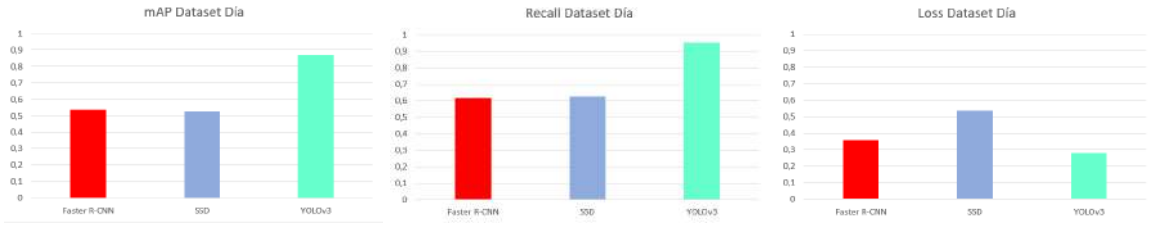
Figura 7: Robo Unitree A1 con la cámara térmica en campo para generación de datasets (izquierda). Ejemplos uno de los entornos de desastre reconstruido para generación de datasets (derecha). Fuente:Autor

Los principales resultados de este TFG muestran una detección con un porcentaje alto de acierto para los tres tipos de redes entrenadas. Esto se puede ver en las figuras 8, 9 y 10.



(a) mAP (b) Recall (c) Loss

Figura 8: mAP, Recall y Loss para las redes con dataset de noche. Fuente: Autor



(a) mAP (b) Recall (c) Loss

Figura 9: mAP, Recall y Loss para las redes con dataset de día. Fuente: Autor

(a) mAP (b) Recall (c) Loss

Figura 10: mAP, Recall y Loss para las redes con dataset combinado. Fuente: Autor

En la actualidad la detección de víctimas utilizando redes neuronales e imágenes térmicas ha sido un campo poco explorado, por lo cual uno de los pilares de relevancia de este Trabajo de Fin de Grado es contribuir al estado del arte mediante la comparativa establecida entre los tres tipos de redes neuronales.

Las aportaciones de este TFG son parte de un proyecto mayor de Grupo de Robótica y Cibernética (ROBCIB) del Centro de Automática y Robótica, denominado TASAR (Team of Advanced Search and Rescue Robots) por sus siglas en inglés. Uno de los objetivos de este proyecto es emplear robots móviles colaborativos para la detección de víctimas en entornos de desastre.

La continuación directa de este trabajo sería la implementación de estos modelos

a sistemas a tiempo real mediante la herramienta ROS, y con ello realizar pruebas y evaluaciones a tiempo real utilizando robots de campo. Como preámbulo se ha evaluado el alcance de este TFG con un video capturado por el robot Unitree A1 utilizado en dicho proyecto, donde se ha demostrado la eficacia de las redes entrenadas en este Trabajo de Fin de Grado.

Palabras clave : Redes Neuronales, Visión Artificial, Búsqueda y Rescate, Imagen Térmica, Termografía y Robot.

Códigos Unesco: 1203 Ciencia de los ordenadores

120304 Inteligencia Artificial

3311 Tecnología de la Instrumentación

331116 Instrumentos de Medida de la Temperatura

3304 Tecnología de los ordenadores

330417 Sistemas en tiempo real

Índice

1. INTRODUCCIÓN

1.1. Marco y Motivación

Según un informe de la UNDRR (Oficina de Naciones Unidas para la Reducción del Riesgo de Desastres), en las últimas dos décadas se han producido 7.348 desastres naturales a nivel mundial que han provocado aproximadamente 1,23 millones de muertes y afectado a más de 4.000 millones de personas. En su mayoría son producidas por inundaciones, tormentas y terremotos, siendo estas el 40 %, 28 % y 8 % respectivamente [2]. En catástrofes como grandes tormentas, las víctimas mortales rondan el 10 % de los afectados, mientras que en los terremotos las víctimas mortales rondan en torno al 49 %. A todo esto hay que añadirle las víctimas por desastres provocados por el ser humano.

En estos escenarios, la robótica ayuda a la rapidez y eficacia de su gestión. Las víctimas pueden estar tanto en la superficie como enterradas bajo escombros. Los robots de búsqueda y recate especializados en reconocimiento y búsqueda de víctimas están diseñados para ayudar a la localización de aquellas víctimas que se encuentran fuera de la visión de los operarios o en lugares poco accesibles. Asimismo, su utilización permite que los operarios no se expongan a peligros, como por ejemplo nuevos derrumbamientos.

Mediante la utilización de una cámara infrarroja en un robot de búsqueda y rescate se pueden llegar a realizar las tareas de reconocimiento y búsqueda en escenarios de accidente con poca luminosidad, como puede ser por la noche o en espacios derrumbados o colapsados, lo que ayudaría a salvar vidas en situaciones de peligro.

1.2. Objetivos

El objetivo principal de este Trabajo de Final de Grado es detectar víctimas en entornos de catástrofe mediante un modelo que emplee redes neuronales e imágenes térmicas.

En la actualidad existen multitud de arquitecturas que implementan redes neuronales para tareas de detección, y es objeto de estudio aplicar algunas de estas arquitecturas para la detección de víctimas. Para lograrlo es necesario entrenar dichas redes, es decir, adaptarlas de manera que sean capaces de desarrollar una tarea en específico, que en este caso sería la detección de víctimas.

Para el entrenamiento y validación de las redes es necesario utilizar conjuntos de datos (datasets) en diferentes condiciones. Es objeto de estudio la creación de dichos datasets, de manera que se entrenen y validen redes utilizando distintos datasets.

De entre las redes que se utilizarán, es objetivo secundario la comparación de la eficacia y viabilidad de estas, a través del análisis de diferentes métricas que se obtendrán tras la validación de las redes en distintas condiciones, de manera que se realice una comparativa donde se muestren sus ventajas e inconvenientes, y así poder hallar la más adecuada para la tarea.

Por último, otro objetivo secundario de este proyecto trata de diferenciar transeúntes y víctimas dentro del entorno de desastre.

A continuación se enumeran de forma resumida los objetivos de este trabajo:

1. Detectar víctimas en entorno de desastre.
2. Generar datasets en distintas condiciones.
3. Entrenar distintas redes con los datasets para la tarea de detección de víctimas.
4. Generar una serie de resultados que permitan establecer comparativas.
5. Evaluar la robusted de los modelos obtenidos a distintas condiciones.
6. Elaborar un sistema que logre distinguir transeúntes de víctimas.

1.3. Principales aportaciones del trabajo

El principal aporte del trabajo es desarrollar un modelo que sea capaz de detectar víctimas bajo diferentes condiciones ambientales.

Ya que los principales desarrollos que combinan los campos de la termografía y el procesamiento digital de imágenes con redes neuronales se enfocan principalmente en tareas de vigilancia y de reconocimiento facial, la aplicación de estos campos para tareas de rescate es un campo con poco desarrollo. Es por ello que cualquier aportación o avance que se pueda dar en este este proyecto puede llegar a ser significativa.

1.4. Estructura del documento

Este documento comienza con un capítulo de introducción en el que se presenta la motivación, contexto, principales objetivos y alcance del proyecto.

Posteriormente se presenta un capítulo en el que se muestra el estado del arte en los ámbitos de búsqueda y rescate de víctimas y aplicación de redes neuronales a imágenes térmicas para detectar víctimas.

Seguidamente se encuentra un capítulo extenso dedicado a explicar el marco teórico, que aborda tres áreas de conocimiento: la primera de termografía, la segunda de Deep Learning y la tercera de Visión Artificial y su aplicación con Deep Learning.

A continuación, se presentan en un capítulo en el que se describen las herramientas de desarrollo utilizadas en este proyecto, como son la cámara térmica empleada y el software requerido para la detección.

En el siguiente capítulo se plantean un capítulo de metodología, en donde se muestra el procedimiento seguido para la configuración de los entornos de trabajo, la generación de datasets que utilizarán las redes, el entrenamiento y validación de las mismas.

Sigue un apartado dedicado a exponer las distintas pruebas que se van a realizar para evaluar la eficacia de los modelos obtenidos, para después exponer los resultados obtenidos, junto con el análisis de dichos resultados y las conclusiones pertinentes.

Finalmente se analiza brevemente la aplicación, beneficios, impactos y futuras líneas de investigación de este proyecto.

Adicionalmente se encuentra al final del documento en forma de anexos información adicional, tales como informes de planificación y coste del proyecto, imágenes y tablas de datos adicionales, y la aplicación del proyecto a tiempo real.

2. ESTADO DEL ARTE

2.1. Tareas de búsqueda y rescate en entornos de desastre

Las labores de búsqueda y rescate (SAR en inglés por ser siglas de Search and Rescue) de víctimas son las actividades realizadas por equipos de emergencia destinadas a localizar a personas en situación de emergencia para proceder a sacarlas del peligro. Además, contempla las labores de tratado y transporte de heridos a centros hospitalarios.

Normalmente los ambientes resultantes de una catástrofe o un grave accidente son muy hostiles, resultando de suma importancia la eficacia y rapidez de las tareas de búsqueda y rescate para que, como resultado, se maximicen las posibilidades de supervivencia de los afectados.

Hay que hacer una distinción entre desastres naturales y artificiales. Los naturales, como terremotos, inundaciones, huracanes o erupciones volcánicas, suelen abarcar áreas extensas por lo que es todo un reto poder detectar todas las posibles víctimas en el menor tiempo posible. En los artificiales, como pueden ser derrumbamientos de edificaciones, incendios provocados o deslizamientos de tierras deforestadas, el reto reside en la accesibilidad de los lugares en donde se producen estos desastres.

Hay múltiples posibles escenarios de catástrofe. Por ello las labores de rescate se suelen clasificar según su ambiente, destacando los de montaña, urbano, en estructuras colapsadas, en combate, marítimo por aire, en aguas rápidas o inundaciones, en zanjas o en cuevas [3].

Durante una emergencia, se sigue un protocolo que consta de múltiples etapas [4]:

- Averiguar el paradero de posibles supervivientes en el área accidentada.
- Analizar el entorno del desastre.
- Realizar un plan de rescate, estimando las necesidades que la situación requiera.
- Evacuar víctimas fácilmente accesibles y retirar escombros en caso de haberlos.
- Atender a las víctimas en situaciones más difíciles y tratarlas in situ si se trata de un caso urgente.
- Transportar las víctimas al centro sanitario más cercano.
- Informar a la base de operaciones, valorar las actuaciones y planificar las nuevas actuaciones.

En la actualidad se están desarrollando robots móviles como apoyo a las tareas de búsqueda y rescate, adaptados a diferentes entornos y para distintas utilidades. Estos robots deben de soportar condiciones extremas, acceder a entornos de difícil acceso y permitir la interacción entre humanos y robots.

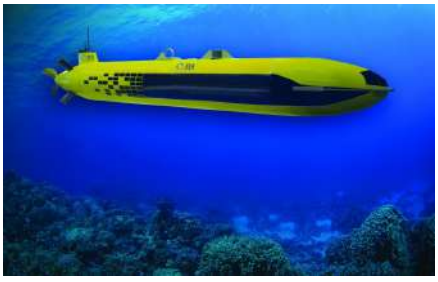
Hay cuatro vertientes en la investigación y desarrollo de robots de rescate no tripulados en función del ambiente en el que van a operar, que se muestran en la figura 11:

- UAVs (Unmanned Air Vehicles): Son robots diseñados para ofrecer una vista aérea del entorno, luego resultan de gran utilidad para abarcar grandes distancias como en el caso de desastres naturales.
- UGVs (Unmanned Ground Vehicles): son robots de tierra, empleados generalmente como apoyo a operarios para localizar y rescatar víctimas en espacios pequeños o peligrosos.
- USVs (Unmanned Surface Vehicles): son embarcaciones que operan en la superficie del agua sin tripulación a bordo, cuya tarea principal es inspeccionar la costa tras un accidente.
- UUVs (Unmanned Underwater vehicles): son robots diseñados para el reconocimiento y búsqueda a moderada profundidad subacuática.

(a) UAV [5]

(b) UGV [6]

(c) USV [7]



(d) UUV [8]

Figura 11: Tipos de robots de rescate en función del medio en el que operan.

Las labores que pueden desempeñar los robots de rescate son las siguientes [4]:

- Reconocimiento: los robots de reconocimiento proveen información de las condiciones del entorno, evaluando la peligrosidad de este.
- Búsqueda: los robots diseñados específicamente para llegar a lugares difícilmente accesibles para localizar y atender a víctimas.
- Retirar escombros: los robot que se dedican a retirar escombros sirven de ayuda complementaria a operarios y excavadoras, aumentando la velocidad con la que se desarrolla esta tarea.
- Asistencia general: comprende labores auxiliares como son la iluminación y transporte de equipo médico de primeros auxilios.

Actualmente se están desarrollando robots de rescate con configuraciones no convencionales, con el fin de que se adapten mejor a ambientes específicos. Es el caso de los robots con orugas, patas, forma de serpiente, esféricos o los polimórficos.

2.2. Aplicación de redes neuronales a procesamiento de imágenes térmicas para detección de personas

En los últimos años, con el apogeo de la Visión Artificial debido a las nuevas técnicas que emplean redes neuronales para tareas como la clasificación y detección de objetos en imágenes y videos, se han explorado e investigado su aplicación en múltiples áreas. Una de estas áreas es la de termografía, en la que se busca la aplicación de los nuevos modelos de visión a la detección de objetos o personas, haciendo especial énfasis en el estudio de estas últimas.

Las principales áreas de aplicación de la visión artificial junto a termografía actualmente son vigilancia y reconocimiento facial.

En cuanto a la vigilancia destaca el trabajo de Park et al [9], llamado *CNN-Based Person Detection Using Infrarred Images for Night-Time Intrusion Warning Systems* cuyo propósito es detectar automáticamente intrusos en ambientes peligrosos durante la noche mediante redes neuronales e imágenes térmicas, para dar la alarma. Los resultados fueron buenos, teniendo buenos parámetros de precisión y una velocidad en inferencia lo suficientemente alta como para aplicar el modelo a tiempo real.

Otro trabajo relacionado con la vigilancia es el de Portmann et al [10], llamado *People Detection and Tracking from Aerial Thermal Views*, que consiste en la detección y seguimiento de personas con una cámara térmica implantada en un robot UAV. La figura 12 muestra algunos de los resultados.

Figura 12: Resultados de People Detection and Tracking from Aerial Thermal Views [10]

Un trabajo relacionado con el reconocimiento facial es el de Ilikei et al [11], llamado *Heat-Map Based Emotion and Face Recognition from Thermal Images*, en el que se consigue distinguir emociones en las caras de las personas, como pueden ser enfado, miedo, felicidad, tristeza o sorpresa, mediante una red neuronal YOLO. Se pueden observar alguno de los resultados en la figura 13.

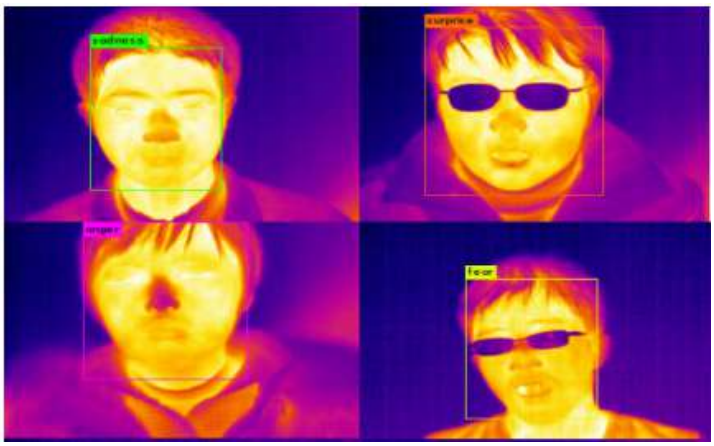


Figura 13: Resultados de Heat-Map Based Emotion and Face Recognition from Thermal Images [11]

Existen estudios que investigan la eficacia de utilizar imágenes térmicas con redes neuronales para distintas condiciones.

Hay un estudio llamado *Evaluation of Thermal Imaging for People Detection in Outdoor Scenarios* [12] que compara la detección de personas en interiores y exteriores. En este se llega a la conclusión de que los detectores de personas funcionan mejor en escenarios de interior que los de exterior. Otra conclusión que se extrae, tanto para interiores como para exteriores, es que, para escenarios de desastre como pueden ser incendios, las redes neuronales entrenadas para ese motivo tienen un buen rendimiento.

En el estudio llamado *Thermal Object Detection in difficult Weather Conditions Using YOLO* [13] se saca la conclusión de que el tiempo atmosférico tiene un papel menor en la detección de personas, pudiendo llegar a obtener buenos resultados en condiciones como pueden ser de lluvia entrenando la red utilizando únicamente imágenes en tiempo despejado.Un ejemplo de los resultados del proyecto se puede ver en la figura 14.

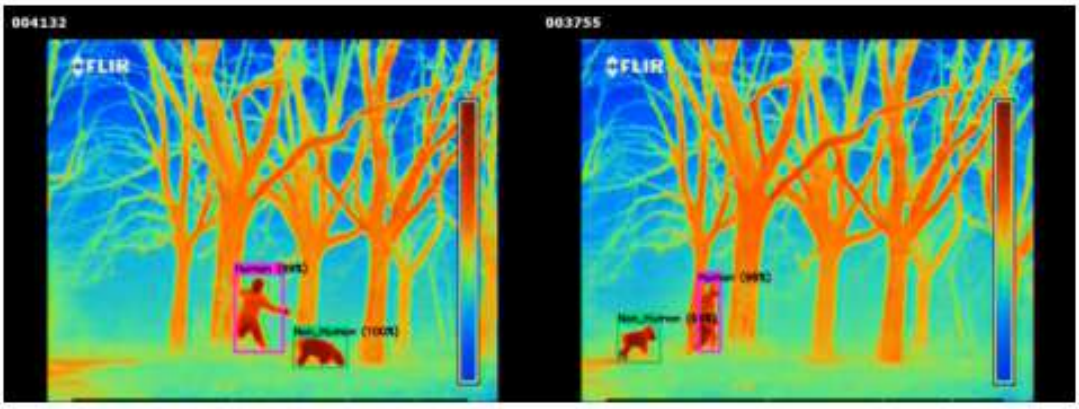


Figura 14: Resultado de Thermal Object Detection in difficult Weather Conditions Using YOLO [13]

Existe una rama de investigación de la detección de personas con cámaras térmicas de muy baja resolución o incluso a partir de sensores infrarrojos. Hay múltiples trabajos relacionados, como los de [14], [15], [16] y [17], que abordan la viabilidad y eficacia de emplear cámaras de muy poca resolución a redes neuronales para sistemas con pocos recursos. Las conclusiones obtenidas de estos trabajos son similares, que destacan la buena precisión que presentan las redes, la rápida velocidad de inferencia y el poco consumo energético que estos sistemas tienen. En la figura 15se muestra la arquitectura empleada en el trabajo [15].

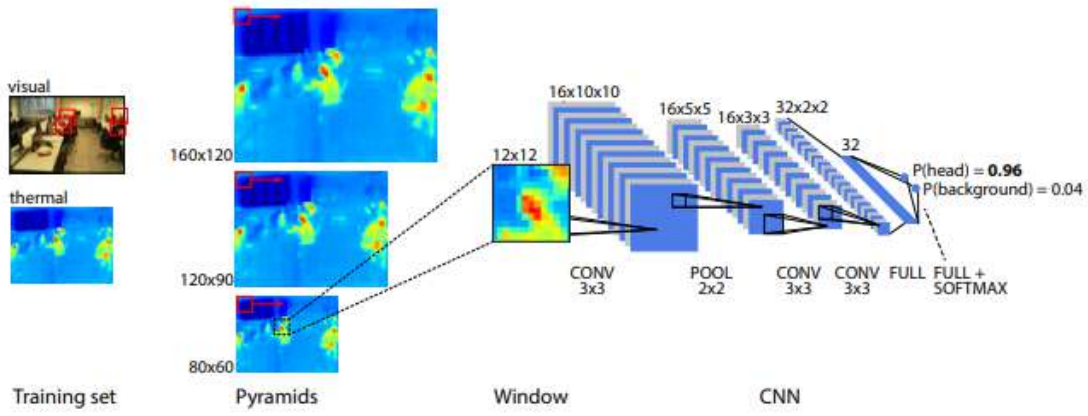


Figura 15: Arquitectura empleada en Thermal Image-Based CNN's for Ultra-Low Power People Recognition[15]

Destacar que están surgiendo actualmente algunos trabajos como el de Perdana et al [18], de título *Automatic Aerial Victim Detection on Low-Cost Thermal Camera Using Convolutional Neural Network*, en los que se están haciendo los primeros pasos en la detección de víctimas mediante redes neuronales e imágenes térmicas.

3. CONCEPTOS TEÓRICOS

3.1. Termografía

3.1.1. Introducción a la termografía

La termografía es una técnica que permite determinar temperaturas de seres vivos u objetos a distancia, sin establecer contacto físico, mediante la captación de la radiación infrarroja del espectro electromagnético que emite todo cuerpo mediante el uso de cámaras térmicas, también conocidas como cámaras termográficas o cámaras infrarrojas [19].

De acuerdo con la ley de Planck, que describe la radiación electromagnética emitida por un cuerpo negro en equilibrio térmico a una temperatura dada, todo cuerpo por encima del cero absoluto emite radiación infrarroja [20]. Este principio es la base que se utiliza para estimar las temperaturas de sujetos, representadas luego en imágenes térmicas.

Las imágenes térmicas o termogramas generadas con estas cámaras son, en realidad, representaciones de la cantidad de energía emitida, transmitida y reflejada por un objeto o ser. La energía emitida, como su nombre indica, es la que emite el sujeto que queremos medir, la energía transmitida es la energía que pasa a través del sujeto en cuestión proveniente de otra fuente térmica remota, y la energía reflejada es la energía proveniente de una fuente remota que es reflejada por la superficie del sujeto [21].

La suma de estas tres energías, que es lo que capta la cámara térmica, se le conoce como energía incidente. Esta relación también se describe en torno a las potencias radiantes de la misma manera, ya que potencia radiante * tiempo = energía radiante:

Potencia radiante incidente = Potencia radiante emitida + Potencia radiante transmitida + Potencia radiante reflejada

Realmente lo que interesa medir para obtener la temperatura real del objeto o ser en cuestión es la energía emitida. Es por ello que, debido a la interacción de las otras fuentes de energía, es difícil medir con precisión la temperatura con este método.

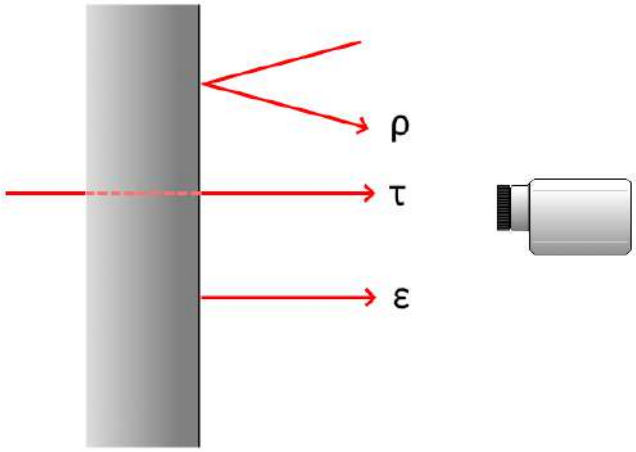


Figura 16: Energía emitida, transmitida y reflejada por un objeto. Fuente: Autor

En la figura 16 ϵ representa la emisividad, τ la transmitancia y ρ la reflectividad.

La emisividad (ϵ) es la capacidad que tiene la superficie de un material de emitir radiación infrarroja. Es una propiedad óptica de la materia que depende tanto de la temperatura como la longitud de onda del infrarrojo. Se define un cuerpo gris como aquel cuerpo ideal que posee una emisividad constante para toda longitud de onda. Esta definición es útil en ingeniería para tratar la emisividad como constante en la banda infrarroja, llamando a esta regla la "suposición del cuerpo gris".

Cuantitativamente, la emisividad es la relación entre la radiación emitida por una superficie de un objeto respecto a la emitida por un cuerpo negro a la misma temperatura, tal y como expresa la ley de Stefan–Boltzmann. El cuerpo negro, que como se explicará más adelante es el emisor ideal , tiene una emisividad de 1, por lo que cualquier objeto real tiene una emisividad menor, luego esta relación está comprendida entre 0 y 1.

La emisión o poder emisor es la energía total emitida por unidad de área y tiempo para toda longitud de onda en el infrarrojo. Depende tanto de la emisividad del propio material como de la superficie de este.

La absorbancia (α) indica el grado en el que la energía infrarroja es absorbida por el material. Un cuerpo negro absorbe toda la radiación incidente, por lo que se le trata como el absorbedor perfecto, $\alpha = 1$.

La transmitancia (τ) es la capacidad que tiene una superficie de transmitir energía térmica. Indica la permeabilidad del material a la radiación infrarroja y, al igual que la emisividad, es función de la longitud de onda. La mayoría de materiales no transmiten la radiación infrarroja, por lo que se suele despreciar la transmitancia.

La reflectividad (ρ) es la capacidad de una superficie de reflejar radiación infrarroja. Es función de la longitud de onda y se trata de una propiedad direccional, por lo que el comportamiento varía según el tipo de superficie. En superficies especulares, como metal pulido o vidrio, la reflectividad es mayor en la dirección del reflejo del ángulo de incidencia mientras que tiende a 0 en los ángulos distintos. En superficies dispersas o rugosas la reflectividad es igual para cualquier dirección. Estas superficies con difusión óptima también se las conoce como superficies de Lambert [22].

Según la ley de radiación de Kirchhoff, si un cuerpo está en equilibrio termodinámico con su entorno, la radiación absorbida y la emitida son iguales. De esta manera se deduce que la emisividad es igual a la absorbancia:

$$\epsilon_\lambda = \alpha_\lambda \tag{1}$$

Por esto último, un cuerpo negro ideal que absorbe toda radiación, también la repele. Es por ello que, como se mencionaba anteriormente, el cuerpo negro también es el emisor ideal:

$$\epsilon = \alpha = 1 \tag{2}$$

El intercambio de energía por radiación térmica se caracteriza por la siguiente ecuación:

$$\alpha + \tau + \rho = 1 \tag{3}$$

Como la emisividad es igual a la absorbancia como se vio en la ecuación (1), la expresión anterior se puede reescribir como:

$$\alpha + \rho = 1 \tag{4}$$

Esta última expresión permite distinguir dos tipos de objetos:

- Objetos de emisividad alta ($1 < \epsilon < 0,8$) y emisividad media ($0,8 < \epsilon < 0,6$): la temperatura es fácilmente medible con precisión.
- Objetos de emisividad baja ($0,6 < \epsilon < 0$): la temperatura es difícilmente medible con precisión.

Además, ciertos materiales presentan diferentes emisividades dependiendo de la longitud de onda y con ello la temperatura. A estos materiales se les conoce como cuerpos coloreados o cuerpos de color. Ejemplos de este tipo de cuerpos son la mayoría de materiales metálicos, como es el aluminio, que posee una ϵ = 0,02 a 25 °C y una ϵ = 0,03 a 100 °C [23].

3.1.2. Propiedades térmicas de los materiales

Como ya se ha mencionado anteriormente, la fracción de la energía incidente que hay que captar para medir la temperatura de un objeto es la energía emitida. La emisividad juega un papel fundamental para la estimación de esta energía en base a la energía recibida. Diferentes materiales tienen diferentes valores de emisividad, lo que lleva a mediciones de temperatura más o menos precisas. Por ello, es necesario calibrar los instrumentos de medida adecuándolos a la emisión del objeto de estudio y la temperatura ambiente. A mayor diferencia de temperaturas entre el objeto y la temperatura ambiente, y a menor emisividad de dicho objeto, los errores en medición serán mayores.

La obtención de la emisividad de un material a una temperatura ambiente y humedad dada se puede realizar de tres maneras diferentes:

- Mediante tablas de emisividad, donde se muestra la emisividad de materiales comunes en diferentes ámbitos.
- Mediante comparación y ajuste de medidas con un termómetro de contacto.
- Mediante la comparativa con un objeto con emisividad conocida.

Tabla 1: Tabla emisividad [24]

Material	Temperatura (°C)	ϵ
Aluminio, laminado brillante	170	0,04
Asfalto	20	0,93
Hormigón	25	0,93
Plomo, oxidado	20	0,28
Hielo	0	0,97
Hierro, esmerilado	20	0,24
Hierro, brillante	150	0,13
Hierro, oxidado	20	0,85
Tierra	20	0,66
Vidrio	90	0,94
Plata	20	0,02
Madera	70	0,94
Plásticos (PE,PP,PVC)	20	0,94

Los materiales orgánicos tienen por lo general una alta emisividad, por lo que su medición suele ser sencilla. Materiales como papel, cerámica, madera, tierra, plantas, arena, goma, piedra, pinturas, recubrimientos oscuros o mates, etc, tienen una emisividad de aproximadamente 0,95 en el rango espectral de 8 a 14 μm. La emisividad de las cámaras térmicas suele estar preajustada a este valor de emisividad, por la gran cantidad de materiales que tienen una emisividad cercana a este valor, para así evitar errores de medición por ajustar la emisividad incorrectamente. Además, el agua, hielo y nieve también presentan una emisividad muy elevada, de 0,92 – 0,96, 0,96 – 0,98 y 0,83 respectivamente.Por ello, los ambientes naturales de exterior tienen una emisividad media entre 0,8 y 0,95.

Para entornos urbanos, de construcción e industriales, hay diferencias de emisividad entre los diferentes materiales comunes.

Para materiales de construcción como hormigón (0,93), ladrillo normal (0,92 – 0,94), ladrillo esmaltado (0,94 – 0,89), vidrio (0,95 – 0,98) y asfalto (0,98), la emisividad es muy elevada luego la temperatura es fácilmente medible.

Los materiales metálicos tienen un rango muy amplio de emisividades, dependiendo del tipo de metal a medir, estado superficial, recubrimientos, etc.

Los metales brillantes tienen muy baja emisividad en el rango de 8 a 14 μm normalmente menor a 0,10, por lo que son difíciles de medir.

Si se observa una superficie de un metal con poca emisividad pulido, la superficie de este metal actuará como un espejo, por lo que en vez de medir la temperatura del propio objeto la cámara mostrará a temperatura reflejada, también llamada temperatura de fondo. Es una radiación térmica procedente de otros objetos que reflejan en el que se está midiendo.

Para casos extremos con emisividad prácticamente nula, como por ejemplo la plata pulida que tiene un coeficiente de emisividad de tan solo 0,02 [21], la radiación que realmente se mide procede del entorno alrededor del objeto a medir, no la radiación del propio objeto. Esto puede llegar a dar mediciones totalmente erróneas, como es el caso de objetos fríos que aparentan estar calientes en la imagen térmica por estar reflejando en su superficie radiación originada en un foco externo, o el caso contrario donde objetos se muestran mucho más fríos de lo que realmente se encuentran.

Todo esto último tiene una influencia mucho menor en los metales con emisividad más alta, puesto que con ello la reflectancia es menor. Este fenómeno no ocurre en metales con recubrimientos como son con pintura, aceite o cinta adhesiva para superficies

reflectantes, que aumentan considerablemente la emisividad.

Los óxidos metálicos varían su emisividad desde 0,3 a 0,9 y por lo general dependen mucho de la longitud de onda. Este comportamiento tan poco uniforme dentro de este grupo hace que existan materiales que no tengan problema alguno para la medición y otros cuya medición con precisión sea prácticamente imposible.

Para los plásticos un factor muy importante es el grosor del objeto. Plásticos gruesos suelen presentar una emisividad alta que ronda 0,86 – 0,95. El problema surge con las películas plásticas finas. Estas cuentan con una transmisividad muy elevada, por lo que al medir la temperatura con una cámara la temperatura, lo que se muestra no es la temperatura de la propia película, sino que la de los objetos colocados detrás de esta, como se ve en la figura 17.

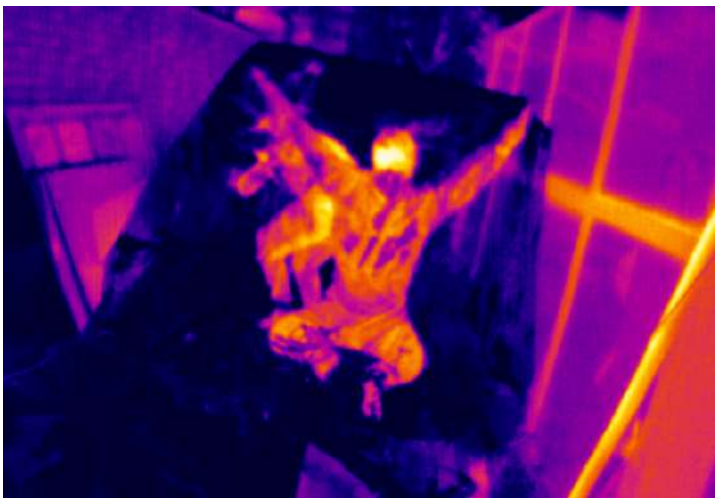


Figura 17: Ejemplo de transmisividad de películas finas. Fuente: Autor

Para el objetivo principal de este estudio, que es la detección de personas, cabe destacar dos valores de emisividad relevantes. El primero es la emisividad media de la piel humana, la cual es sorprendentemente alta, con un valor de 0,98. El segundo valor de emisividad media es el de la ropa, la cual se sitúa en torno a 0,95. Ambos son muy próximos entre ellos y son coincidentes a la calibración predeterminada de las cámaras.

3.1.3. Condiciones ambientales

Además de ser necesario conocer el comportamiento de los materiales a estudiar, es de suma importancia tener en cuenta los factores ambientales que puedan influir en la medición.

- Temperatura ambiente

Esta tiene una gran influencia en materiales con una emisividad baja, en los que gran parte de la radiación que emiten es la reflejada. En ausencia de fuentes de radiación como pueden ser calefactores o la radiación celestial, la temperatura reflejada es la del ambiente, por lo que la temperatura aparente medida difiere en unos pocos grados de la temperatura ambiente.

La temperatura ambiente además juega un rol muy importante en la detección de objetos o seres en imágenes térmicas. Si la temperatura ambiente es similar a la del objeto a detectar, el contraste será pequeño luego la calidad de los resultados obtenidos puede disminuir. A temperatura ambiente de alrededor de 20-25°C [16] la detección de personas se dificulta considerablemente, ya que esta suele ser la temperatura exterior del cuerpo humano en ambiente externo.

- Radiación celestial fría difusa y radiación solar

La radiación celestial fría es la radiación infrarroja procedente del cielo. En días despejados es la radiación reflejada por los objetos, junto con la radiación solar. Para evitar la inferencia de estas radiaciones es preferible trabajar en exteriores con clima nublado.

- Humedad

Debido a que el agua, hielo y nieve cuentan con una emisividad muy alta, lo que lleva a una transmitancia baja que no deja pasar la radiación infrarroja. Además, en el caso de la humedad, las gotas en suspensión a su vez absorben la radiación y la emiten, dispersándola. Por tanto, la humedad relativa debe de ser la menor posible, y así además se evita la condensación en el objeto que se quiere medir o en la lente de la cámara. De lo contrario la radiación llegará parcialmente a la lente de la cámara dando lecturas incorrectas.

- Corrientes de aire

Por el intercambio de calor por convección, el aire cercano a la superficie del objeto que se quiere medir tiene la temperatura prácticamente idéntica a la de dicho objeto. Al haber corrientes de aire, esta capa se ve desplazada y la sustituye una capa que todavía no está en equilibrio térmico con el objeto. Para lograrlo el objeto cede o absorbe calor hasta que la temperatura del objeto y del aire se acaben igualando. El intercambio de calor es mayor a mayor diferencia de temperaturas entre la superficie del objeto y el ambiente.

- Polución

Algunas partículas en suspensión como son el humo negro, hollín o el polvo presentan una emisividad elevada, que igual que en el caso de la humedad con el agua, no deja transmitir la radiación infrarroja, sino que la absorbe y emite, dispersándola. La radiación del objeto llegará parcialmente a la cámara dando lecturas incorrectas. Es por este motivo por el que se deben de evitar ambientes con mucha contaminación atmosférica.

- Luz

A pesar de no afectar la luz visible a las mediciones, sí puede llegar a afectar la radiación infrarroja que pueda generar la fuente de esta luz. Ejemplos de este tipo de fuentes son las bombillas incandescentes o la luz solar. Las luces frías como LEDs o neones son más eficientes energéticamente, por lo que la mayor parte de la energía recibida para su funcionamiento se convierte en luz visible y no en calor, emitidiendo mucha menos radiación infrarroja, lo que afecta mucho menos las mediciones

3.1.4. Tipos de cámaras térmicas y representación de la imagen

La atmósfera está compuesta de una mezcla de gases que absorbe y dispersa una gran proporción de la radiación del total del espectro electromagnético [25]. Esta es permeable a unas determinadas longitudes de onda llamadas ventanas atmosféricas, como se ve en la figura 18. Por ejemplo, existe una ventana atmosférica para la luz visible que permite que penetre un 50 % de esta, lo que permite la vida tal y como la conocemos.

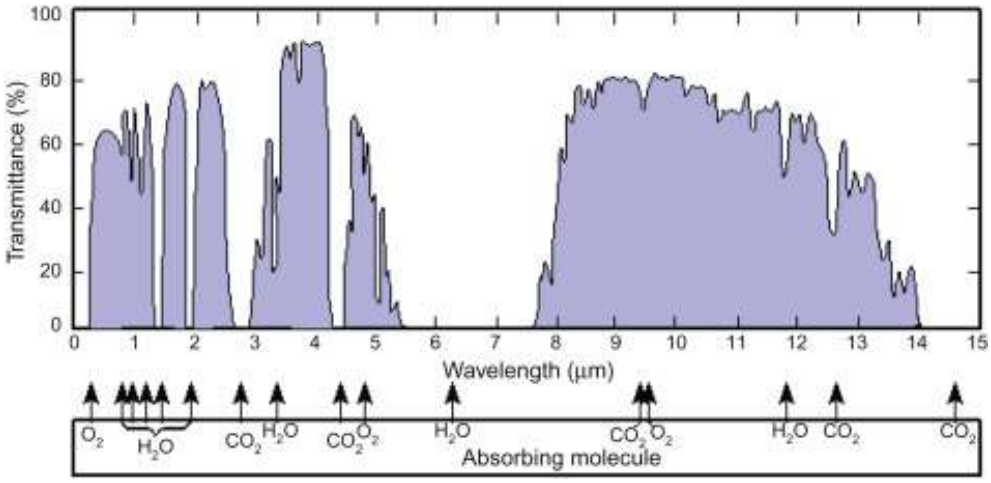


Figura 18: Transmitancia de la atmósfera terrestre en función de la longitud de onda, y las moléculas que contribuyen a la absorción de la radiación [26].

En estas ventanas no hay o hay muy poca absorción o emisión de radiación debido al aire entre el objeto que se desea estudiar y el instrumento de medición, en nuestro caso la cámara térmica. La radiación infrarroja es un tipo de radiación electromagnética que abarca desde los 700 nm hasta 1 mm, o lo que equivale en frecuencia 430 THz a 300 GHz. Al abarcar un rango tan amplio, el infrarrojo se ha clasificado en bandas de diferentes maneras como división CIE, la ISO 20473, división de astronomía, bandas de telecomunicación, etc. En la figura 19 y tabla 2 se mostrará la subdivisión de uso común.

Figura 19: Espectro electromagnético del infrarrojo. Fuente: Autor

Tabla 2: Clasificación infrarrojos

	Abreviatura	Longitud de onda (μm)	Frecuencia (THz)	Energía Fotónica (MeV)	Temperatura (K)
Infrarrojo cercano	NIR	0.75 – 1.4	214 – 40	886 – 1,653	3,864 – 2,070
Infrarrojos de onda corta	SWIR	1.4 – 3	100 – 214	413 – 886	2.070 – 966
Infrarrojos de longitud de onda media	MWIR	3 – 8	37 – 100	155 – 413	966–362
Infrarrojos de longitud de onda larga	LWIR	8 – 15	20 – 37	83 – 155	362-193
Infrarrojo lejano	FIR	15 – 1,000	0.3 – 20	1.2 – 83	193–3

Al infrarrojo cercano y de onda corta se les conoce también como infrarrojo reflejado y al infrarrojo de longitud de onda media y larga se les conoce como infrarrojo térmico. Otras abreviaturas utilizadas son IIR para el MWIR y VLWIR para el FIR.

En base a esto se puede deducir lo siguiente:

- Para la medición de temperaturas superiores a los 1000 °C, el rango espectral adecuado es el visible, el infrarrojo cercano e infrarrojo de onda corta.
- Para la medición de temperaturas medias, el rango espectral adecuado es el de las ventanas ambientales de 2 a 2,5 μm y de 3,5 a 4,2 μm, que corresponde con las franjas SWIR y MWIR.
- Para la medición de temperaturas bajas, el rango espectral adecuado es el de la ventana ambiental de 8 a 14 μm, siendo una banda de energía ancha, que corresponde con la franja LWIR.

La gran mayoría de las cámaras térmicas comerciales trabajan en el rango LWIR y MWIR.

El rango LWIR es ideal para mediciones técnicas, puesto que en este rango hay una transmisividad atmosférica muy elevada y constante debida a la gran banda de la ventana atmosférica. De esta manera las mediciones se pueden realizar a distancias largas, mucho mayores a las de cámaras que trabajan en otras bandas, por lo que este rango es el más común.

Comparado con LWIR, el rango MWIR tiene la desventaja de la debilitación de la señal a largas distancias por la menor transmisividad.

La banda LWIR funciona mejor en niebla y en ambientes fríos, y la MWIR tiene menor afectación por la humedad y funciona mejor en ambientes calientes. Existen sistemas que trabajan con ambas bandas [27].

El rango SWIR es poco utilizada, y se utiliza principalmente para capturar detalles nítidos a través de smog, nubes y niebla, y en aplicaciones especiales como la astronomía.

En cuanto a la representación, la escala de temperaturas se configura para que dependa de la temperatura ambiente. En cuanto a los colores, hay diferentes paletas de colores utilizadas para distintas aplicaciones, que se muestran en la figura 20.

La paleta gris o la paleta ámbar son paletas monocromáticas que son empleadas en aplicaciones donde se requiere un análisis eficaz y comúnmente en vigilancia. Es la paleta que mejor visualiza en pantalla, con más claridad.

Para aplicaciones con diferencias pequeñas de temperatura, donde se requiere un mayor contrate, se utilizan paletas ricas en colores como la paleta de hierro o la paleta arcoíris. En estas paletas se representa la temperatura más fría con color azul y la más caliente con el blanco [28].

(a) Paleta de hierro (b) Paleta gris (c) Paleta arcoíris

Figura 20: Paletas de colores empleadas en termografía. Fuente: Autor

3.1.5. Comparativa entre termografía y fotografía

Una vez vistas las bases teóricas de la termografía, queda responder la pregunta más importante: ¿Qué ventajas e inconvenientes presenta el empleo de cámaras térmicas frente a las fotográficas convencionales?

Para poder comparar ambos tipos de cámaras primero hay que describir brevemente las cámaras fotográficas. Las cámaras fotográficas o cámaras RGB son cámaras que generan imágenes a partir de captación de la luz visible, en el rango del espectro electromagnético de 400 – 700 nm. Su funcionamiento se basa en el principio de la cámara oscura, en el que a través de un agujero, o en este caso una o varias lentes llamadas objetivo, se capta una imagen y se proyecta sobre una superficie. En la actualidad, con las cámaras digitales, esa superficie consiste en un sensor de imagen conformado por una serie de fototransistores [29].

Por lo tanto, el factor principal que permite la fotografía es la cantidad de luz que pueda captar la cámara para formar la imagen. Es por este motivo que este tipo de cámaras tengan una gran dependencia de la iluminación, lo que lleva a que no funcionen correctamente en ambientes oscuros.

Esto último puede representar un problema para actividades como puede ser la vigilancia, ya que este tipo de cámaras no pueden operar de noche o con condiciones climáticas adversas.

Una solución posible para esto puede ser el uso de cámaras con visión nocturna, las cuales simplemente amplifican la señal luminosa recibida. Estas, aparte de presentar otros problemas adversos como son el contrate sujeto-ambiente y la distancia efectiva de funcionamiento, no solucionan los casos de escasez de luz severa.

En campos de investigación como el reconocimiento de gestos faciales o corporales, tanto la iluminación como la posición del foco de iluminación influyen en las detecciones y reconocimientos, lo cual disminuye la precisión de los resultados esperados.

La alternativa más popular para solucionar este tipo de problemas es el uso de la termografía. En el ámbito de reconocimiento de personas, tema a focalizar en este trabajo, las imágenes térmicas o infrarrojas, al resaltar objetos y seres en base a su temperatura, destacan sobre las imágenes en el espectro visible en su mayor inmunidad a condiciones temporales, iluminación o posturas corporales. [30]

Para aplicaciones como por ejemplo la detección de personas en lugares concretos, tales como pueden ser espacios públicos tales como calles, plazas estaciones, o incluso en

espacios más reducidos como pueden ser aulas, las imágenes en el espectro visible tienen, a parte de las desventajas ya mencionadas, problemas relacionados con la privacidad y con la resolución de estas cámaras requerida para dicha tarea. Para una misma precisión y rendimiento, las cámaras convencionales requieren una resolución de imagen mucho mayor al de las cámaras térmicas.

Al utilizar imágenes térmicas con una resolución mucho menor a la de cámaras ordinarias, su procesamiento y almacenamiento en memoria requieren en consecuencia mucho menor capacidad computacional [30]. En algunas situaciones, para cámaras con muy baja resolución, el coste computacional es tan bajo que puede llevarse a cabo en sistemas embebidos con recursos muy limitados, como se puede ver en trabajos como el de Cerutti et al.,2018[16]. En estos casos el consumo energético también es menor.

Una vez vistas las principales ventajas de las cámaras térmicas, ahora hay que tratar sus desventajas.

Una de las grandes desventajas de esta tecnología es el precio. A pesar de su expansión en los mercados en los últimos años, las cámaras térmicas tienen un coste por píxel mucho mayor al de las cámaras convencionales,[30] por lo que en general las cámaras térmicas suelen tener peor resolución y son mucho más caras. Es por ello que, en multitud de ocasiones, se utilizan alternativas como sensores de radiación infrarroja, como son los Passive infraRed sensors (PIR), que se muestra en la figura 21. Estos dispositivos se basan en sensores piroeléctricos que no cuentan con una gran resolución y tienen un coste reducido, y son utilizados principalmente como sensores de movimiento en seguridad.[31]

Figura 21: Sensor PIR [32]

Las cámaras térmicas además tienen la desventaja de tener que ser calibradas periódicamente y según la aplicación, ajustando parámetros como son la emisividad del objeto de estudio o la temperatura ambiente y la humedad del entorno en el que se trabaja.

La frecuencia de refresco es la frecuencia con la que se actualizan las imágenes por segundo en un video. Para grabación en cámaras térmicas, estas frecuencias son generalmente muy inferiores a las de las cámaras convencionales, siendo las de las primeras de en torno a los 5 – 15 Hz (con excepciones de equipos más caros que llegan incluso a los 180 Hz), y las de las última de 60-120 Hz (las más modernas llegan a operar a 240 Hz).

Otro aspecto que no se debe olvidar es la influencia del estado superficial del objeto de estudio, que puede modificar los valores de temperatura obtenidos.

Para el caso de reconocimiento de personas, las imágenes térmicas son empleadas normalmente por la noche o en lugares con mala iluminación, ya que por el día el contraste térmico entre el ambiente y las personas es insuficiente, lo que conlleva a malos resultados [27]. Esto es debido a que la temperatura del cuerpo en ambientes de exterior (con ropa, al aire libre) ronda entre los 20 y 25 °C. Cuando la temperatura ambiente es similar a esta, no se genera apenas contraste entre el ambiente y la persona, lo que dificulta la detección.

Hoy en día en multitud de aplicaciones se utilizan ambos tipos de cámara simultáneamente, con equipos con óptica dual, en aplicaciones como vigilancia y supervisión de perímetros de alta seguridad [33] y reconocimiento de expresiones faciales.

3.2. Deep Learning y Visión Artificial

3.2.1. Fundamentos del Deep Learning - Redes Neuronales

Una red neuronal artificial es un modelo computacional para la representación por capas de datos, inspirado en el funcionamiento de un cerebro. Consiste en una serie de nodos, o también llamados neuronas artificiales, conectadas entre sí mediante enlaces y organizadas en capas, de manera que se transmiten señales entre ellas [34]. Las señales de entrada a la red se transmiten por esta, donde se someten a diversas operaciones, hasta producir las señales de salida correspondientes. La figura 22 muestra un ejemplo de la estructura de una red neuronal.

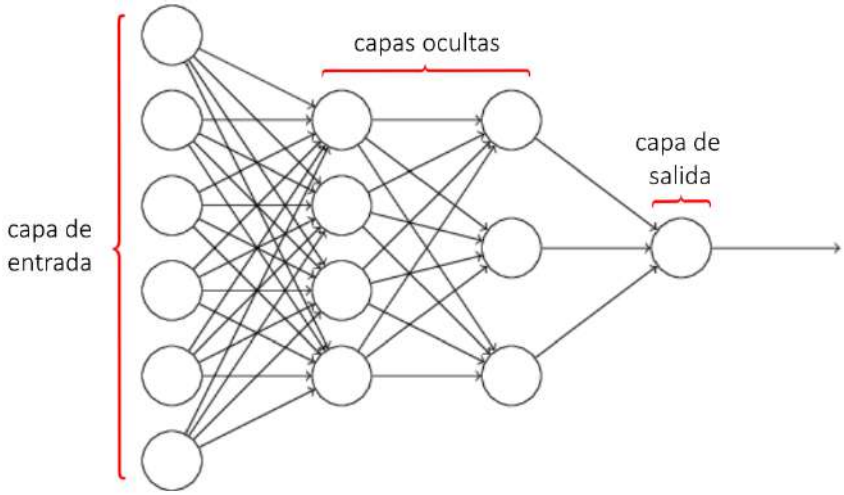


Figura 22: Estructura red neuronal. Fuente: Autor

El objetivo principal de este tipo de redes es que aprendan por sí mismas a realizar una o varias determinadas tareas, modificando sus parámetros internos, para llegar a automatizarlas.

Para simplificar la explicación se va a tomar como ejemplo una red de dos capas, una con n neuronas y otra con solo una tal y como se muestra en la figura 23

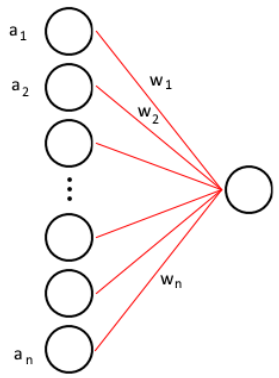


Figura 23: Red neuronal básica. Fuente: Autor

Se puede entender cada neurona como un elemento que almacena un valor, concretamente entre 0 y 1, dependiendo de la entrada de la red. Este número representa la activación de dicha neurona (a), es decir, cuánto de encendida o de apagada está. Esta activación influirá en la activación de las neuronas en las capas sucesivas, según cómo estén conectadas.

Las conexiones entre neuronas son representadas con una variable llamado peso (w), un valor numérico que representa la influencia que tiene un nodo sobre otro en la siguiente capa. Estos pueden ser tanto positivos como negativos con diferentes magnitudes, indicando la tendencia que tiene una neurona de activar o desactivar la de la siguiente capa en mayor o menor medida.

De esta manera, para saber cómo se activará una neurona conociendo las activaciones de la capa anterior y el valor de los pesos correspondientes, basta con hacer una suma ponderada de las activaciones de acuerdo a sus respectivos pesos.

$$a_1 \cdot w_1 + a_1 \cdot w_1 + \cdots + a_n \cdot w_n \tag{5}$$

Esta suma puede dar cualquier valor, pero lo que interesa es que dicha activación esté en el rango de 0 a 1. Es por ello que se necesita una función que reduzca el rango del eje real al rango de valores de 0 a 1. Hay multitud de funciones que realizan esta tarea, pero la que se suele utilizar es la función sigmoide, representada en la figura 24.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

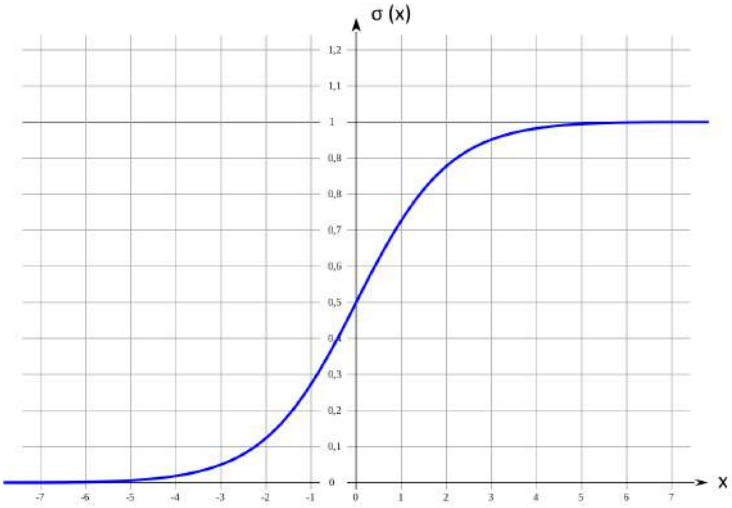


Figura 24: Función sigmoide. Fuente: Autor

Esta función además permite que, para un pequeño cambio en los parámetros como son los pesos o los biases (que se explicarán a continuación), los cambios en la salida también sean pequeños. Esto facilita el ajuste gradual de la red al ser entrenada para acercarse al comportamiento deseado [35].

En la actualidad se utilizan también otras funciones como la unidad lineal rectificada, conocidad como ReLU por sus siglas en inglés, que a pesar de no reducir el rango de activación de 0 a 1, acelera considerablemente el proceso de entrenamiento de la red. La función se expresa como $ReLU(a) = max(0, a)$, representada en la figura 25

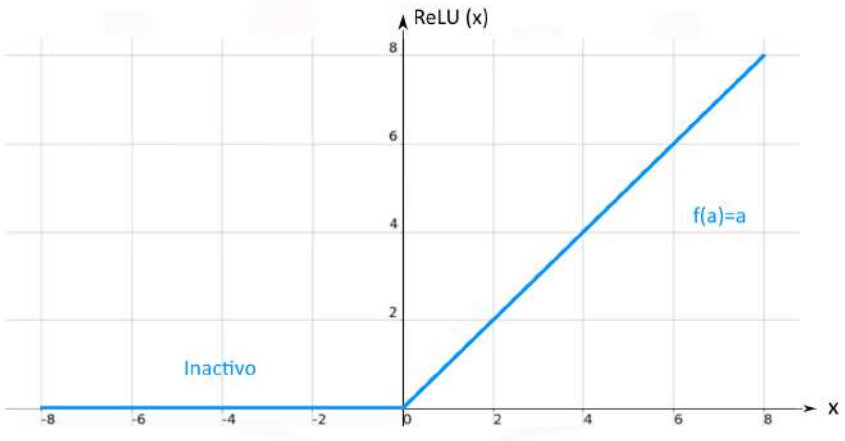


Figura 25: Función ReLU. Fuente: Autor

Por tanto, la activación de la neurona es básicamente la medida de cuan positiva es la suma ponderada.

$$\sigma(a_1 \cdot w_1 + a_2 \cdot w_2 + \cdots + a_n \cdot w_n) \tag{7}$$

Sin embargo, en numerosas ocasiones se requiere que una neurona no se active hasta que la suma no alcanza un cierto umbral, llamado sesgo de inactividad (b), también conocido como "bias" en la literatura inglesa. Este valor representa cuan elevada debe de ser la suma ponderada antes de que la neurona se active significadamente, es decir, indica la tendencia de esa neurona a ser activa o inactiva. Esta variable se añade a la suma ponderada a la que se aplica la función sigmoide.

$$a_0{}^{(1)} = \sigma(w_{0,0} \cdot a_0{}^{(0)} + w_{0,1} \cdot a_1{}^{(0)} + \cdots + w_{0,n} \cdot a_n{}^{(0)} + b_0) \tag{8}$$

En la ecuación 8, para las activaciones los subíndices identifican las neuronas dentro de una capa y los superíndices indican la capa a la que pertenece. En el caso de los pesos, el subíndice hace referencia a la conexión entre neuronas, donde el primer elemento es la neurona de la capa posterior de la conexión y el segundo la neurona de la capa anterior. Finalmente, el subíndice del bias indica a qué neurona pertenece dicho bias.

Cada neurona cuenta con su propio valor de bias, por lo que se trata de un parámetro ajustable que añade complejidad a la red.

La activación de las neuronas se puede representar con notación matricial. Se agrupan los valores de las activaciones de la capa anterior en el vector columna. A continuación, se agrupan en una matriz donde cada fila corresponde con las conexiones entre la capa anterior y una neurona de la capa. Después se organizan los biases en un vector columna que se suma al producto de la matriz y el vector anteriores. Finalmente, solo queda aplicar la función sigmoide a la función anterior.

Por ejemplo, para el caso de una red con dos capas de n y m neuronas respectivamente, la notación para la activación de la segunda capa quedaría de la siguiente manera:

$$\underbrace{\begin{bmatrix} a_0{}^{(1)} \\ a_1{}^{(1)} \\ \vdots \\ a_m{}^{(1)} \end{bmatrix}}_{\mathbf{a^{(1)}}} = \sigma\left(\underbrace{\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,0} & w_{m,1} & \dots & w_{m,n} \end{bmatrix}}_{\mathbf{W}} \underbrace{\begin{bmatrix} a_0{}^{(0)} \\ a_1{}^{(0)} \\ \vdots \\ a_n{}^{(0)} \end{bmatrix}}_{\mathbf{a^{(0)}}} + \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}}_{\mathbf{b}}\right) \tag{9}$$

$$a^{(1)} = \sigma(W \cdot a^{(0)} + b) \tag{10}$$

Al comienzo del apartado se mencionó que se podía tratar a una neurona como un elemento que almacena cierto valor, que depende de la entrada de la red. Realmente, como se acaba de ver, una neurona es una función que toma como parámetros las activaciones de la capa anterior y devuelve una activación de 0 a 1.

3.2.2. Entrenamiento de una red - Backpropagation

La adaptación de una red, variando los valores de sus pesos y biases, para desarrollar una aplicación concreta se le conoce como entrenar la red. Para ilustrar cómo se entrena una red se va a utilizar como ejemplo la clasificación de imágenes de números. Para entrenar una red hace falta un conjunto de datos para que la red aprenda, llamados datos de entrenamiento o training data set en inglés. Para esta tarea existe un conjunto de datos de entrenamiento llamado MNIST data set [35]. Se trata de un conjunto de más de 60000 imágenes, en escala de grises de 28x28 pixeles, en los que aparecen imágenes escaneadas de dígitos a mano junto a los dígitos que representan.

La red consiste en una capa de entrada de 784 neuronas, cada una asociada a un píxel de la imagen. La activación de cada una de ellas está asociada con el valor en la escala de grises en una escala de 0 a 1 del píxel. Posteriormente se encuentran una serie de capas ocultas con un determinado número de neuronas. La cantidad de estas capas y neuronas en estas varía arbitrariamente dependiendo de la aplicación, siendo labor del ingeniero hallar la configuración óptima para cada caso. La última capa es la de salida, en la que se encuentran 10 neuronas, en donde cada una simboliza los dígitos del 0 al 9 que se desean clasificar.

Para entrenar una red se comienza inicializando los pesos y biases aleatoriamente. Al introducir los datos de entrenamiento en la red se obtendrán resultados no deseados completamente aleatorios.

Por ejemplo, para el caso introducir una imagen de un 3 a mano, se activan las neuronas de la capa de salida de manera aleatoria, mientras que lo que se busca es que se active la neurona asociada al dígito 3, como se muestra en la figura 26.

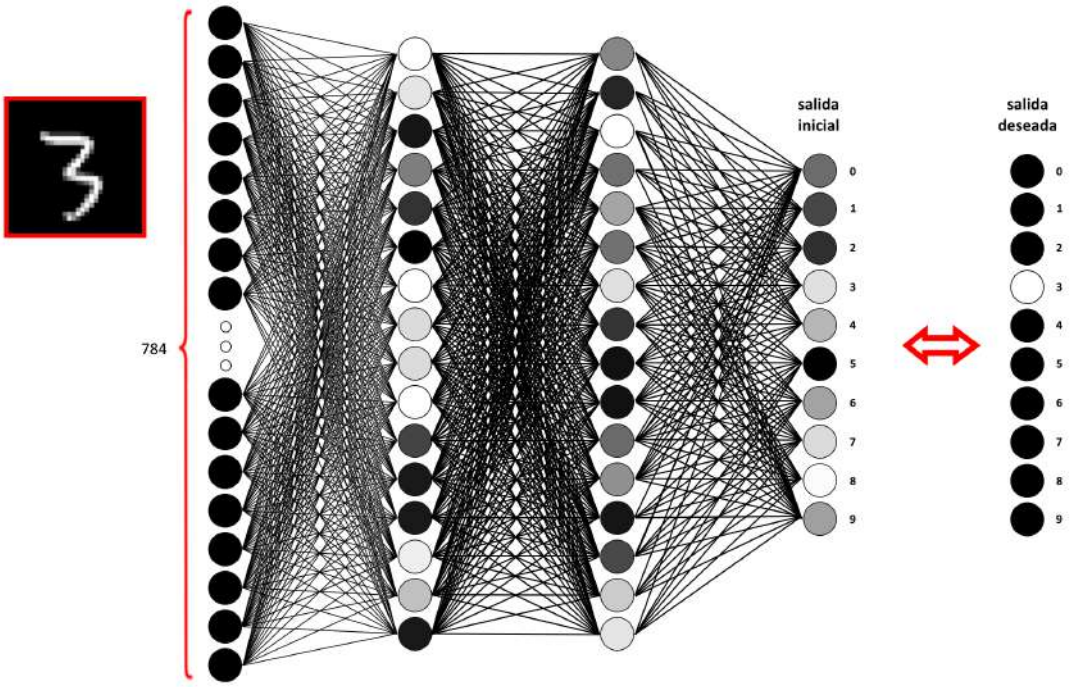


Figura 26: Red neuronal para clasificación de números inicializada. Fuente: Autor

Es por ello que se define una función de coste, normalmente cuadrática, en donde se suma el cuadrado de las diferencias entre las activaciones de la capa de salida actuales y las que deben de tenerse. Una función de coste básicamente cuantifica la desviación de una salida que predice la red neuronal de la esperada [36]. Aplicando esto último a un dato de entrenamiento x da como resultado un valor numérico de coste que se conoce como coste del ejemplo de entrenamiento.

$$C_x(w,b) \equiv \frac{1}{2}\left\|a^{(L)} - y\right\|^2 = \frac{1}{2}\sum_j (a_j^{(L)} - y_j)^2 \tag{11}$$

En la expresión 11, w y b representan todos los pesos y biases de la red, respectivamente, a^L las activaciones de la última capa (capa L) e y las activaciones deseadas en esta capa.

A mayor cercanía del valor de las activaciones a las que se desean, menor es el valor del coste para ese ejemplo. Posteriormente, se toman los costes de todos los datos de entrenamiento y se hace la media, obteniendo el coste medio.

$$C(w,b) \equiv \frac{1}{2n}\sum_x^n \left\|a^{(L)} - y(x)\right\|^2 \tag{12}$$

En la expresión 12, x representa el índice de cada dato de entrenamiento y n es el número total de estos.

Por tanto, el objetivo para entrenar una red es hallar la combinación de pesos y biases tal que el coste sea mínimo, optimizando la red. Los algoritmos que se encargan de actualizar los parámetros de la red para minimizar el coste se llaman optimizadores, y el más común es un algortimo llamado descenso del gradiente.

Normalmente para hallar el mínimo de una función analíticamente se recurre al cálculo, aplicando derivadas y así hallar el extremo. Esto es relativamente sencillo para pocas variables, sin embargo, las redes neuronales cuentan con miles o incluso millones de pesos y biases que actúan como variables en la función de coste, por lo que el cálculo es extremadamente complejo.

Por ello se recurre a algoritmos más sencillos, como es descenso del gradiente. Este algoritmo parte de la analogía de tratar a la función como un valle donde hay una pelota, que parte de un punto aleatorio, que desciende por este hasta llegar al punto más bajo.

Matemáticamente hablando, el descenso del gradiente es un algoritmo iterativo que empieza en un punto aleatorio de la función de coste y desciende por esta a través de su pendiente en pequeños pasos, calculando su derivada para nuevo punto, hasta hallar un mínimo local de la función. Las derivadas de la función dan información de la "forma" de la función de coste, y con ello, cómo se descenderá. La figura 27 muestra un ejemplo del algoritmo descenso del gradiente aplicado a una hipotética función de coste de tan solo un solo parámetro.

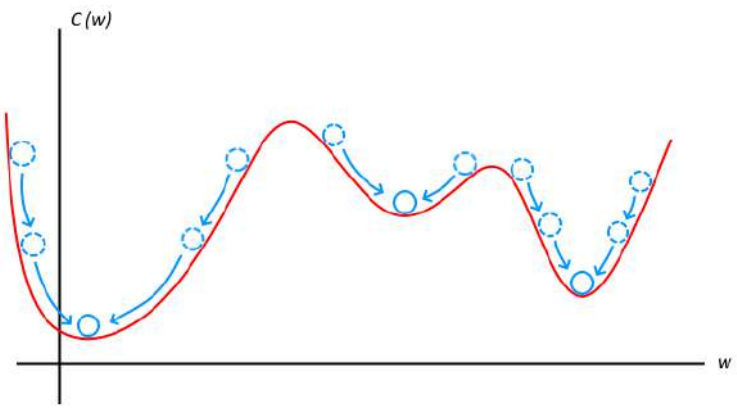


Figura 27: Descenso del gradiente. Fuente: Autor

La expresión matemática que relaciona la variación del coste con la variación de los

pesos y biases es la siguiente:

$$\Delta C \approx \frac{\partial C}{\partial w_0} \cdot \Delta w_0 + \frac{\partial C}{\partial b_0} \cdot \Delta b_0 + \cdots + \frac{\partial C}{\partial w_n} \cdot \Delta w_n + \frac{\partial C}{\partial b_n} \cdot \Delta b_n \tag{13}$$

Si se atiende a la ecuación 13 se puede observar que se deben de encontrar los valores de los incrementos de los pesos y biases tal que desciendan el coste. Para ello conviene definir un vector de incrementos v que recoja los cambios en las variables.

$$\Delta v = (\Delta w_0, \Delta b_0, \ldots, \Delta w_n, \Delta b_n)^T \tag{14}$$

El gradiente de la función de coste se define como el vector de derivadas parciales de este.

$$\nabla C \equiv (\frac{\partial C}{\partial w_0}, \frac{\partial C}{\partial b_0}, \ldots, \frac{\partial C}{\partial w_n}, \frac{\partial C}{\partial b_n})^T \tag{15}$$

Reescribiendo 13 en función de 14 y 15, el incremento de coste se puede expresar como:

$$\Delta C \approx \nabla C \cdot \Delta v \tag{16}$$

Esta expresión permite ver qué Δv hacen $DeltaC$ negativa. Si escogemos un Δv tal que sea $\Delta v = -\eta \cdot \nabla C$, donde η es un parámetro conocido como tasa de aprendizaje o learning rate en literatura inglesa, la variación de l coste queda:

$$\Delta C \approx -\eta \nabla C \cdot \nabla C = -\eta \, \|\nabla C\|^2 \tag{17}$$

Como $\|\nabla C\|^2$, $\Delta C \leq 0$, por lo que se asegura que el coste siempre decrece. La tasa de aprendizaje debe de ser pequeña para que se cumpla la aproximación 16, pero no demasiado, o de lo contrario el descenso del gradiente será muy lento.

Para cada paso de este algoritmo, las variarán según $v' = v - \eta \nabla C$. Aplicando esta fórmula a los pesos y biases, se obtiene ${w_k}' = w_k - \eta \frac{\partial C}{\partial w_k}$ y ${b_l}' = b_l - \eta \frac{\partial C}{\partial b_l}$ respectivamente. Repitiendo esta operación se logrará alcanzar el mínimo local de la función de coste.

En la práctica, realizar todo esto para todos los ejemplos de entrenamiento y luego hacer la media requiere mucho tiempo. Una solución a ello es lo que se conoce como *descenso del gradiente estocástico* [35]. La idea es mezclar aleatoriamente los datos de entrenamiento y dividirlos en lotes (batches en inglés), de manera que se calcula ∇C para cada lote. Para los m datos de entrenamiento de un lote, $X_1 + X_2 + \cdots + X_m$, no se obtendrá el gradiente real, pero sí una aproximación válida:

$$\nabla C = \frac{1}{n} \sum_x \nabla C_x \approx \frac{1}{m} \sum_{j=1}^{m} \nabla C_{X_j} \tag{18}$$

Una vez visto todo esto, ya se sabe cómo minimizar la función de coste utilizando el algoritmo de descenso del gradiente. Sin embargo, ha quedado un vacío en la explicación: hasta ahora no se ha mostrado cómo obtener ∇C. El algoritmo que se encarga de obtener el gradiente de la función de coste se le conoce como Backpropagation.

El objetivo de Backpropagation es calcular las derivadas parciales $\frac{\partial C}{\partial w}$ y $\frac{\partial C}{\partial b}$ de cada peso y bias de la red. Para utilizarlo se deben de asumir dos cuestiones.

La primera es asumir que la función de coste se puede expresar como una media $C = \frac{1}{n}\sum_x C_x$ de los costes de cada entrenamiento C_x . Esto lo verifica la función de coste cuadrática ya que $C_x = \frac{1}{2}\left\|a^{(L)} - y\right\|^2$ y $C = \frac{1}{2n}\sum_x^n \left\|a^{(L)} - y(x)\right\|^2$, tal y como se vio en la definición de función de coste.

Esta condición sirve para poder calcular las derivadas parciales de los pesos y biases para un único ejemplo de entrenamiento, $\frac{\partial C_x}{\partial w}$ y $\frac{\partial C_x}{\partial b}$, y recuperar $\frac{\partial C}{\partial w}$ y $\frac{\partial C_x}{\partial b}$ haciendo la media sobre los ejemplos de entrenamiento.

La segunda es asumir que el coste se puede expresar en función de la salida de la red: $C = C(a^{(L)})$. Esto también se verifica para una función de coste cuadrática, ya que, como se ha visto, $C_x = \frac{1}{2}\sum_j (a_j^{(L)} - y_j)^2$. El coste no es función de la salida deseada y, aunque no parezca evidente a primera vista, ya que $y$es un valor prefijado según el ejemplo de entrenamiento [37].

Para ilustrar cómo funciona backpropagation, se van a tomar dos neuronas de una red formada por una sola neurona por capa, una de la capa final de una red (L) y otra de la anterior (L-1), tal y como se muestra en la figura 28.

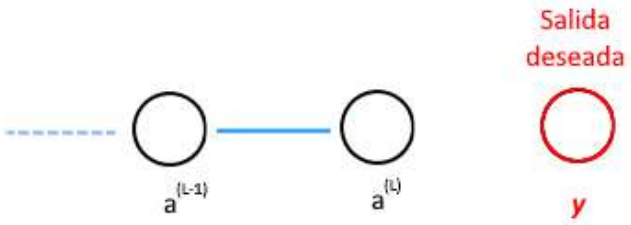


Figura 28: Red con una neurona por capa. Fuente: Autor

La activación de la última capa ($a^{(L)}$) está determinada por el peso ($w^{(L)}$) multiplicado por la activación de la neurona anterior ($a^{(L-1)}$), sumado a un bias ($b^{(L)}$), introducido a la función de activación correspondiente:

$$a^{(L)} = \sigma(w^{(L)} \cdot a^{(L-1)} + b^{(L)}) = \sigma(z^{(L)}) \tag{19}$$

De la última igualdad de la expresión 19, se define z como la entrada ponderada a la neurona en la capa L:

$$z^{(L)} = w^{(L)} \cdot a^{(L-1)} + b^{(L)} \tag{20}$$

De esta manera, el peso w, la activación de la neurona anterior $a^{(L-1)}$. y el bias b se usan para obtener la entrada pondesada z, que a su vez se utiliza para obtener la activación de la neurona $a^{(L)}$. Juntando esta última activación con la salida deseada y se puede calcular el coste. De la misma manera, $a^{(L-1)}$ está influenciada por su peso, activación anterior y su bias, y así sucesivamente. En la figura 29 se muestra un esquema que recoge estas dependencias.

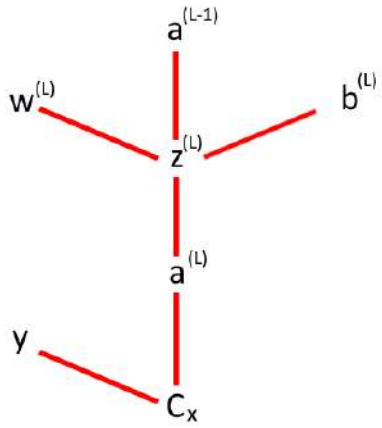


Figura 29: Esquema de las dependencias de la función de coste. Fuente: Autor

Para ver cuan sensible es la función de coste a cambios en el peso, activaciones de la capa anterior y bias, se deriva parcialmente el coste respecto a estas variables.

La derivada parcial del coste respecto al peso, usando la regla de la cadena, es la siguiente:

$$\frac{\partial C_x}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdot \frac{\partial C_0}{\partial a^{(L)}} \tag{21}$$

Utilizando las definiciones 11, 20 y 19, se puede calcular la derivada parcial del coste respecto del peso calculando las derivadas parciales obtenidas de la regla de la cadena de la expresión 21.

$$\frac{\partial C_x}{\partial a^{(L)}} = (a^{(L)} - y) \qquad \frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'(z^{(L)}) \qquad \frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)} \tag{22}$$

Sustituyendo las derivadas parciales en 22 en 21 se obtiene lo siguiente:

$$\frac{\partial C_x}{\partial w^{(L)}} = a^{(L-1)} \cdot \sigma'(z^{(L)}) \cdot (a^{(L)} - y) \tag{23}$$

Teniendo en cuenta que el coste medio es la media de todos los costes de cada ejemplo de entrenamiento, la derivada parcial del coste medio respecto al peso es la siguiente:

$$\frac{\partial C}{\partial w^{(L)}} = \frac{1}{n} \cdot \sum_{x=0}^{n-1} \frac{\partial C_x}{\partial w^{(L)}} \tag{24}$$

La derivada parcial del coste respecto al bias y la derivada parcial del coste respecto a la activación de la capa anterior, usando la regla de la cadena y empleando las expresiones 22, son las siguientes:

$$\frac{\partial C_x}{\partial b^{(L)}} = \frac{\partial z^{(L)}}{\partial b^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdot \frac{\partial C_0}{\partial a^{(L)}} = \sigma'(z^{(L)}) \cdot (a^{(L)} - y) \tag{25}$$

$$\frac{\partial C_x}{\partial a^{(L-1)}} = \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdot \frac{\partial C_0}{\partial a^{(L)}} = w^{(L)} \cdot \sigma'(z^{(L)}) \cdot (a^{(L)} - y) \tag{26}$$

Las expresiones 23 y 25 aplicadas a cada capa de la red son utilizadas para formar el gradiente de la función de coste, como se vio en su definición en 15.

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w^{(1)}} \\ \frac{\partial C}{\partial b^{(1)}} \\ \vdots \\ \frac{\partial C}{\partial w^{(L)}} \\ \frac{\partial C}{\partial b^{(L)}} \end{bmatrix} \tag{27}$$

Para una red con múltiples neuronas por cada, como la de la figura 30, las derivadas parciales se realizan de manera casi idéntica.

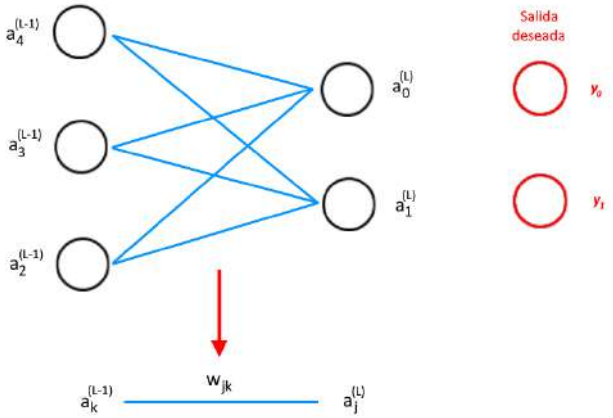


Figura 30: Red con múltiples neuronas por capa. Fuente: Autor

Ahora las definiciones de coste de entrenamiento, entrada ponderada y activación de la neurona serán las siguientes:

$$C_x = \frac{1}{2}\sum_{j=0}^{n_L-1}(a_j{}^{(L)} - y_j)^2 \tag{28}$$

$$z_j{}^{l} = w_{j0}{}^{(l)} \cdot a_0{}^{(l-1)} + w_{j1}^{(l)} \cdot a_1{}^{(l-1)} + w_{j2}^{(l)} \cdot a_2{}^{(l-1)} + \cdots + b_j{}^{(l)} = \sum_{j=0}^{n_l-1} w_{jk}{}^{(l)} \cdot a_k{}^{(l-1)} + b_j{}^{(l)} \tag{29}$$

$$a_j{}^{l} = \sigma(z_j{}^{l}) \tag{30}$$

Las derivadas parciales homólogas a 22 para este caso serán:

$$\frac{\partial C_x}{\partial a_j^{(L)}} = (a_j^{(L)} - y_j) \qquad \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} = \sigma'(z_j^{(l)}) \qquad \frac{\partial z_j^{(l)}}{\partial w_{jk}^{(l)}} = a_j^{(l-1)} \tag{31}$$

Se define la variable intermedia error de la neurona j en la capa L o gradiente local de la neurona como la derivada parcial de la función de coste frente a la entrada ponderada de la misma.

$$\delta_j^{(l)} = \frac{\partial C_x}{\partial z_j^{(l)}} = \frac{\partial C_x}{\partial a_j^{(l)}} \cdot \sigma'(z_j{}^{(l)}) \tag{32}$$

Las derivadas de la función de coste respecto al peso, bias y activación de la neurona de la capa anterior, expresados en función de 32 son los siguientes:

$$\frac{\partial C_x}{\partial w_{jk}{}^{(l)}} = \frac{\partial z_j{}^{(l)}}{\partial w_{jk}{}^{(l)}} \cdot \frac{\partial a_j{}^{(l)}}{\partial z_j{}^{(l)}} \cdot \frac{\partial C_x}{\partial a_j{}^{(l)}} = a_k{}^{(l-1)} \cdot \sigma'(z_j{}^{(l)}) \cdot \frac{\partial C_x}{\partial a_j{}^{(l)}} = a_k{}^{(l-1)} \cdot \delta_j^{(l)} \tag{33}$$

$$\frac{\partial C_x}{\partial b_j{}^{(l)}} = \frac{\partial z_j{}^{(l)}}{\partial b_j{}^{(l)}} \cdot \frac{\partial a_j{}^{(l)}}{\partial z_j{}^{(l)}} \cdot \frac{\partial C_x}{\partial a_j{}^{(l)}} = \sigma'(z_j{}^{(l)}) \cdot \frac{\partial C_x}{\partial a_j{}^{(l)}} = \delta_j^{(l)} \tag{34}$$

$$\frac{\partial C_x}{\partial a_k{}^{(l-1)}} = \sum_{j=0}^{n_l-1} \frac{\partial z_j{}^{(l)}}{\partial a_k{}^{(l-1)}} \cdot \frac{\partial a_j{}^{(l)}}{\partial z_j{}^{(l)}} \cdot \frac{\partial C_x}{\partial a_j{}^{(l)}} = \sum_{j=0}^{n_l-1} w_{jk}{}^{(l)} \cdot \sigma'(z_j{}^{(l)}) \cdot \frac{\partial C_x}{\partial a_j{}^{(l)}} = \sum_{j=0}^{n_l-1} w_{jk}{}^{(l)} \cdot \delta_j^{(l)} \tag{35}$$

En la derivada parcial 35 el sumatorio aparece debido a que hay que sumar la influencia de la activación de la neurona de la capa l-1 ($a_k^{(l-1)}$) sobre todas las activaciones en las neuronas de la capa l. Esto puede verse en la figura 31.

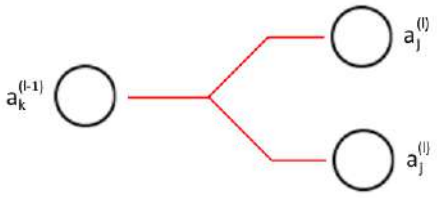


Figura 31: Influencia de la activación de una neurona sobre las de la siguiente capa. Fuente: Autor

La neurona a_k^{L-1} influencia al coste por varios caminos. En la figura 31 se puede observar que a_k^{L-1} afecta tanto a a_0^L como a a_1^L, ambos influyendo en la función de coste.

Los valores que forman el gradiente, como se acaba de ver en 33 y 34, dependen de los errores de todas las neuronas. Para la última capa es sencillo, puesto que $\delta_j^{(L)} = \frac{\partial C_x}{\partial z_j^{(L)}} = \frac{\partial C_x}{\partial a_j^{(L)}} \cdot \sigma'(z_j{}^{(L)}) = \sigma'(z_j{}^{(L)}) \cdot (a_j^{(L)} - y_j)$, pero para obtener el valor del error en el resto de capas hace falta propagar hacia atrás el error.

Esto se hace obteniendo el error de las neuronas de una capa en función del error de las neuronas de la siguiente.

$$\delta_j^{(l)} = \frac{\partial C_x}{\partial z_j^{(l)}} = \sum_k \frac{\partial C_x}{\partial z_k^{(l+1)}} \frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} = \sum_k \frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} \delta_j^{(l+1)} \tag{36}$$

Para resolver 36 se toma la entrada ponderada de una neurona en la capa *(l+1)*.

$$z_k^{(l+1)} = \sum_j w_{kj}^{(l+1)} a_j^{(l)} + b_k^{(l+1)} = \sum_j w_{kj}^{(l+1)} \sigma(z_j^{(l)}) + b_k^{(l+1)} \tag{37}$$

Derivando parcialmente 37 respecto a la entrada ponderada de una neurona en la capa *(l)* se obtiene:

$$\frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} = w_{kj}^{(l+1)} \sigma'(z_j^{(l)}) \tag{38}$$

Sustituyendo 38 en 36 se obtiene finalmente el error de las neuronas de una capa en función del error de las neuronas de la siguiente como:

$$\delta_j^{(l)} = \sum_k w_{kj}^{(l+1)} \delta_k^{(l+1)} \sigma'(z_j^{(l)}) \tag{39}$$

De esta forma, repitiendo estas operaciones a cada capa, se obtienen todos los errores de las neuronas y con ello los componentes del gradiente, como se ve en la figura 32

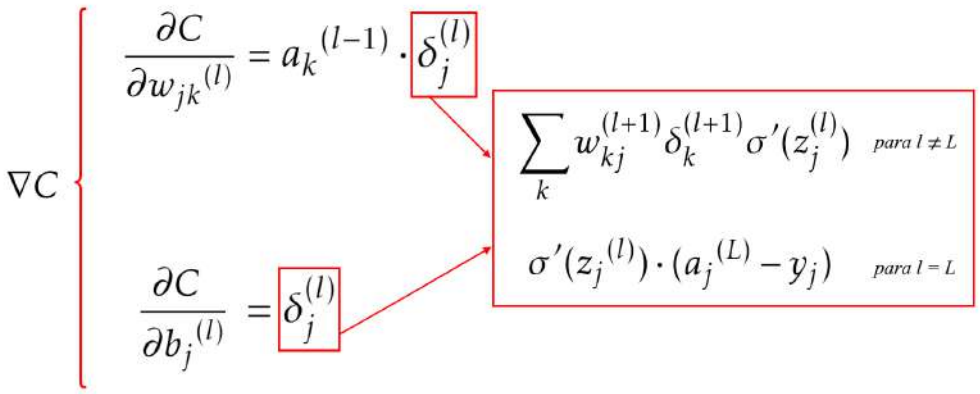


Figura 32: Gradiente función de coste. Fuente: Autor

El algoritmo Backpropagation queda finalmente de la siguiente manera [37]:

1. Poner el ejemplo de entrenamiento x como entrada: así se obtienen las activaciones de la primera capa $a^{(1)}$.
2. Propagación: Para las capas $l = 2, 3, \ldots, L$ calcular $z^{(l)}$ y $a^{(l)}$.
3. Calcular el error de las neuronas de la última capa: $\delta^{(L)} = \sigma'(z_j{}^{(L)}) \cdot (a_j^{(L)} - y_j)$
4. Propagar hacia atrás ese error (Backpropagate): $\delta_j^{(l)} = f(\delta_l^{(l+1)})$
5. Obtener el gradiente: el gradiente viene dado por $\frac{\partial C_x}{\partial w_{jk}{}^{(l)}} = a_k{}^{(l-1)} \cdot \delta_j^{(l)}$ y $\frac{\partial C_x}{\partial b_j{}^{(l)}} = \delta_j^{(l)}$

Cabe mencionar que existen multitud de redes neuronales pre-entrenadas de las que partir de base para entrenar un sistema. Esto ahorra una gran cantidad de tiempo y recursos computacionales. La adaptación de una red pre-entrenada a una determinada aplicación se conoce como fine tuning [38].

3.3. Visión Artificial y Redes Neuronales Convolucionales

La Visión Artificial, o Computer Vision en inglés, es un campo de Machine Learning que se dedica a adquirir, procesar y analizar imágenes para que lo ordenadores puedan llegar a ver y comprender el contenido de estas.

La visión artificial cuenta con multitud de aplicaciones, como pueden ser el análisis del movimiento, reconstrucción de escenarios en 3D a partir de imágenes bidimensionales, restauración de imágenes, etc. Sin embargo, la aplicación más relevante de la visión artificial es el reconocimiento de imágenes. Dentro del área del reconocimiento hay subapartados como pueden ser la clasificación, identificación y detección de imágenes

Actualmente los mejores algoritmos para desarrollar estas últimas tareas consisten en emplear redes neuronales convolucionales. Una red neuronal convolucional (CNN o ConvNet) es un tipo de red neuronal diseñado específicamente para trabajar con imágenes [39].

Hasta cierto punto, la visión artificial se basa en el reconocimiento de patrones. Las redes neuronales densas, es decir, las convencionales que se ha visto en los apartados anteriores, cuentan con capas formadas por neuronas donde cada una está conectada a todas las neuronas de la capa anterior. Cada neurona de esta red ve toda la información, por lo que la red es únicamente capaz de analizar la información globalmente. Estas redes buscan patrones en el área específica de la imagen donde estaba dicho patrón en el entrenamiento, por lo que si se analiza una imagen que cuenta con dichos patrones en lugares diferentes, no los detectará.

Otro aspecto importante derivado del hecho de que las neuronas de una capa estén conectadas a todas las de la capa anterior y de que no estén conectadas entre sí dentro de la misma capa es que, por ejemplo, para el caso de trabajar con imágenes como entrada, estas redes son ineficientes computacionalmente para abordar el problema. Poniendo como ejemplo una imagen de entrada RGB 200x200x3 (200 píxeles de ancho, 200 píxeles de alto y 3 canales de color), cada neurona de la primera capa oculta tras la de entrada tendrá $200x200x3 = 120000$ pesos. Sumando los pesos de todas las conexiones de la red, se puede intuir que el número de parámetros en la red será enorme. Esta conectividad completa es ineficiente, malgastando gran cantidad de recursos computacionales en el proceso, además de generar problemas adversos como sobre-entrenamiento de estos parámetros.

Para solucionar estos problemas se adoptó la red a una configuración de capas tridimensionales, de manera que las neuronas se conectan localmente, formando lo que se conoce como red neuronal convolucional. Así se consigue que se pase de detectar patrones globalmente a detectarlos localmente, siendo esta la diferencia fundamental entre las redes convolucionales y las densas. Para el caso de la imagen de entrada, como se ha pasado a una red tridimensional, ahora se considera como un volumen de entrada a la red. Las neuronas en una capa ahora se conectarán con solo una pequeña región de la capa anterior. Cada capa transformará este volumen, teniendo la capa final dimensiones 1x1xN, siendo N el número de clases a detectar, dando como salida un vector con puntuaciones de clase. Estas puntuaciones son utilizadas para clasificar las detecciones. La comparativa de la estructura de las redes densas y las CNNs se puede ver en la figura 33.

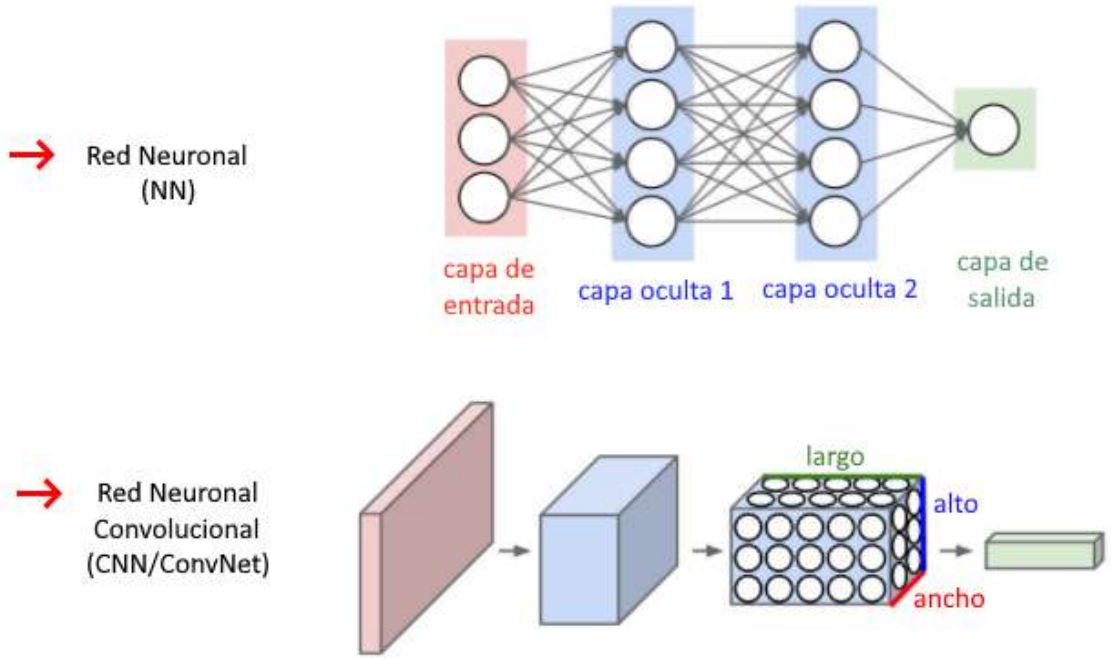


Figura 33: Comparativa de la estructura de NN y CNN. Adaptado de [39]

Un concepto importante es el de vectores o mapas de características. Estos son tensores tridimensionales con los que trabajan las capas convolucionales, que recogen, como su nombre indica, características concretas de una región de la imagen, obtenidas de las activaciones de salida de un determinado filtro. Las capas convolucionales toman estos mapas como entrada y devuelven como salida otro mapa de características llamado mapa de respuesta, que indica la mayor o menor presencia de un determinado filtro en el mapa de entrada.

Un filtro es un patrón de mxn píxeles que se buscan en una imagen. El número de estos filtros representa cuantos patrones se buscan por cada capa, determinando la dimensión profunda del mapa de respuesta. Cada capa de profundidad del mapa es una matriz que contiene valores que indican si el filtro está presente o no en el lugar donde se mira. El tamaño de estos filtros puede variar. Las capas convolucionales funcionan deslizando estos filtros sobre la imagen dando como resultado el mapa de características/respuesta, como se ve en la figura 34.

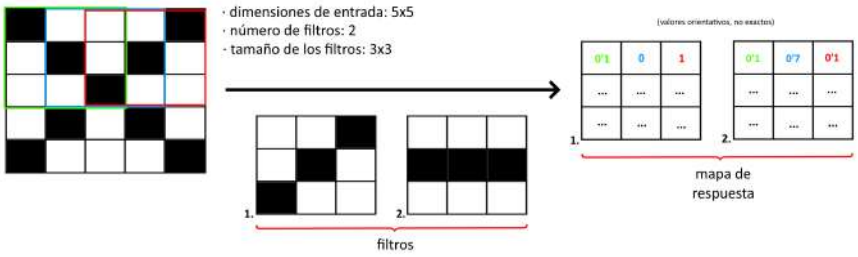


Figura 34: Filtros y mapas de respuesta. Fuente: Autor.

En la figura 34 se ve que para una imagen de 5x5 píxeles y el uso de dos filtros, se obtiene un mapa de respuesta de tamaño 3x3 y profundidad 2 (el número de filtros determina la profundidad del mapa de respuesta). La comparación de una región de la imagen y el filtro se realiza con operaciones como pueden ser el producto escalar de los valores de los píxeles de la imagen con los del filtro, introduciendo el resultado en la posición correspondiente del mapa de respuesta.

Al utilizar filtros en los mapas de características, se obtienen mapas de respuesta de menor tamaño que los de características, como se acaba de ver. En numerosas ocasiones interesa que el mapa de respuesta sea del mismo tamaño que el de características, por lo que se utiliza una técnica llamada padding. Padding consiste en añadir filas y columnas vacías a los datos de entrada para que cada píxel pueda estar en el centro del filtro. Esto puede verse en la figura 35.

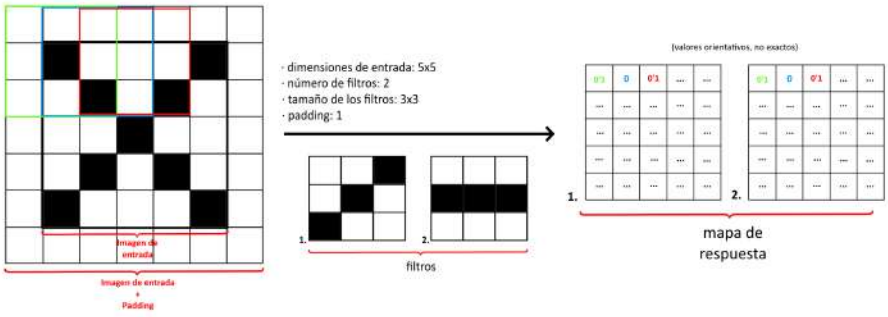


Figura 35: Ejemplo de padding. Fuente: Autor.

Existen una serie de hiperparámetros que controlan la convolución en estas capas:

- Profundidad: cantidad de filtros que se emplean.
- Paso: paso con el que se deslizan los filtros.
- Cantidad de Padding: cantidad de filas y columnas de ceros con que se rellenan los bordes de una imagen.
- Campo receptivo: cantidad de neuronas con las que se conecta cada neurona dentro de una capa. Es equivalente al tamaño del filtro [39].

Cabe mencionar que las CNNs no consisten únicamente en capas convolucionales, si no que también cuentas con capas densas (completamente conectadas) y con unas capas llamadas capas pooling. Estas últimas capas realizan la operación de pooling, que se describirá con mayor detalle más adelante, que consiste en agrupar valores del mapa de características, reduciendo su dimensión.

Las redes convolucionales suelen contar con arquitecturas que poseen capas alternadas de convolución y pooling, y terminan con capas densas. Un ejemplo se muestra en la figura 36.

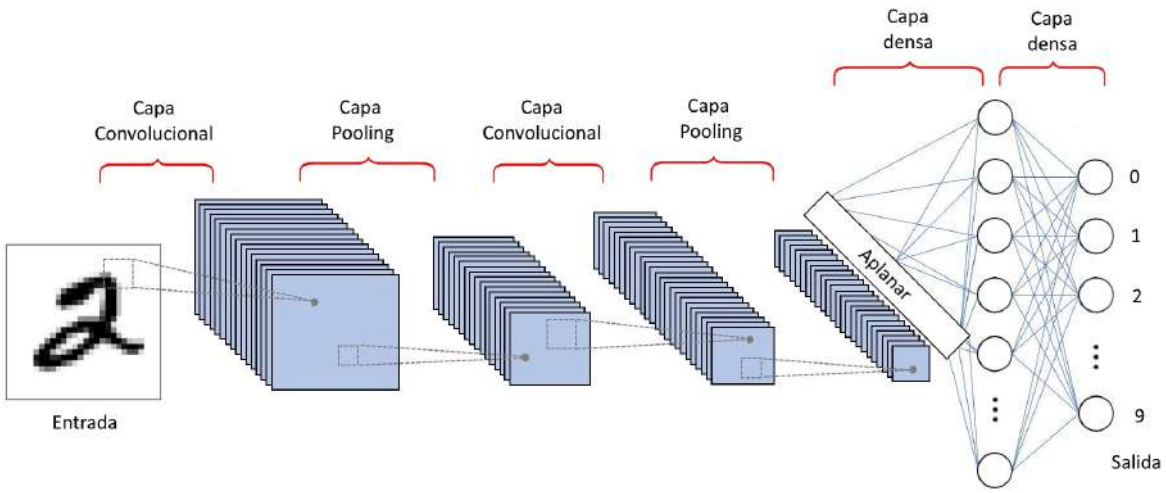


Figura 36: Arquitectura estándar CNN. Adaptado de [40]

Finalmente, para entrenar una red que reconozca imágenes simplemente habría que introducir en ella una gran cantidad de imágenes de entrenamiento con un etiquetado que identifique qué se está reconociendo, y realizar el proceso de entrenamiento con el algoritmo backpropagation. Una vez hecho esto el ordenador simplemente buscará patrones en imágenes que se quiera reconocer mediante una serie de algoritmos y dará como salida el etiquetado correspondiente [41]. En ejemplo de etiquetado de salida es el de la figura 37.

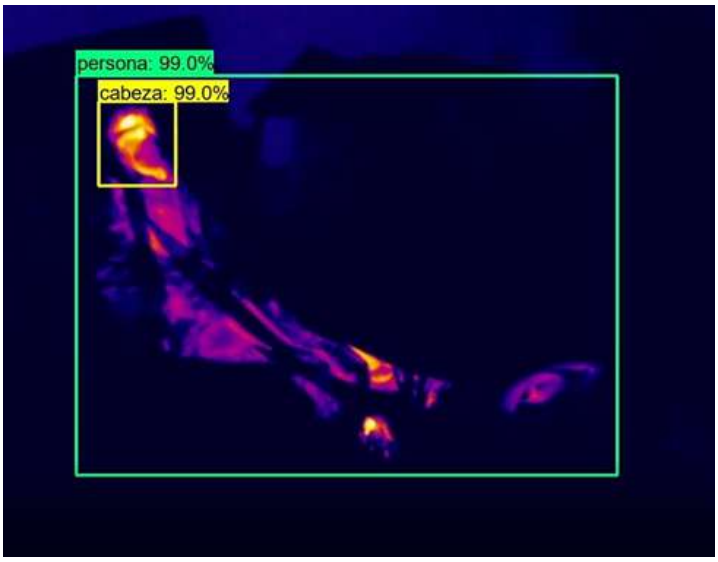


Figura 37: Ejemplo de detección y etiquetado de una víctima en una imagen térmica.
Fuente: Autor

3.3.1. Detección de objetos

Los sistemas de detección de objetos, es como se ha mencionado en el apartado anterior, una técnica de visión artificial que permite identificar y localizar objetos dentro de una imagen o video, recuadrándolos. Estos sistemas son utilizados hoy en día en múltiples tareas, como pueden ser la videovigilancia, reconocimiento facial, contadores de personas o vehículos autónomos.

Al contrario que otros sistemas, los sistemas de detección no se limitan a reconocer y clasificar los distintos objetos en una imagen, sino que también deben de localizarlos y señalarlos dentro de un cuadro delimitador. Esto hace que la detección sea mucho más compleja que tareas como la clasificación [42].

Tradicionalmente las técnicas de detección de objetos en imágenes o videos seguían tres pasos [43], resumidos en la figura 38:

1. Generar propuestas de regiones: Las regiones propuestas son regiones que pueden albergar el objeto a detectar en ella. Algunos algoritmos que realizan esta función se llaman Selective Search y EdgeBoxes, que pueden llegar a generar miles de regiones en una imagen.

2. Extracción de las características: Para cada región propuesta se extrae un vector de características por medio de descriptores de imagen. Este vector debería describir al objeto a pesar de contar con transformaciones como pueden ser la traslación o rotación.

3. Clasificación: Mediante el vector de características, se procede a clasificar las regiones propuestas en dos grupos, la clase ambiente o la clase objeto. Si se busca detectar más de un objeto, se requerirán más clases para cada objeto. Esto dificulta la distinción entre clases. Un modelo de clasificación ampliamente usado es el SVM, máquina de vectores soporte traducido de sus siglas en inglés.

Figura 38: Pasos para la detección de objetos. Fuente Autor

En esencia, lo que se busca al realizar este proceso es transformar el problema de la detección en uno de clasificación.

3.3.1.1. Faster R-CNN

Faster R-CNN es un modelo que pertenece a la familia R-CNN, siglas de Region-based Convolutional Neural Network, que se traduce como red neuronal convolucional basada en regiones.

Para entender Faster R-CNN hace falta primero entender a sus predecesores: R-CNN y Fast R-CNN.

R-CNN

R-CNN es una red neuronal convolucional desarrollada en el año 2014 por un equipo de investigadores en visión artificial. Consiste en una red que es capaz de detectar, localizar y clasificar hasta 80 objetos en imágenes. Su estructura se pueder ve en la figura 39.

Los pasos que sigue la detección de objetos con esta red son los siguientes:

- Se escanea la imagen buscando posibles objetos usando Selective Search, generando unas 2000 regiones por imagen.
- Se ajustan las regiones propuestas a un tamaño predefinido y se pasan por una red convolucional, donde se extrae un vector de características de longitud 4096 para cada región.
- Se utiliza el vector de características para clasificar las regiones propuestas mediante SVM.

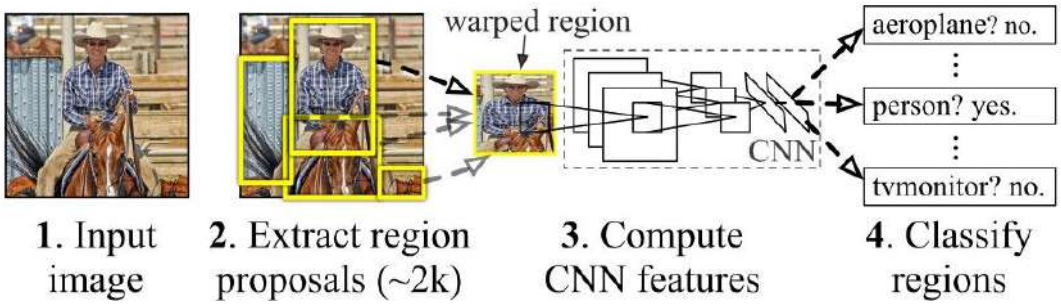


Figura 39: Esquema de R-CNN [43]

Como se puede observar, la diferencia entre R-CNN y la estructura de detección tradicional reside en el empleo de redes neuronales convolucionales en la extracción de características.

Esta red presenta una serie de desventajas, como pueden ser [43]:

- Requiere una cantidad enorme de recursos computacionales, ya que las características extraídas de la red convolucional deben de almacenarse en memoria caché para luego entrenar las SVM, pudiendo llegar a requerirse cientos de gigas.
- Es un modelo con múltiples etapas independientes entre ellas.
- Selective Search es un algoritmo lento y no personalizable para detección.
- Cada región se pasa por la red de manera independiente para la extracción de características, lo que impide la ejecución a tiempo real.

Fast R-CNN

Para solucionar algunos de los problemas que surgieron con R-CNN se desarrolló Fast R-CNN, una mejora a su modelo predecesor, que hace especial énfasis en la mejora de la velocidad.Su arquitectura se muestra en la figura 40. Algunas de dichas mejoras y adaptaciones son las siguientes:

- Se añade una nueva capa detrás de las capas convolucionales llamada RoI Pooling, Region of Interest Pooling por sus siglas en inglés, que extrae vectores de características de igual tamaño de todas las regiones propuestas de la imagen.
- Se reduce de 3 etapas a tan solo una, en la que se toma la imagen como entrada y se devuelve como salida las probabilidades y cuadros delimitadores de los objetos detectados.
- Se realizan los cálculos de todas las regiones de manera simultanea gracias a la nueva capa de RoI Pooling.
- Las características ya no se guardan en memoria caché, por lo que ya no es necesaria una memoria tan grande.
- Se reemplaza SVM con una capa softmax.
- Hay un aumento de precisión en los resultados.

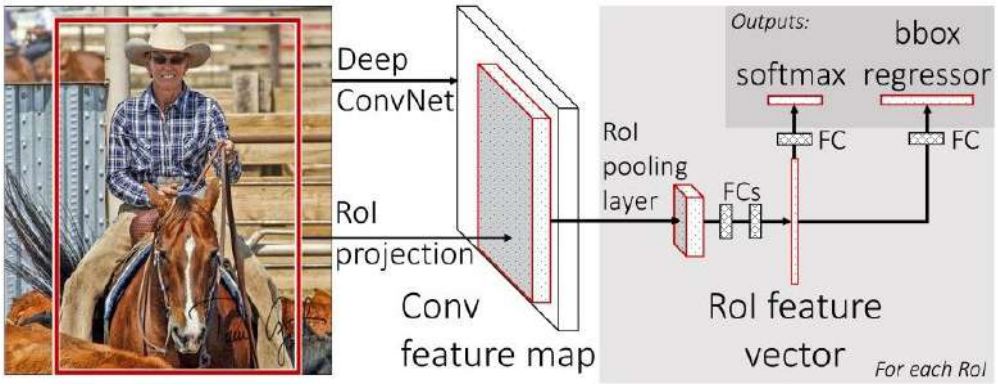


Figura 40: Esquema de Fast R-CNN [43]

El mapa de características de la última capa de la red convolucional se introduce a la capa RoI Pooling. Esta capa separa capa región propuesta en una cuadrícula con secciones de agrupación de celdas. La operación de pooling o agrupación consiste en aplicar una operación a cada sección de la cuadrícula y devolver un único valor para cada una. Las operaciones de pooling más comunes son Average Pooling, donde de calcula el valor medio de las celdas de cada sección, o Max Pooling, donde se obtiene el valor máximo de las celdas de cada sección. En la figura 41 se puede ver un ejemplo de esta operación.

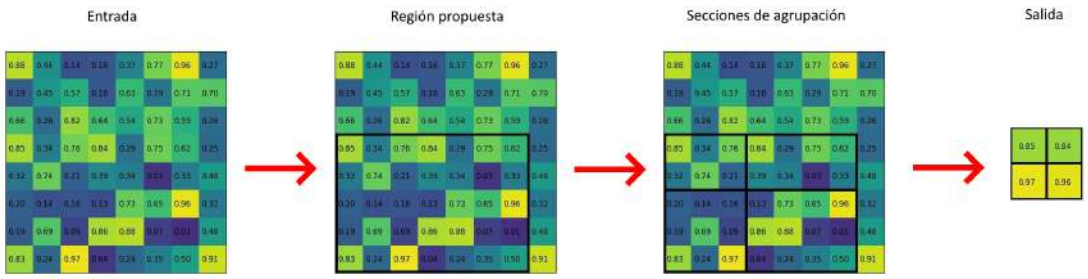


Figura 41: Ejemplo de Max Pooling [44]

Los vectores de características extraídos de esta capa pasan por una serie de capas completamente conectadas y su respectiva salida se divide en dos partes, una que va a otra capa completamente conectada donde se predicen los cuadros delimitadores, y otra que va a la capa softmax, capa que aplica la función softmax de regresión logística, que se muestra en la figura 42. Esta función asigna la probabilidad de 0 a 1 de pertenecer a una clase, por lo que la suma de las probabilidades de todas las clases debe ser siempre 1. La función es de la siguiente manera:

$$P(x)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_j}} \quad para\ j = 1, \ldots, K \tag{40}$$

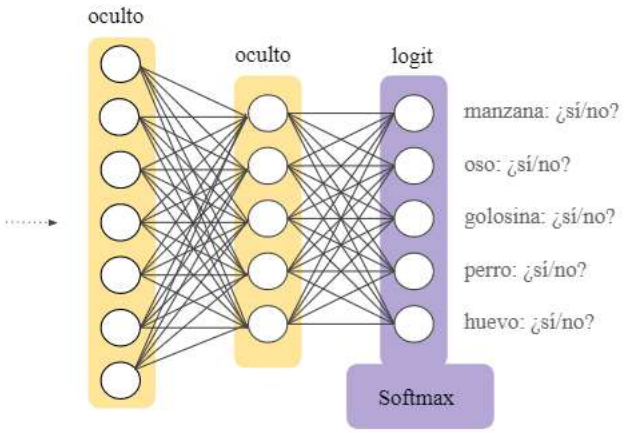


Figura 42: Ejemplo de capa softmax [45]

Como desventaja del Fast R-CNN cabe mencionar que se sigue dependiendo del algoritmo Selective Search para realizar las propuestas de regiones, que relentiza la red, no puede ser personalizado para la tarea de detección y no cuenta con la suficiente precisión para detectar todos los posibles objetos del conjunto de datos.

Faster R-CNN

La principal mejora del Faster R-CNN reside en la sustitución del algoritmo Selective Search por RPN, Region Proposal Network. En la figura 43 se muestra la estructura de Faster R-CNN.

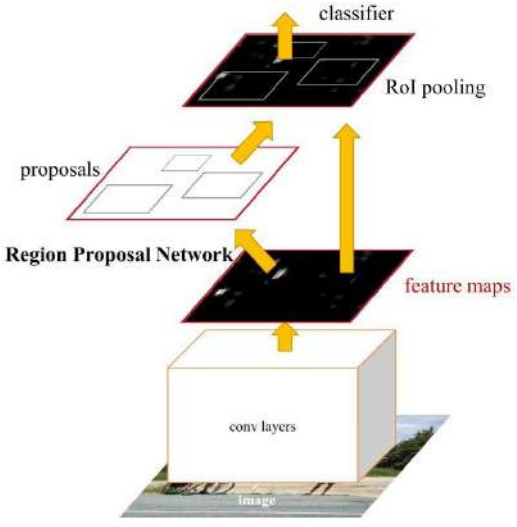


Figura 43: Esquema de Faster R-CNN [43]

Region Proposal Network es una red neuronal convolucional que genera las propuestas de regiones a varias escalas y relaciones de aspecto. Utiliza el concepto de atención en redes neuronales, que consiste en indicar al módulo de detección (en este caso Fast R-CNN) donde buscar objetos en la imagen.

RPN presenta una serie de ventajas sobre Selective Search, como pueden ser:

- Las propuestas de regiones son generadas con redes que pueden ser personalizadas y entrenadas especificamente para la tarea de detección.
- RPN procesa la imagen utilizando las mismas capas de la red que las usadas en detección, por lo que no se requiere tiempo adicional para este proceso a diferencia de Selective Search. Además, al utilizar una única red, solo hace falta entrenar la red una vez.
- Propone regiones mejores comparadas a Selective Search.

RPN trabaja utilizando el mapa de características obtenido de la última capa de la red compartida con Fast R-CNN. Se emplea una ventana rectangular de tamaño nxn que se desliza por el mapa de características, como se muestra en la figura 44, generando propuestas iniciales de regiones que serán posteriormente filtradas según la puntuación de clase, como se explicará a continuación.

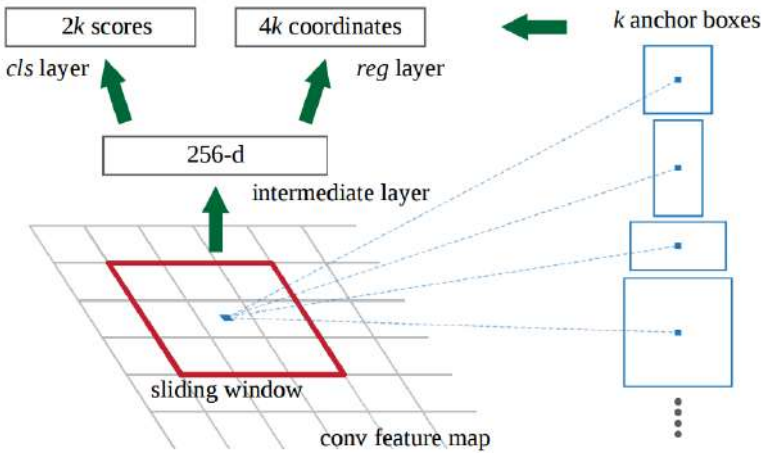


Figura 44: Ventana deslizante [42]

Para cada ventana antes mencionada se generan K propuestas de región. Cada región propuesta es parametrizada según escala y relación de aspecto de acuerdo a un cuadro de referencia que se denomina cuadro de anclaje. Normalmente se toman 3 escalas y 3 relaciones de aspecto, por lo que se producen K=9 propuestas de región con diferentes

escalas y relaciones. Gracias a estos cuadros de anclaje, es posible utilizar una única imagen a una cierta escala para detectar objetos a otras escalas sin utilizar filtros ni otras imágenes a dichas escalas.

Para cada región cuadrada propuesta se extrae un vector de características que se introduce en dos capas completamente conectadas. La primera capa es un clasificador binario llamado cls que genera una puntuación de clase, término que indica si una región contiene un objeto o pertenece al ambiente. La segunda genera los cuadros delimitadores, llamada reg.

La pregunta que surge en este momento es cómo se dan esas puntuaciones de clase, es decir, cómo puede RPN evaluar si una determinada región contiene un objeto o es parte del ambiente. Es por ello que RPN debe entrenarse para esta función, usando el concepto IoU, Intersection-over-Union.

IoU mide el solape de dos cuadros delimitadores. En este caso IoU es la relación entre el área de intersección entre el cuadro de anclaje y el cuadro verdadero, como puede ser el del etiquetado de los datos de entrenamiento, y el área de unión de dichos cuadros. De esta manera se obtiene como resultado un valor de 0 a 1, siendo 0 cuando no hay intersección entre ambos cuadros y 1 cuando coinciden. Esto se puede ver en la figura 45.

Figura 45: Intersection-over-Union. Fuente: Autor

Se asignará a la puntuación de clase un valor positivo o negativo en función del valor de IoU, siendo positivo cuando Iou sea mayor de 0.5, nulo entre 0.3 y 0.5, y negativo para IoU menor a 0.3.

A pesar de todos los avances mencionados, Faster R-CNN, al igual que sus predecesoras, tiene dificultades con la detección en tiempo real, en parte debido a lo lento que es en inferencia, es decir, al tratar con datos no entrenados con anterioridad. Otro problema que tiene está relacionado con el entrenamiento. Este se realiza en diferentes fases,

tanto para el RPN como para el clasificador, siendo además relativamente muy largo y pesado.

Para terminar este apartado, cabe mencionar que actualmente se están desarrollando modelos mejora dentro de esta familia para solventar estos problemas, como el Mask R-CNN, que devuelve máscaras delimitadoras a cada objeto detectado mostrando su silueta, o el R-FCN, que aumenta la velocidad compartiendo cálculos usando una red completamente convolucional.

3.3.1.2. SSD

SSD, siglas que significan Single Shot MultiBox Detector, es una arquitectura para detección de objetos diseñada en el año 2016 con el objetivo de su implementación en tiempo real [46]. Su arquitectura se muestra a continuación en la figura 46.

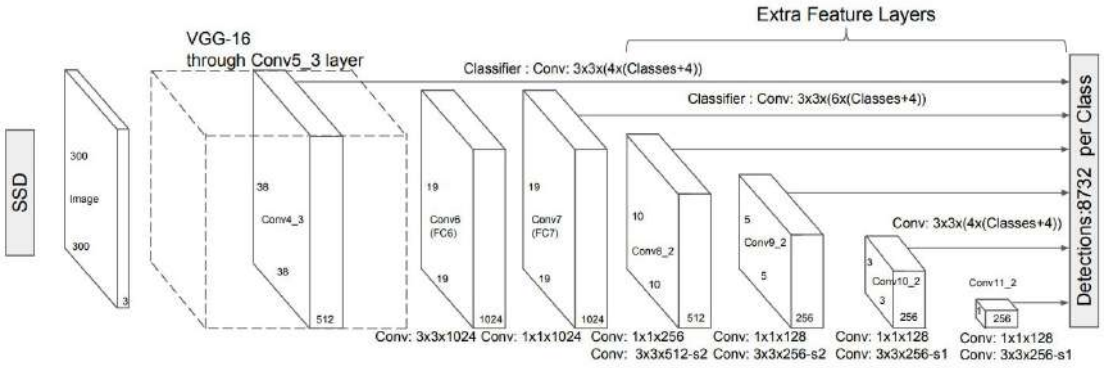


Figura 46: Esquema de SSD [47]

A diferencia de Faster R-CNN, que realiza las propuestas de región con la red RPN y su clasificación en capas completamente conectadas en etapas distintas, SSD realiza ambas operaciones simultaneamente, de ahí el nombre de 'single-shot', eliminando de la arquitectura la red RPN. Esto supondría una pérdida de precisión, pero para compensarlo SSD implementa un sistema de características de múltiples escalas y cuadros predeterminados, que permite mejorar la precisión respecto a Faster R-CNN usando imágenes de menor resolución.

SSD utiliza la red convolucional VGG-16 como base para extraer el mapa de características, descartando sus capas completamente conectadas. Esta es una red con un gran rendimiento en clasificación de imágenes de alta calidad, formada por una serie de filtros consistentes en capas convolucionales seguidas de operaciones de pooling. Se muestra en la figura 47.

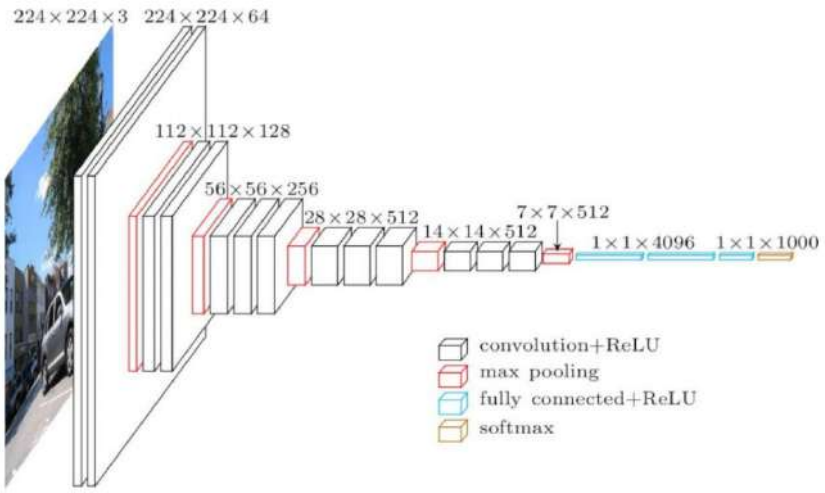


Figura 47: Red convolucional VGG-16 [46]

La última capa de esta red VGG-16 sin las capas complemente conectadas, llamada conv4_3, tiene un tamaño de 38x38 celdas. Para cada una de ellas se crean 4 predicciones, compuestas por un recuadro delimitador y puntuaciones para cada clase, incluyendo puntuación para una clase adicional, ambiente, también llamada clase '0'. Se calculan simultáneamente la localización y la puntuación de clase de cada objeto usando filtros convolucionales pequeños de 3x3 a cada celda.

Las capas completamente conectadas del VGG-16 son sustituidas por capas convolucionales auxiliares, permitiendo extraer características en múltiples escalas, disminuyendo progresivamente la entrada a la siguiente capa, y con ello disminuye a su vez la resolución del mapa de características.

La detección de objetos se realiza utilizando, además de la capa ya mencionada, la salida de varias de estas capas, por ello se dice que la detección utiliza mapas de características de múltiples escalas. SSD utiliza las capas con menor resolución para detectar los objetos a escala mayor, y las de mayor resolución para los de menor. Concretamente, la capa conv4_3 detecta objetos a escala 0.2 y la última capa la escala 0.9.

SSD añade 6 capas auxiliares a la base VGG-16, de las cuales 5 son utilizadas para la detección. En 3 de ellas se hacen 6 predicciones por celda en vez de 4, dando un total de 8732 predicciones sumando todas las capas.

La arquitectura SSD utiliza cuadros delimitadores predeterminados para detectar objetos de diferentes clases con distinta forma, utilizando un método equivalente a los cuadros de anclaje del Fast R-CNN. Los cuadros delimitadores predeterminados son escogidos manualmente.

Para calcular el ancho y alto de los cuadros predeterminados se tiene en cuenta la escala de la capa con las relaciones de aspecto del objeto. Por ejemplo, en las capas de 6 predicciones se empieza con relaciones 1, 2, 3, 1/2 y 1/3, y se calcula el ancho y alto del cuadro según lo siguiente:

$$ancho = escala \cdot \sqrt{relacion\ de\ aspecto} \qquad altura = \frac{escala}{\sqrt{relacion\ de\ aspecto}}. \tag{41}$$

Las predicciones serán entonces clasificadas según sean positivas o negativas, según se tenga un valor de IoU del cuadro predeterminado respecto al verdadero mayor o menor a 0.5. Solo las predicciones positivas tendrán repercusión en la función de coste de localización.

Debido al gran número de predicciones que se hacen por imagen (8732), la gran mayoría de estas acaban resultando ser negativas, creando un gran desbalance entre predicciones positivas y negativas. Esto genera un gran problema en el entrenamiento, ya que realmente lo que se está haciendo es entrenar la clase ambiente "0" en vez de lo que se pretende realmente detectar. Es por ello que se reduce la proporción hasta un máximo de 3 a 1, seleccionando únicamente los negativos con un cierto valor de confianza.

Se aumenta, como en muchas otras redes, el número de datos de entrenamiento para que el sistema sea más robusto. Se utilizan imágenes a distintas escalas y se voltean horizontalmente alatoriamente.

SSD utiliza un sistema para evitar predicciones duplicadas, utilizando non-maximum suppression (nms), sistema por el cual únicamente toma la predicción de cuadro con mayor IoU.

Para terminar con este apartado caben mencionar una serie de ventajas e inconvenientes de este detector.Como ventajas destacan las siguientes:

- Los mapas de características de múltiples escalas mejorar la detección a diferentes escalas.
- Tiene un sistema de localización de objetos mejorado respecto a Faster R-CNN, que reduce considerablemente el error de localización.

- SSD es bueno para sistemas embebidos.

Como desventajas, destacan las siguientes:

- SSD tiene un mal rendimiento para la detección de objetos pequeños, ya que la primera capa usada en la detección, la conv4_3 de 38x38, reduce mucho de la imagen respecto a la original, lo que puede provocar que los objetos no aparezcan en los mapas de características. Este problema puede solucionarse parcialmente siempre que se usen imágenes de resolución mayor.
- Si se quiere aumentar el número de cuadros delimitadores predeterminados para aumentar la precisión, se reduce la velocidad.
- SSD tiene menor error de localización que Faster R-CNN, pero mayor error de clasificación cuando trabaja con clases similares entre sí.

3.3.1.3. YOLO

YOLO, siglas de You Only Look Once, es una arquitectura para detección de objetos diseñada en el año 2016 que revolucionó este campo. YOLO presenta un nuevo enfoque para la detección, en el que los procesos de extracción de características y localización del objeto quedan unificadas [48]. Además, la localización y clasificación quedaban también unidas, por lo que el sistema se simplifica enormemente, quedando una arquitectura de una sola etapa, dando como resultado una velocidad en inferencia muy alta.

Existen diferentes versiones de YOLO, desde su aparición con YOLOv1 hasta versiones más modernas como YOLOv5, en las que progresivamente se han ido añadiendo mejoras. La versión utilizada en este proyecto ha sido YOLOv3.

YOLOv1

La primera versión de YOLO, YOLOv1, fue la primera arquitectura en aplicar el concepto de detector en una única etapa. Su estructura está básicamente conformada por 24 capas convolucionales junto con capas de pooling, y dos capas completamente conectadas al final, como se muestra en la figura 48.

Para la detección, cada imagen se divide en una cuadrícula dividida en casillas. Cada celda predice un determinado número de cuadros delimitadores y da valores de puntuación de clase. Si un objeto se encuentra en alguna de las casillas de la cuadrícula, estas casillas son responsables de su detección.

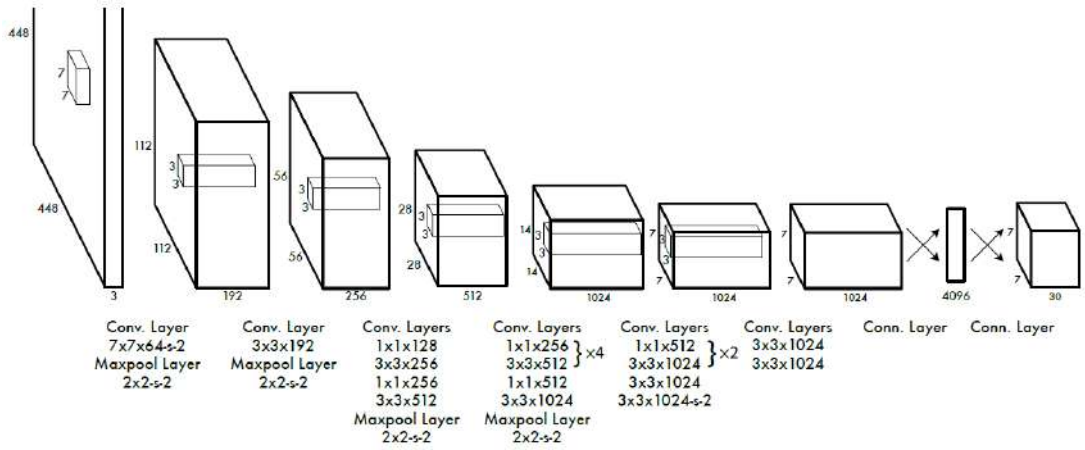


Figura 48: Esquema de YOLOv1 [49]

YOLO se entrena en dos fases: en la primera se entrena la clasificación de la red con imágenes 224x224, y en la segunda se entrena la detección quitando las capas completamente conectadas y sustituyéndolas con una capa convolucional, reentrenando la red ahora con imágenes de 448x448.

Otra innovación de YOLOv1 fue el uso de una nueva función de activación llamada *leaky ReLU*, variante de la ya mencionada ReLU, representada en la figura 49. Esta soluciona problemas relacionados con la inactividad de algunas neuronas de una red a cualquier entrada.

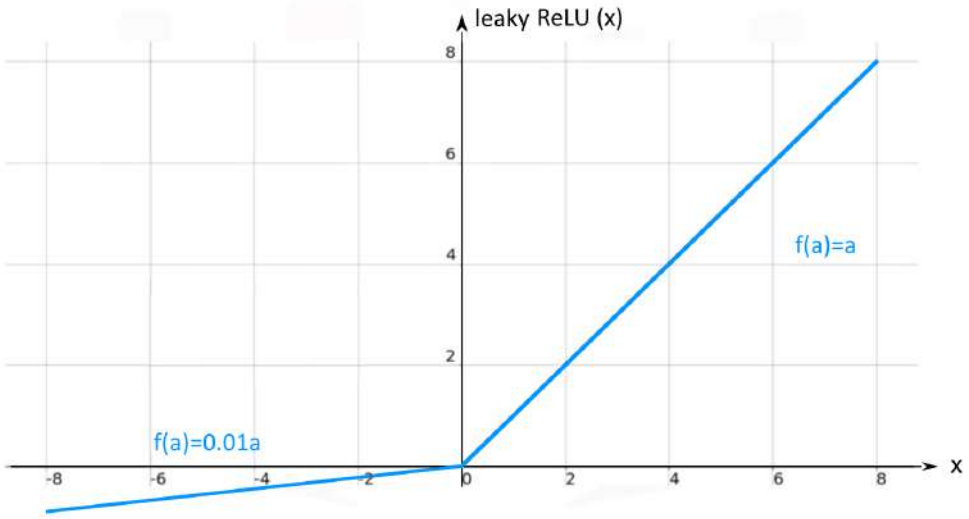


Figura 49: Leaky ReLU. Fuente: Autor

Algunas de las ventajas que ofrece YOLOv1 respecto a otras redes son su rapidez, arquitectura en una única etapa, mayor generalización de los datos y menores falsos positivos en áreas del ambiente.

YOLOv2

YOLOv2, también conocido como YOLO 9000, es una versión mejorada de YOLOv1. Utiliza una arquitectura llamada *darknet-19*, consistente en 19 capas convolucionales junto a otras 11 para la detección de los objetos, dando un total de 30 capas, mostrada en la figura 50.

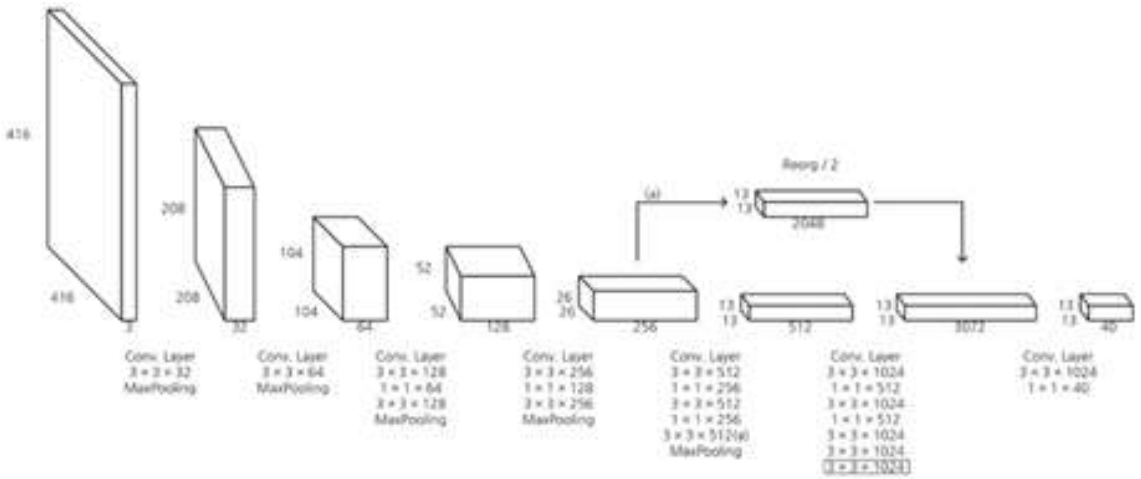


Figura 50: Esquema de YOLOv2 [49]

Algunas de las mejoras se describen a continuación.

- YOLOv2 utiliza un concepto llamado normalización de los lotes, una técnica para el entrenamiento de la red que consiste en normalizar las entradas a esta, aplicado a las activaciones de las neuronas de las capas anteriores o a la entrada misma. Normalizar los lotes permite a la red usar tasas de aprendizaje mayores, requiriendo menos épocas de aprendizaje y por tanto reduciendo el tiempo empleado en entrenamiento. Una época de aprendizaje es cuando se han utilizado todas las imágenes del dataset de entrenamiento durante el entrenamiento. Para entrenar una red se requieren normalmente varias épocas, como se verá más adelante.
- En el entrenamiento, YOLOv2 utiliza, además de las imágenes 224x224, imágenes de 448x448 para entrenar la clasificación, mejorando la resolución en clasificación.
- YOLOv2 elimina de su estructura las capas completamente conectadas y pasa a emplear cuadros de anclaje similares a los de Faster R-CNN.
- Hay que destacar que hay mejoras en las predicciones de localización.

Una de las ventajas de YOLOv2 respecto a YOLOv1 es que la arquitectura tiene mayor independencia a la resolución de las imágenes, operando mejor con imágenes a más baja resolución.

Un problema que tiene YOLOv2 es detectar objetos pequeños, debido a la pérdida de detalles a medida que las capas reducen la imagen de entrada.

YOLOv3

YOLOv3 mejora en muchos aspectos a sus predecesoras. Utiliza una arquitectura llamada *darknet-53*, consistente en 53 capas más otras 53 para la detección, dando un total de 106 capas. Esto supone una disminución en la velocidad comparado con YOLOv2, que tan solo cuenta con 30 capas, pero gana en precisión. La figura 51 muestra la estructura de YOLOv3. siguiente.

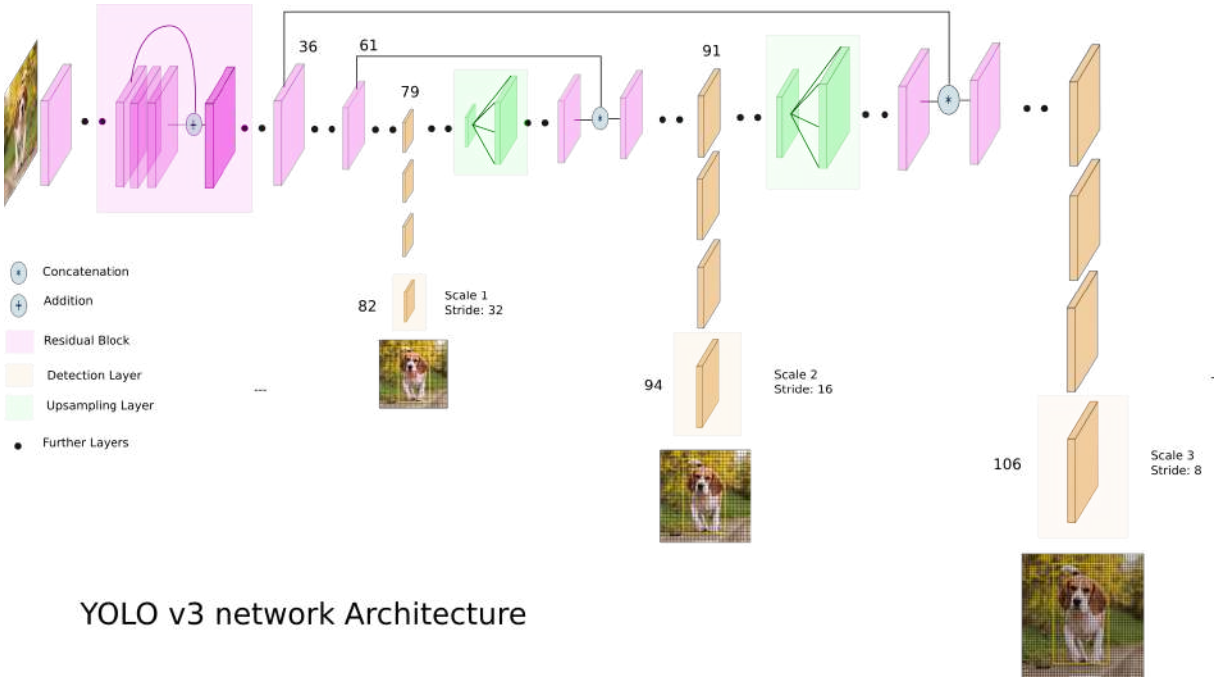


Figura 51: Esquema de YOLOv3 [48]

La principal mejora de YOLOv3 es que hace detecciones a 3 escalas. Esto se consigue mediante la reducción de las dimensiones de la imagen en 32, 16 y 8. La reducción consiste en bajar la resolución de las imágenes para reducir el número de datos. La primera detección se hace en la capa 82. Para las primeras 81 capas la imagen se reduce hasta que en la capa 81 queda con un paso de 32. Por ejemplo, si se introduce una red de tamaño 416x416, el mapa de características queda de tamaño 13x13.

Cada escala utiliza 3 cuadros de anclaje por capa, dando un total de 9 cuadros de anclaje. Así la red es capaz de detectar objetos de distintos tamaños, solventando parcialmente el problema que tenía YOLOv2 para detectar objetos pequeños.

Otra mejora respecto a YOLOv2 es el número de cuadros delimitadores que predice. YOLOv3 predice 10 veces dicho número. Esto resulta en una pérdida de velocidad pero gana precisión.

Otro cambio que incorpora YOLOv3 es una diferente función de coste. YOLOv2, como la mayoría de redes, emplea una función de coste cuadrática. YOLOv3 pasa ahora a calcular el coste usando regresión logística.

Otra mejora es el uso de regresión estadística para dar las puntuaciones de clase. La función softmax (figura 40) empleada en la mayoría de redes hasta el momento tiene un carácter exclusivo. Esto quiere decir que se hace la suposición de que las clases se excluyen mutuamente, es decir, que si un objeto pertenece a una clase ya no puede pertenecer a otra. Por ejemplo, no funcionaría correctamente la clasificación con clases Animal y Perro, ya que la clase Perro está incluida en la de Animal. La regresión logística junto al empleo de umbrales soluciona este problema.

Para terminar el apartado de detección de objetos se detalla a continuación, en la tabla 3, un resumen que recoge las principales ventajas e inconvenientes de las redes Faster R-CNN, SSD y YOLOv3:

Tabla 3: Resumen de las ventajas e inconvenientes de Faster R-CNN, SSD y YOLOv3

	Ventajas	Inconvenientes
Faster R-CNN	Gran precisión.	Velocidad en inferencia insuficiente para la mayoría de aplicaciones en tiempo real, a pesar de las mejoras de velocidad introducidas con el RPN.
SSD	Buena velocidad en inferencia. Error de localización bajo.	Mal rendimiento para la detección de objetos pequeños. Error de clasificación alto para clases similares.
YOLOv3	Alta velocidad en inferencia. Localización de objetos muy eficiente. Estructura en una única etapa. Buena precisión.	Problemas leves para la detección de objetos pequeños.

4. HERRAMIENTAS DE DESARROLLO

4.1. Cámara infrarroja

Para la elaboración de este trabajo se ha contado con una cámara infrarroja Optris PI 640 de la compañía OPTRIS.

Esta cámara es de las más compactas del mercado, teniendo un tamaño de 46 x 56 x 76 - 100 mm dependiendo de la lente equipada. Su peso oscila entre los 269 y 340 g, también dependiendo de la lente. La figura 52 muestra sus dimensiones.

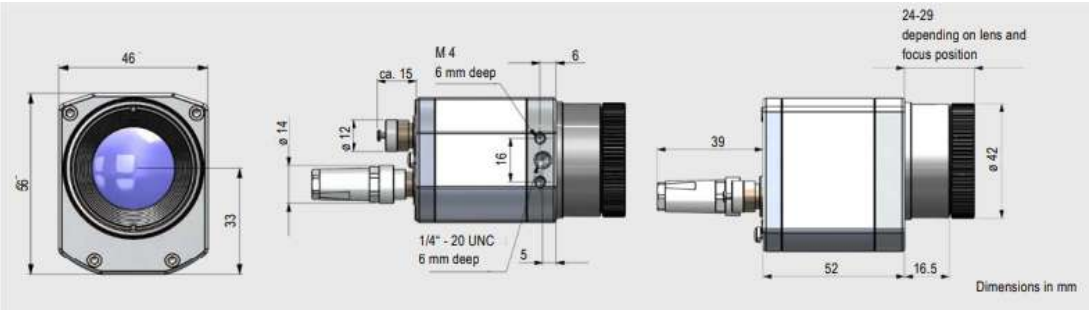


Figura 52: Dimensiones Optris PI 640 [1]

En cuanto a características ópticas, la cámara posee una resolución de 640 x 480 pixeles, con una frecuencia de refresco de imagen de 32 Hz. En cuanto a características térmicas, la cámara opera en el rango espectral de 8 a 14 µm, pudiendo detectar temperaturas dentro el rango de los -20 a los 900 °C. Tiene una sensibilidad térmica de 75mK y una precisión de ±2°C o ±2 %, según sea el mayor.

La cámara se conecta a otros dispositivos a través de USB, por donde además se alimenta.

Esta cámara utiliza el software de análisis de infrarrojos Optris PIX Connect, desarrollado por la misma compañía, que se muestra en la figura 53. Esta herramienta permite registrar y analizar imágenes termográficas a tiempo real, además de implementar herramientas de automatización y control de procesos.

Al conectar la cámara al ordenador con este software activo, se muestra por pantalla lo que la cámara está captando a tiempo real. Junto a la imagen aparecen múltiples herramientas y datos de apoyo como perfiles de temperatura vertical y horizontal, escala de temperaturas, histogramas de temperatura por regiones, etc.

Si se acerca el cursor a la imagen, se muestra junto a este el valor de temperatura del píxel al que se apunta. Otra herramienta útil es la creación de áreas de trabajo, donde se

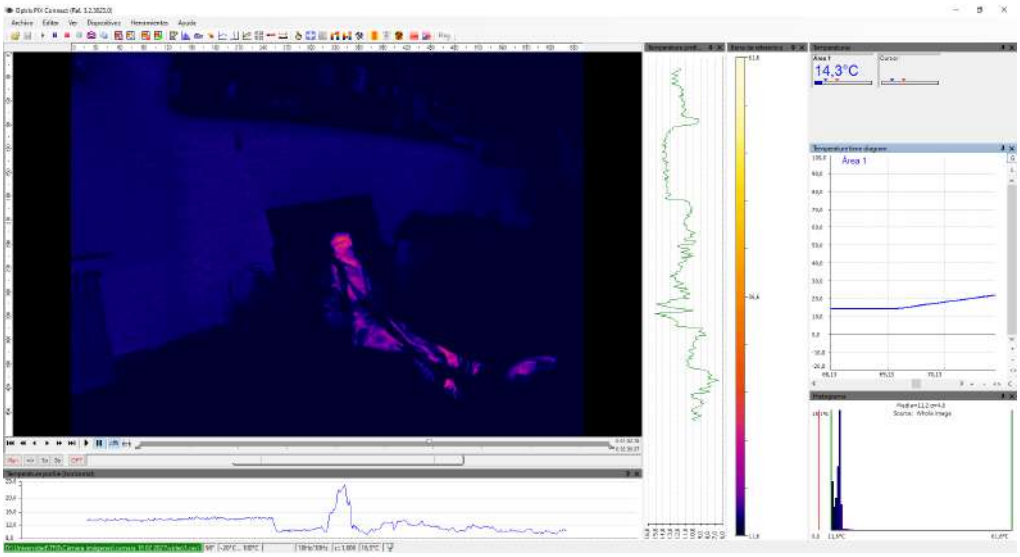


Figura 53: Software Optris PIX Connect. Fuente: Autor

selecciona un área con una forma predeterminada o totalmente personalizada, donde luego se puede obtener datos como la temperatura media.

Otra opción es configurar la paleta deseada. Como se ha visto en el apartado de Conceptos Teóricos, existen múltiples formas de representar imágenes térmicas, y el software permite escoger entre una gran cantidad de ellas. En cuanto a la escala de temperatura, esta puede ser ajustada manualmente o ajustada automáticamente según la estadística de las temperaturas de la imagen.

Para la grabación y el guardado de datos, el software emplea formatos especiales. Para el caso de los videos, estos se graban en un formato '.ravi', y para las imágenes en uno llamado '.tif, que almacenan la información de la temperatura de cada pixel de la imagen y las diferentes configuraciones establecidas en el análisis.

También otro formato de guardado de datos de temperatura es en '.csv', para poder trabajar con los valores de temperatura en hojas de Excel.

Se pueden hacer capturas de pantalla dentro del software en formato '.png' o '.jpeg', pero en ellas sale, a parte de la imagen, el resto de herramientas empleadas en pantalla.

4.2. Entorno de programación

El lenguaje de programación empleado para este proyecto es Python. Phyton es un lenguaje interpretado, multiparadigma y multiplataforma cuya filosofía reside en la legibilidad del código. Tiene un gran respaldo por su comunidad, que permite el fácil

uso de multitud de librerías y aplicaciones desarrolladas por otros usuarios. La figura 54 muestra su logo.

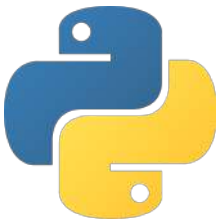


Figura 54: Logo de Python [50]

Tiene un gestor de paquetes llamado 'pip' por el cual cualquier usuario puede instalar librerías de la comunidad en su sistema. La página web pypi.org actúa como repositorio de estos paquetes, ofreciendo un sistema de búsqueda de paquetes, donde se puede encontrar la descripción de los mismos junto a los comandos de instalación.

Python se aplica a multitud de sectores, como desarrollo web, Data Science, automatización de sistemas o Machine Learning.

Para la instalación de Python se ha requerido lo que se conoce como entorno virtual, realizado con Anaconda. Anaconda es una plataforma de distribución de Python para aplicaciones de Machine Learning y Data Science. La figura 55 muestra su logo.

Figura 55: Logo de Anaconda [51]

Un entorno virtual es una copia de Python aislada que mantiene sus propios ficheros, directorios y direcciones de manera que el usuario pueda trabajar con versiones específicas de librerías o de Python, sin afectar a otros proyectos en Python [52] .

La tarea de detección usando redes neuronales requiere múltiples librerías de distintas versiones, por lo que si no se empleara un entorno virtual y se utilizara Python para otro proyecto que emplee alguna de estas librerías pero en otras versiones, surgirían conflictos de compatibilidades. Mantener la instalación de Python y sus paquetes aislada para cada proyecto es de suma importancia, ahorrando incontables problemas.

Al instalar Anaconda se instala también la interfaz gráfica del usuario Anaconda Navigator, donde se pueden lanzar aplicaciones y manejar paquetes y entornos virtuales. El

gestor de paquetes de anaconda se llama 'conda', diseñado para trabajar con dependencias de paquetes, aunque es posible seguir usando 'pip' en los entornos virtuales.

Algunos de los principales paquetes requeridos para este proyecto son:

- Tensorflow 2.3: TensorFlow es una plataforma de código abierto para el aprendizaje automático desarrollado por Google que permite crear y entrenar redes neuronales [53]. Su nombre se debe a que la librería trabaja con tensores, vectores multidimensionales. La figura 56 muestra su logo.

Figura 56: Logo de Tensorflow [53]

- Tensorboard: Tensorboard es la librería de herramientas de visualización de Tensorflow. Esta librería permite visualizar las medidas de la red durante el entrenamiento y evaluación de la red, algo prácticamente imprescindible para el aprendizaje automático. Otras librerías como Pytorch también tienen acceso a estas herramientas con Tensorboard.
- Pytorch: Pytorch es una librería para Machine Learning desarrollada por Facebook. Al igual que Tensorflow, Pytorch trabaja con tensores de grandes dimesiones. La figura 57 muestra su logo.

Figura 57: Logo de Pytorch [54]

- OpenCV python: OpenCV es una librería de Visión Artificial y Machine Learning. Su propósito es servir como infraestructura para las aplicaciones de visión Procesa videos e imágenes. Actualemente OpenCV se utiliza en multitud de aplicaciones, destacando la de procesamiento de imágenes en tiempo real. La figura 58 muestra su logo.

Figura 58: Logo de OpenCV [55]

- Protobuf: Protocol Buffers es una librería desarrollado por Google utilizada para serializar datos, es decir, traducir estructuras de datos a un formato que pueda ser utilizado para transmitir, reconstruir y almacenar información. En este proyecto se usará para generar ficheros '.record' que almacenarán información sobre todas las imágenes utilizadas en el entrenamiento y evaluación de la red como se verá más adelante.

Otros paquetes importantes que se han requerido para este proyecto han sido Numpy (librería que permite trabajar con vectores y matrices de grandes dimensiones), Matplotlib (librería que representa gráficas a partir de datos de arrays de Python y Numpy) y Pandas (librería que incluye herramientas para el análisis de datos).

Las tareas relacionadas con la visión artificial suelen requerir una gran capacidad computacional, por manejar mucha información de manera independiente simultaneamente. Es por ello que el entrenamiento y evaluación de una red es muy lento o diréctamente no se cuenta suficientes recursos para llevarlos a cabo en sistemas poco potentes. Las CPUs de los ordenadores entán formados por uno o más núcleos dispuestos de manera secuancial, por lo que puede saturarse con mucha facilidad al tratar de procesar muchas acciones paralelas.

Una alternativa a emplear las CPUs es emplear GPUs, procesadores gráficos especializados en trabajar con gran cantidad de información en paralelo. Para poder utilizar la GPU de los ordenadores para Visión Artificial hace falta instalar CUDA y CuDNN.

CUDA, siglas en inglés de Arquitectura Unificada de Dispositivos de Cómputo, es una plataforma de computación en paralelo desarrollado por la compañía nVidia que permite emplear tarjetas gráficas para acelerar porcesos. Aprovecha el paralelismo y el alto ancho de banda de las GPUs en operaciones con mucho coste aritmético [56]. La librería que permite utilizar CUDA para aplicaciones de Deep Learning se llama CuDNN.

Tanto Tensorflow como Pytorch cuentan con versiones que trabajan con CPU o GPU. Es conveniente, si se cuenta con una tarjeta gráfica lo suficientemente potente, utilizar la versión GPU de las librerías.

5. METODOLOGÍA

5.1. Configuración del entorno de trabajo

La configuración del entorno de trabajo es el proceso en el que se procede a la creación de los entornos virtuales y a la instalación de los módulos requeridos para realizar las tareas deseadas. La configuración de los dos entornos virtuales empleados en este proyecto, aunque parezca que sea un proceso rápido y sencillo, es uno de los procesos que más errores provocan y que más tiempo requiere para enmendarlos.

Para la tarea de detección, como se ha podido ver en el apartado de herramientas de desarrollo, hace falta instalar múltiples módulos de Python en los entornos virtuales de anaconda. En algunos procesos se requiere la instalación de versiones específicas de alguno de estos módulos, y en múltiples ocasiones estos módulos tienen dependencias entre ellos, pudiendo llegar a haber alguna incompatibilidad.

Esto puede suponer un gran problema si una determinada versión de un módulo del que dependen otros ya no tiene soporte, y con ello solo se pueden instalar versiones alternativas, generando un conflicto de compatibilidades en cadena.

El gestor de paquetes 'conda' de Anaconda está específicamente diseñado para resolver este tipo de problemas de dependencias, aunque no siempre es capaz de solucionarlos. Se pueden revisar los paquetes instalados y sus versiones en el entorno virtual utilizan el comando *conda list*.

Para este proyecto se van a emplear dos entornos virtuales: uno que utilizará Tensoflow como base para entrenar redes Faster R-CNN y otro que utilizará Pytorch para YOLOv3. Para crearlos simplemente hay que escribir en la terminal de Anaconda *conda create ENTORNO* seguido de la versión de Python que se quiera descargar. Para activar este entorno solo hay que escribir *conda activate ENTORNO*.

Ambas librerías Tensorflow y Pytorch, como se ha mencionado en el apartado anterior, deben de trabajar preferiblemente con la GPU, y para ello se instalaron tanto CUDA como CuDNN. Para saber que versión hay que descargar de estas dos últimas hay que revisar los datos de la GPU del ordenador y prodecer a instalar la adecuada.

Para comprobar la correcta instalación de CUDA, dentro de los entornos una vez instalados el resto de módulos se pueden ejecutar en Python dos sencillas líneas de código:

- Para el entorno de Tensorflow
  import tensorflow as tf
  tf.test.is_gpu_available()

- Para el entorno de Pytorch
  import torch
  torch.cuda.is _availabe()

Para las redes que utilizan Tensorflow, se ha utilizado la interfaz de programación Tensorflow Object Detection API. Es un entorno de trabajo creado por Google que facilita la creación, entrenamiento e implementación de los modelos de detección. Consiste en un repositorio que contiene todas las carpetas y ficheros de programación necesarios para realizar todos los pasos para conseguir un detector. Únicamente habría que seguir una serie de pasos, como añadir la red pre-entrenada a la que se desea hacer el fine tuning, añadir el dataset con el que se quiere trabajar y configurar determinados ficheros para adaptarlos a lo necesitado.

Hay una lista de redes llamada Tensorflow 2 Detection Model Zoo [57], que consiste en una serie de redes que han sido entrenadas previamente con un dataset llamado COCO (Common Objects in Context). Aquí se pueden encontrar los modelos de multitud de redes en los que se muestran datos como velocidad, precisión y tipo de salida que producen. Una vez elegida la red pre-entrenada para que funcione de base para el detector, simplemente habría que descargarla. En este proyecto se han elegido las redes Faster R-CNN ResNet152 V1 640x640 y SSD ResNet50 V1 FPN 640x640 (RetinaNet50).

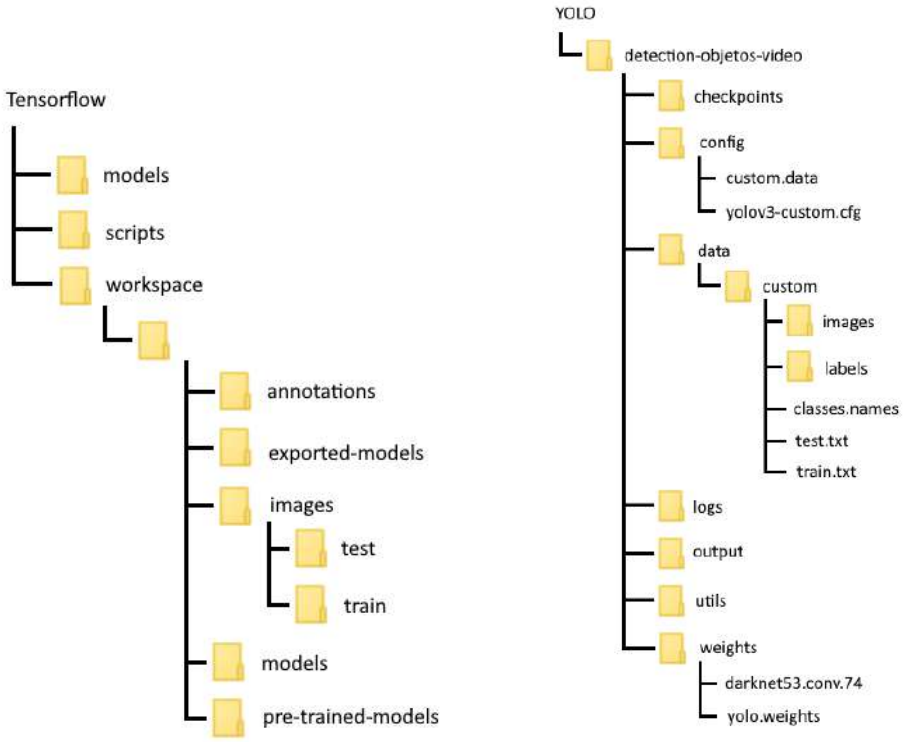
Para YOLO sucede de manera similar, hay multitud de repositorios que contienen todo lo necesario para implementarla. La versión de la red empleada en este proyecto es YOLOv3. Se utilizarán los pesos pre-entrenados *darknet53.conv.74*.

Se ha seguido una cierta organización en los archivos, siguiendo ciertas guías como las encontradas en páginas como [58] y [59].

Para el caso de Tensorflow, se ha creado un directorio donde dentro se encuentran 3 carpetas: models (donde se encuentran los ficheros y carpetas de la API), scripts (donde hay algunos ficheros de preprocesamiento) y workspace (en donde se trabajará la mayor parte del tiempo, donde se encuentran la mayoría de ficheros a modificar para

adaptar la red a la aplicación que se desee). La figura 59 a muestra la organización de los directorios.

Para YOLO se ha creado un directorio donde se encuentran las carpetas de checkpoints (donde se harán copias de la red en distintos puntos del entrenamiento), config (donde se encuentran ficheros de configuración), data (donde se encuentran las imágenes y sus etiquetas), weights (donde se encuentran los pesos de la red y de la red pre-entrenada) y demás ficheros y directorios. La figura 59 b muestra la organización de los directorios.



(a) Tensorflow (b) YOLO

Figura 59: Diagrama de árbol de los directorios. Fuente:Autor

5.2. Generacion de Datasets

Para entrenar redes neuronales que realicen una tarea en concreto se requiere una gran cantidad de datos de entrenamiento. Para la tarea de detección, en este caso de víctimas de accidentes, este conjunto de datos que se provee a la red, como se ha mencionado anteriormente, es en forma de imágenes. Es necesario para crear un modelo eficaz y flexible incorporar imágenes diversas, en diferentes condiciones, que aporten información a la red y logren que, una vez lista para el funcionamiento, pueda operar correctamente en distintos escenarios.

Además, son necesarias imágenes adicionales para la evaluación de dichas redes. Gracias a estas imágenes se verifica la eficacia y el buen funcionamiento de la red una vez se haya terminado el entrenamiento.

La generación del dataset empleado para entrenar y evaluar la red, se realizó siguiendo una serie de pasos.

Primero, se utilizó la cámara térmica previamente mencionada en el apartado de herramientas de desarrollo, Optris PI 640, conectándola a un ordenador portatil y ejecutando el software de grabación y análisis Optris Pix Connect.

Dentro de este software, se procedió a grabar videos cortos en los que se mostraban personas simulando ser víctimas en diferentes escenarios.

Es de suma importancia para obtener modelos robustos hacer grabaciones que muestren una gran variedad de situaciones y facetas. Es por ello que se han grabado vídeos en los que aparecían víctimas simuladas en diferentes posiciones, a diferentes horas del día y en los que no se mostraba el cuerpo completo de la víctima, sino partes del cuerpo concretas como cabezas, piernas, brazos, etc. La mayoría de las grabaciones fueron realizadas en el exterior. Se emplearon además materiales típicos de diferentes ambientes, como puede ser materiales de obra o de ambiente urbano, como placas de metal y madera o películas plásticas y de tela tanto para simular el ambiente como para tapar los cuerpos parcial o totalmente, como se muestra en la figura 60.

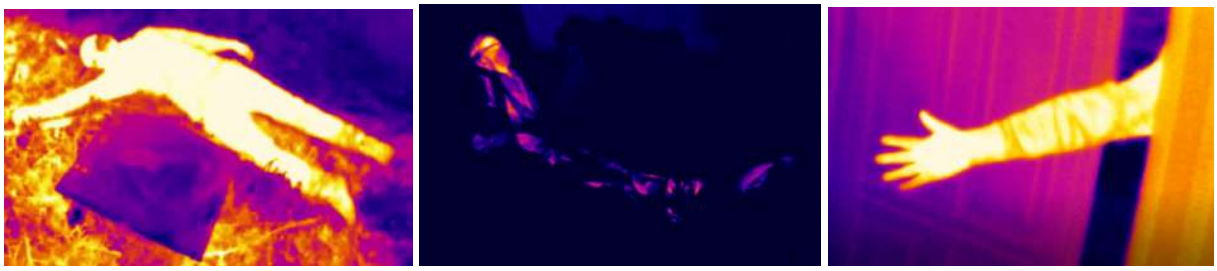


Figura 60: Ejemplos de imágenes variadas recogidas para los datasets. Fuente:Autor

Para el guardado de videos e imágenes, el software emplea los formatos especiales '.ravi'y '.tiff' respectivamente. Estos formatos, al guardar información de la temperatura de cada pixel de la imagen y la configuración de esta, llegan a generar archivos mucho más pesados comparado a otros formatos de video e imagen. De ahí la razón de grabar videos cortos, de lo contrario estos ocuparían un espacio excesivo, lo que dificultaría su procesamiento y almacenamiento.

La grabación en estos formatos supone otro problema, ya que para trabajar con las librerías convencionales de visión es necesario emplear imágenes en formato de imagen convencional, como puede ser '.jpg'. Es por ello que se empleó una grabadora de pantalla, que capturó los videos a pantalla completa en el software Optris Pix Connect y los guardó a formato '.mp4'.

Posteriormente, de estos videos se obtuvieron sus fotogramas utilizando la ayuda de un paquete de python llamado ffmpeg, que se configuró para obtener 5 fotogramas por cada segundo de video. Así se obtuvieron las imágenes en formato '.jpg' necesarias para el entrenamiento y evaluación de la red. Se muestra el procedimiento esquematizado en la figura 61.

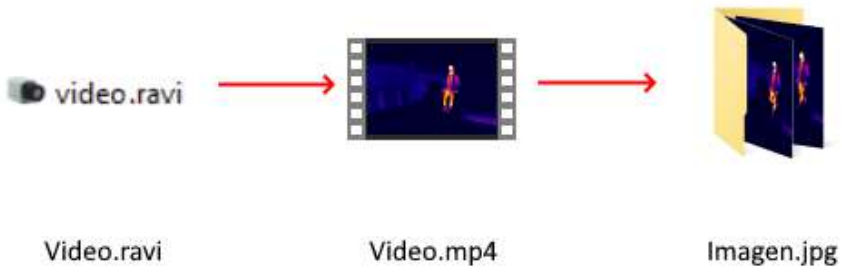


Figura 61: Generación de datasets. Fuente: Autor

5.2.1. Etiquetado

Una vez obtenidas todas las imágenes que se quieren utilizar para entrenar y verificar la red, hay que seleccionarlas y etiquetarlas.

Se ha utilizado una herramienta de anotación llamada LabelImg, que funciona en Python, para etiquetar las imágenes. Este programa permite recuadrar los elementos que se quieren detectar en las imágenes, permitiendo guardar la posición y clase de cada recuadro dentro de un fichero asociado a estas imágenes.

Para utilizar el software primero hay que editar un fichero llamado labels-map, y escribir en este el nombre de las clases que se quieran detectar. Una vez hecho esto simplemente habría que abrir la consola de Anaconda, activar un entorno virtual de python y dirigirse a la carpeta donde se tiene instado LabelImg. Allí simplemente habría que introducir *python LabelImg.py* y el software se ejecutará. Aparecerá un ventana con diferentes opciones. Para etiquetar las imágenes simplemente habría que abrir el directorio donde se localizan y seleccionar un directorio donde se guardarán los ficheros de etiquetado a dichas imágenes.

Para etiquetar la imagen simplemente hay que seleccionar la opción *Create RectBox* y recuadrar el área de la imagen que contienen lo que se pretende etiquetar. Una vez hecho esto, aparecerá una ventana en la que se pide identificar la clase recuadrada. Se repite esto hasta que se tengan todas las clases con sus recuadros etiquetados en la imagen. Para terminar solo hay que elegir el formato de guardado y dar al cuadro de *Next Image* para proceder a etiquetar la siguiente imagen. Un ejemplo de etiquetado se representa en la figura 62.

Figura 62: Ejemplo de etiquetado en LabelImg. Fuente:Autor

Etiquetar imágenes es una tarea larga y tediosa. Para agilizar el proceso hay una serie de atajos en este software, como por ejemplo *w* para crear un recuadro, *ctrl+s* para guardar el etiquetado y *d* para pasar a la siguiente imagen.

Además, es importante destacar que, al generar imágenes al utilizar los frames de videos, se tienen imágenes repetidas o que varían muy poco de unas a otras. Es por ello que, aparte de etiquetar, hay que seleccionar las imágenes que se van a etiquetar. Por poner un ejemplo de la cantidad de imágenes que se descartan, de las **3331** imágenes obtenidas en este proyecto, se han seleccionado y etiquetado un total de **872** imágenes. Estas imágenes han sido repartidas en diferentes datasets, de acuerdo a diferentes pruebas que se describirán más adelante en la sección de Pruebas.

Hay diferentes formatos para el guardado de las coordenadas y clases etiquetadas según la red que se va a entrenar. El formato Pascal VOC se utiliza para las redes que utilizan Tensorflow y el etiquetado se guarda en '.xml'. El formato YOLO se utiliza, como su nombre indica, para la red YOLOv3. En la figura 63 se muestrane ejemplos de ambos formatos.

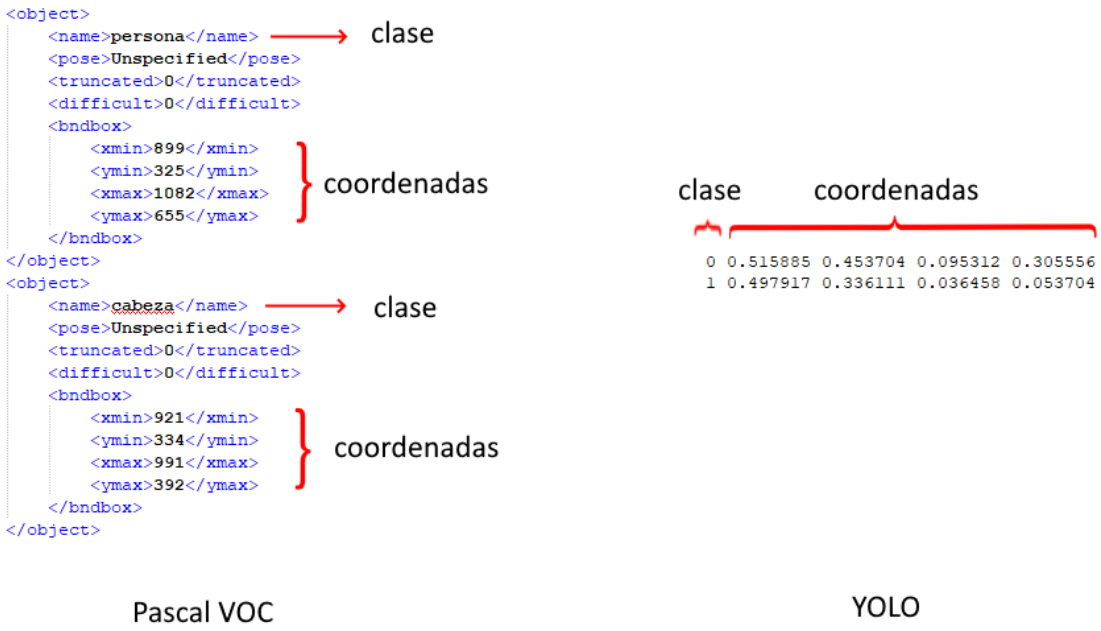


Figura 63: Ejemplo de formato Pascal VOC (izquierda) y YOLO (derecha). Fuente:Autor

Estas imágenes y sus etiquetas se moverán a las carpetas correspondientes de los directorios de trabajo, según el dataset que se vaya a emplear en ese momento. Para la API de Tensorflow hay que separar las imágenes de cada dataset junto a sus etiquetas en carpetas de entrenamiento y validación. La proporción debe de ser de alrededor de 80 % de imágenes para entrenamiento y 20 % para validación.

Para el caso de YOLO hay que meter todas las imágenes en una carpeta y todas las etiquetas en otra. Posteriormente hay que ejecutar un fichero de Python llamado *split_train_val.py* que se separa las imágenes para entrenamiento de las de validación mediante la generación de dos ficheros, train.txt y valid.txt, que contienen la dirección de las etiquetas que se emplearán para entrenamiento y validación, respectivamente.

Hay que mencionar que, para el caso de las redes de Tensorflow, hay que generar unos archivos llamados *train.record* y *test.record* cada vez que cambie de dataset. Estos archivos contienen la información de tanto las imágenes como de su etiquetado guardado en un formato especial, y es esto lo que se emplea para entrenar y validar las redes. Se generan gracias a la librería protobuf.

Si estos ficheros se generan de manera incorrecta o se emplean ficheros generados con anterioridad para otras aplicaciones, se pueden llegar a entrenar o evaluar redes con información de imágenes indeseadas. Este es un error que se tuvo durante la primera evaluación de una red, donde se obtuvieron errores en las mediciones de

los parámetros de evaluación de la red. Se debían a que no se generó correctamente el fichero test.record para el dataset de detección de víctimas, por lo que se estaba utilizando el fichero test.record de la aplicación original del repositorio del que se obtuvo las carpetas de la API.

Finalmente cabe mencionar que, debido al gran volumen de ficheros de imágenes y etiquetas que se han manejado en este proyecto, se han creado ficheros de Python que permitían copiarlos o moverlos automáticamente de un directorio a otro, resultando de gran utilidad al cambiar de datasets.

5.3. Entrenamiento

Una vez configurado los entornos virtuales, etiquetado las imágenes y generado los ficheros correspondientes, llega la etapa fundamental de la detección: el entrenamiento de las redes.

Para el caso de la API de Tensorflow, una vez se haya descargado y guardado en la carpeta *pre-trained-models* el modelo pre-entrenado al que se hará el fine-tuning, se crea en el directorio *models* una carpeta que contendrá la configuración y todos los puntos de guardado de la red que se pretende entrenar. Para ello, se copia el fichero *pipeline.config* de la red pre-entrenada a la carpeta recién creada. Este fichero contiene información básica de la red y debe configurarse para adaptarla a la nueva funcionalidad.

Al abrir el fichero *pipeline.config* de la red Faster R-CNN se puede ver distintos parámetros que configuran la red, como pueden ser el reescalado previo que se le hace a las imágenes a la entrada de la red (640x640), función de activación (ReLU), los diferentes relaciones de aspecto y escalas con las que trabaja ([0.5, 1, 2] y [0.25, 0.5, 1, 2], respectivamente), el paso de los filtros que se aplican (16 de altura y 16 de largo), la función de puntuación de clase (softmax), el valor de la tasa de aprendizaje (0.04 inicial con decaimiento de tipo coseno) y opciones de aumento de datos (volteo de imagen horizontal aleatorio), entre otros.

Para el fichero *pipeline.config* de la red SSD se pueden ver parámetros similares, con algunos valores diferentes como puede ser en las relaciones de aspecto (0.5, 1, 2, 4) o las opciones de aumento de datos (volteo horizontal aleatorio y recorte aleatorio de imágenes a diferentes relaciones de aspecto).

En estos ficheros hay parámetros que deben de adaptarse. Estos son el número de clases, el tipo de red (en este caso es una red de detección) y la dirección de los ficheros

train.record, *test.record*, *label_map.pbtxt* y el fichero de guardado de la red pre-entrenada *ckpt-0*. *label_map.pbtxt* es un fichero en la carpeta *annotations* que hay que configurar simplemente escribiendo los nombres de las clases utilizadas.

Otro parámetro que hay que modificar es el tamaño de los lotes (batches) de imágenes que se usarán durante el entrenamiento. A mayor parámetro batch, más rápido se entrenará la red. Este parámetro depende fundamentalmente de la memoria de la GPU. Si se escoge un valor alto de batch, se tomarán más imágenes a la vez para el entrenamiento, lo que puede llegar a saturar la GPU indicando por pantalla un mensaje tipo *Allocator (GPU_0_bfc) ran out of memory trying to allocate...*. Esto bloqueará el entrenamiento, por lo que debe lograrse encontrar el mayor parámetro de batch posible que pueda procesar la GPU.

Para el caso de Faster R-CNN el valor máximo de batch permitido fue 2 y el de SSD 4.

Una vez hecho esto, puede comenzar el entrenamiento. Para iniciarlo simplemente hay que ejecutar el fichero model_main.py del repositorio, especificando el directorio donde se guardarán los puntos de guardado del modelo y la dirección del ficher*opipeline.config*.

Se tardará un tiempo en abrir las librerías de CUDA y CuDNN para emplear la GPU, y cargar los parámetros de la red pre-entrenada. Una vez hecho esto, comenzará a entrenarse la red.

Irán apareciendo por pantalla mensajes que indican los pasos (steps) de entrenamiento junto con el tiempo que emplea el ordenador en llevar a cabo dichos steps y el valor actual de de la función de coste, también llamada pérdida total y loss en inglés:

INFO:tensorflow:Step 100 per-step time 1.207 loss=1.454

El número de steps se calcula multiplicando el número de fotos del dataset por las épocas de entrenamiento actuales dividido por el tamaño de los batches.

$$steps = \frac{fotos \cdot epocas}{batches} \tag{42}$$

Mientras la red se entrena, se puede monitorizar gráficamente su avance gracias a Tensorboard, como se muestra en la figura 64. Para ejecutarlo solo hay que escribir en la consola del entorno virtual tensorboard y dirección de la carpeta de logs que se genera durante el entrenamiento. Se proporcionará un host local que se puede abrir desde el navegador.

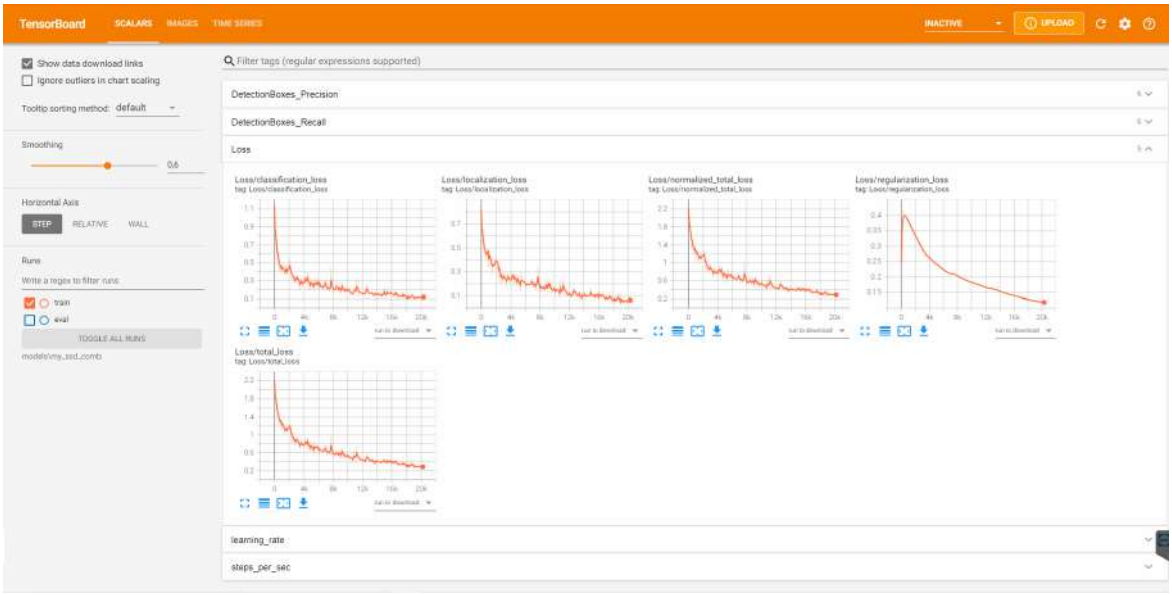
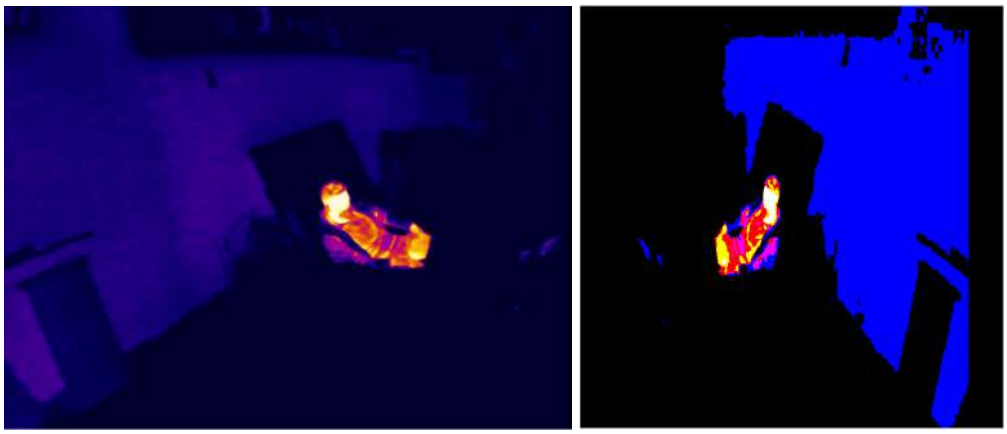


Figura 64: Tensorboard. Fuente:Autor

En el apartado *SCALARS* se podrán ver los diferentes parámetros de entrenamiento y evaluación de la red. Como ahora solo se está haciendo el entrenamiento, únicamente se muestra cómo van decreciendo los valores de los y cómo va variando la tasa de aprendizaje según va entrenándose la red.

En el apartado *IMAGES* pueden verse las últimas imágenes utilizadas en el entrenamiento. Estas aparecen, como se ha comentado, reescaladas a un tamaño de 640x640. Además, como se ha visto en la configuración de *pipeline.config*, se aumentan aleatoriamente los datos con volteos o recortes, como en el caso de la figura 65.



(a) Imagen original (b) Imagen reescalada y volteada

Figura 65: Ejemplo reescalado y aumento de datos. Fuente:Autor

Como se pueda apreciar en la figura 64, existen múltiples tipos de pérdida (losses) según la red que se emplee, que sumados entre sí dan lugar al loss total que se muestra por pantalla durante el entrenamiento.

Para la red Faster R-CNN, los valores de loss provienen del RPN, del clasificador y de la regularización.

Las pérdidas de la RPN son las siguientes:

- Pérdida de localización (RPN Localization loss): pérdida causada por el regresor de cuadros delimitadores para RPN.
- Pérdida de objeto (RPN Objectness loss): pérdida del clasificador que clasifica si un cuadro delimitador contiene un objeto o el ambiente.

Las pérdidas del clasificador son las siguientes:

- Pérdida de localización (Localization loss): pérdida causada por el regresor de cuadros delimitadores
- Pérdida de objeto (Classification loss): pérdida por la clasificación de objetos detectados en clases.

La pérdida por regularización es debida a a función de regularización utilizada para generalizar los datos mejor.

De esta manera, el coste o pérdida total de la red Faster R-CNN queda de la siguiente manera:

$$\text{loss} = \text{RPN\_localization\_loss} + \text{RPN\_classification\_loss} + \text{localization\_loss} + \text{classification\_loss} + \text{regularization\_loss}$$

SSD, al no tener RPN, solo cuenta con errores de localización, clasificación y regularización, por lo que la función de coste queda de la siguiente manera:

$$\text{loss} = \text{localization\_loss} + \text{classification\_loss} + \text{regularization\_loss}$$

Generalmente el entrenamiento se para cuando se llegan a unos valores de loss total de entre 5 y 15%, para evitar problemas de sobre-entrenamiento.

Si se deja entrenando durante muchos ciclos al modelo, puede que llegue a sobre-entrenarse, lo que se conoce como Overfitting. Al entrenarse continuamente en numerosas ocasiones con los mismos datos de entrenamiento, la red queda muy ajustada a estos datos. A la hora de emplear la red una vez entrenada, esta ya no generaliza lo aprendido para casos parecidos, aumentando el loss obtenido al evaluar nuevas imágenes. Como resultado se obtiene una red no funcional. La figura 66 muestra este problema.

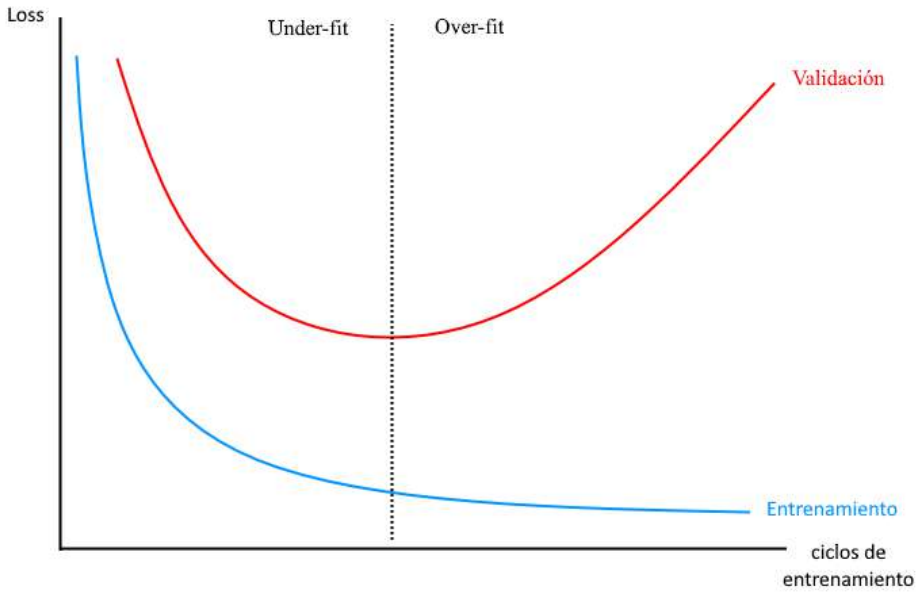


Figura 66: Overfitting. Fuente: Autor

El sobre-entrenamiento se puede evitar de varias formas, como modificando la red regularizando los pesos o parar el entrenamiento antes de tiempo. Esta última opción es la que se ha tomado en este proyecto.

Se evalúa el modelo, como se verá en el siguiente apartado, para diferentes guardados una vez el loss sea lo suficientemente bajo, comparando el valor de loss y viendo en cuál de ellos se produce el punto de inflexión de la curva de loss en la evaluación. Un ejemplo de este procedimiento se muestra en la figura 67, donde se puede ver el entrenamiento y evaluación del loss de la red SSD con dataset de día:

Para el caso de YOLO simplemente hay que configurar un fichero llamado *yolov3-custom.cfg* en el directorio config. Este fichero, al igual que *pipeline.config* en la API de Tensorflow, contiene información sobre la red, como pueden ser el reescalado de la imagen (416x416), valor de la tasa de aprendizaje (0.001), etc.

En este fichero también se encuentran los parámetros de número, tamaño y paso de filtros, la cantidad de padding y la función de activación (leaky ReLU) para cada capa de la red.

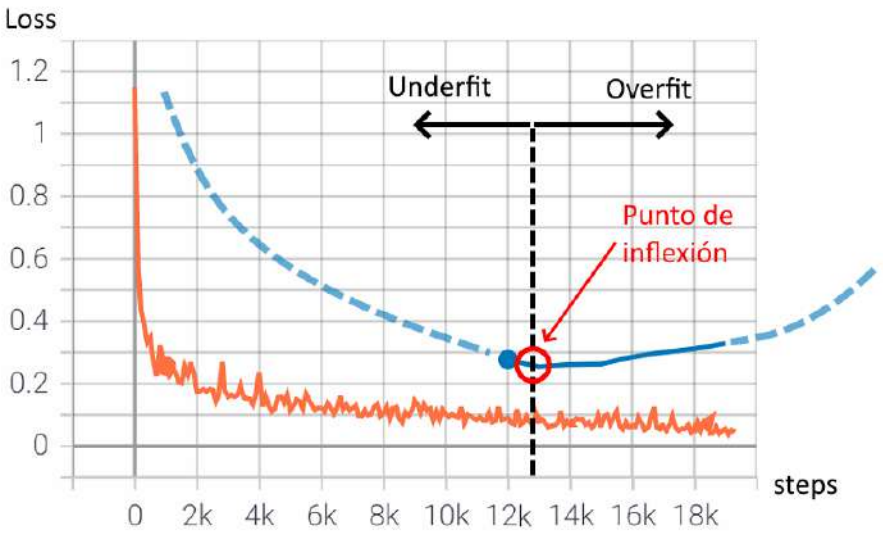


Figura 67: Ejemplo de overfitting para red SSD. Fuente: Autor

Para entrenar la red YOLO simplemente habría que ejecutar el *fichero train.py* del repositorio, especificando en este caso la dirección del fichero *yolov3-custom.cfg*, la dirección de las imágenes y sus etiquetas, la dirección de los pesos pre-entrenados *darknet53.conv.74*, y el tamaño de los batches, en este caso de 4.

Los parámetros de pérdida que contribuyen al loss total en la red YOLO son los siguiente:

- Pérdida de confianza (Confidence loss): es la suma de las pérdidas de confianza de las predicciones de un objeto y las de ausencia de objeto.
- Pérdida de posición (Position loss o x, y, w, h loss): pérdidas en la posición (x y) y ancho y largo (w h) de los cuadros delimitadores.
- Pérdida de clasificación (Classification loss): Al igual que en el caso de Tensorflow, es la pérdida por la clasificación de objetos detectados en clases.

Hay que tener en cuenta que YOLOv3 hace las detecciones a 3 escalas, por lo que se obtienen 3 valores de loss, loss_1, loss_2 y loss_3, que sumados dan loss total. Es por ello que para el posterior apartado de análisis se tome, en vez del loss total suma de los 3 losses, la media de estos.

loss=loss_1+loss_2+loss_3

En la expresión anterior loss_1, loss_2 y loss_3 valen

$$\text{loss\_i=loss\_x+loss\_y+loss\_w+loss\_h+loss\_conf+loss\_cls}$$

Las gráficas del entrenamiento de las redes para cada dataset pueden encontrarse en el Anexo II, en las figuras 89,90,91,92,93,94,95,96 y 97.

5.4. Evaluación

La evaluación es el proceso por el cual se comprueba la eficacia de un modelo, midiendo los valores de unas métricas llamadas precisión y recall que se obtienen al usar la red sobre un dataset de evaluación.

Para entender qué son la precisión y recall conviene recordar el concepto de IoU, que se explicó en el apartado teórico. Consiste en medir la instersección de dos cuadros delimitadores y viene dado por el área de superposición entre el cuadro delimitador predicho y el cuadro delimitador verdadero dividido por el área de unión entre ellos.

Unas definiciones importantes son las siguientes [60]:

- Verdadero Positivo (True Positive o TP): corresponde a una detección correcta, cuando el IoU está por encima de cierto umbral.
- Falso Positivo (False Positive o FP): corresponde a una detección incorrecta, cuando el IoU está por debajo de cierto umbral.
- Falso Negativo(False Negative o FN): corresponde al caso de no detectar un cuadro delimitador verdadero.
- Verdadero Negativo (True Negative o TN): corresponde a todos los posibles cuadros delimitadores desechados correctamente por no contener el objeto. Estos son la mayoría de los cuadros predichos en una imagen, por lo que este parámetro no se usa.

El umbral que se mencionó en TP y FP depende de la métrica que se esté usando, siendo normalmente 50 %, 75 % o 95 %.

La precisión es la tasa de predicciones positivas correctas, es decir, cuan acertadas son las predicciones. Viene dada por la expresión 43.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{todas\ las\ predicciones} \tag{43}$$

El recall es la tasa de verdaderos positivos detectados entre todos los cuadros delimitadores verdaderos, es decir, mide cuan bueno es el modelo detectando todas las clases positivas. Viene dada por la expresión 44.

$$R = \frac{TP}{TP + FN} = \frac{TP}{todas\ los\ verdaderos} \tag{44}$$

Una métrica que se utiliza comúnmente en la evaluación de las detecciones es mean Average Precision (mAP). Para entender que es primero hay que explicar la curva precision-recall y la métrica AP.

La curva precision-recall es una curva obtenida al representar, como su nombre indica, los valores de precisióny recall obtenidos al evaluar dististas imágenes en función de los distintos umbrales de IoU. Tiene normalmente una forma escalonada, como se muestra en la figura 68.

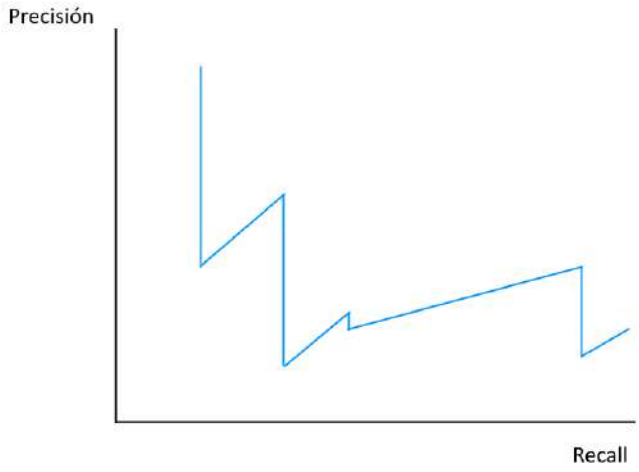


Figura 68: Ejemplo de curva precision-recall. Fuente: Autor

De esta curva se puede obtener la métrica AP, que calcula el área encerrada en la gráfica precision-recall para diferentes umbrales de IoU.

En la figura 69 el valor de AP se obtiene de sumar las áreas A1, A2, A3 y A4.

La métrica mAp es la media de la métrica Ap calculada para todas las clases. Por ejemplo, mAp@0.5 significa que ese mAP se ha calculado con un IoU=0.5.

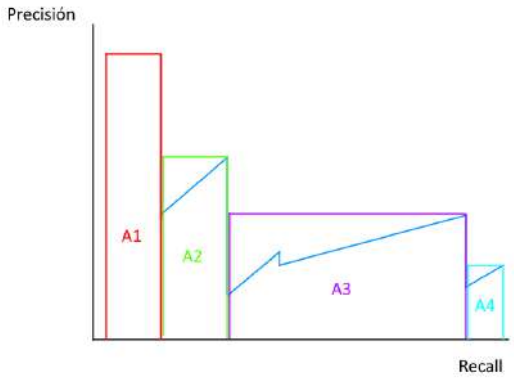


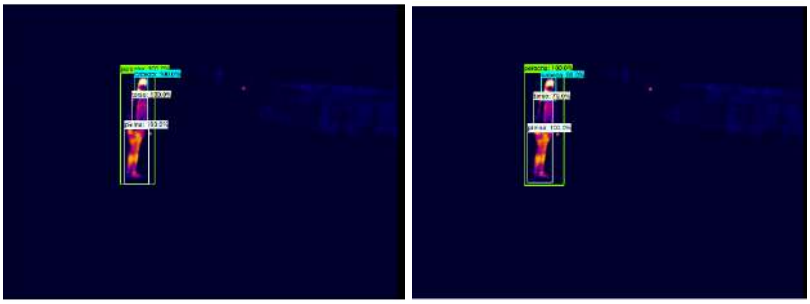
Figura 69: Ejemplo de cálculo de AP. Fuente: Autor

En Tensorflow es posible llevar a cabo de manera simultánea el entrenamiento y la evaluación, de manera que se evalúe el modelo en cada checkpoint creado, que en este caso son cada 1000 steps.

Sin embargo, esto requiere una gran capacidad computacional, por lo que si dicha capacidad es insuficiente se dará el error Allocator ran out of memory trying to allocate..., igual que en el caso de emplear batches altos durante el entrenamiento. Por ello se realizó la evaluación de los modelos únicamente a partir de un valor de loss suficientemente bajo, debido a que cada vez que se hacía la evaluación debía interrumpirse el entrenamiento.

Es por ello que en Tensorboard únicamente aparecen en el apartado *SCALARS* un par de valores puntuales de mAP y recall, correspondientes a cada evaluación realizada para las redes de Tensorflow, por lo que no se aporta gran información sobre la evolución de estos parámetros.

En el apartado *IMAGES*, se pueden ver las imágenes que se están evaluando con sus cuadros delimitadores verdaderos, junto a las predicciones de la red. Un ejemplo de esto se ve en la figura 70



(a) Evaluación (b) Original

Figura 70: Ejemplo de evaluación. Fuente: Autor

Para las redes entrenadas con Tensorflow se obtienen valores de mAP y recall según diferentes valores de IoU (0.5:0.95, 0.5 o 0.75), según diferentes tamaños de las clases detectadas (pequeñas, medianas o grandes) y, para el caso del recall, según el número máximo de clases detectadas (1, 10 o 100).

Para YOLO destaca mencionar que se ha podido realizar el entrenamiento y evaluación de manera simultánea. Las evaluaciones se realizaban cada época de entrenamiento.

La red YOLO cuenta además con otras metricas, como son la precisión de clase (cls_acc), que es la precision de una predicción de clase siempre y cuando un objeto esté presente en la imagen; y la coonfianza de objeto (cnf), que es la confianza media con la que se hace la predicción de que hay un objeto en la imagen.

Al igual que en el caso del loss del entrenamiento, las métricas aparecen repartidas en 3, debido a las 3 escalas con las que trabaja YOLOv3. Se tomará de la misma manera en los resultados el valor medio de estos.

Las gráficas de mAP y recall de la evaluación de la red YOLOv3 para cada dataset pueden encontrarse en el Anexo II, en las figuras 95, 96 y 97.

Una vez evaluados lar redes para cada dataset, pueden exportarse los modelos. De esta manera puede guardarse completamente la red y usarse para otras aplicaciones, como por ejemplo la comprobación de los resultados en video. Este paso no es necesario para la red YOLOv3.

6. PRUEBAS

Como se ha visto en el apartado de Visión Artificial, hay múltiples arquitecturas que implementan CNNs para la tarea de detección de objetos. Cada una de ellas presenta una serie de ventajas y de inconvenientes respecto a otras, como pueden ser la precisión y velocidad con la que trabajan.

Se han ideado una serie de pruebas que buscan corroborar algunos aspectos teóricos y con el fin de hallar la red que mejor opera para detección de víctimas.

6.1. Influencia del contraste de temperaturas a la detección de víctimas

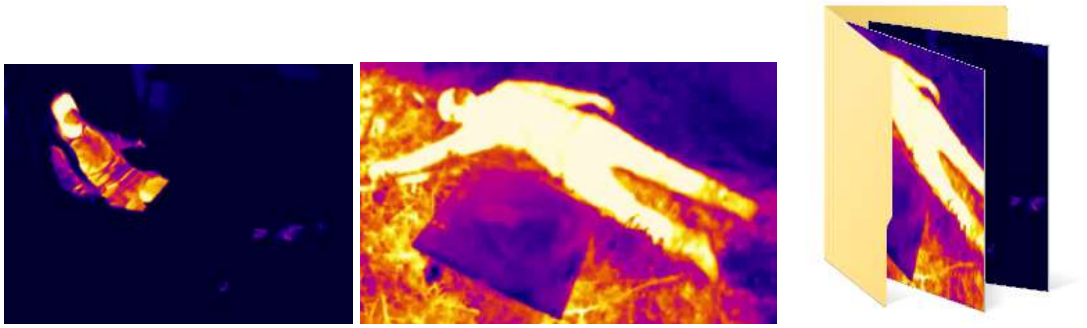
Uno de los factores que se quiere estudiar es la influencia del contraste de las imágenes térmicas en la detección. Este contraste viene dado entre la diferencia de temperaturas captadas por la cámara térmica entre la persona o víctima a detectar y el ambiente.

Hay diferentes factores que influyen al ambiente, y con ello al contraste de lo que se pretende detectar respecto al ambiente, tal y como se mencionaron en el apartado de condiciones ambientales en termografía, como pueden ser la radiación solar, corrientes de aire, polución, etc. Sin embargo, el que más influye es la temperatura ambiente. A temperatura ambiente similar a la temperatura media de las personas en exteriores, de en torno a 20-25 °C, el contraste ambiente-persona es muy bajo.

Se pretende comprobar la influencia del contraste térmico en la imagen realizando pruebas que empleen 3 datasets distintos, que se muestran en la figura 71.

- Noche : dataset que emplea imágenes grabadas de noche, donde la temperatura es menor, luego con mayor contraste de temperaturas entre personas y el ambiente.
- Día : dataset que emplea imágenes grabadas de día, a una temperatura aproximada de 25°C, luego con menor contraste de temperaturas entre personas y el ambiente.
- Combinado : dataset que emplea las imágenes de ambos conjuntos de noche y día.

De esta manera, se pretende realizar un total de 9 pruebas relacionadas con la influencia del contraste, probando todas las combinaciones de redes neuronales y de datasets. De las 872 imágenes etiquetadas, 518 pertenecen al dataset de noche y 354 para el de día.



(a) Dataset Noche (b) Dataset Día (c) Dataset Combinado

Figura 71: Distintos datasets empleados en el entrenamiento. Fuente: Autor

En el dataset de día, aparte de tener menor constraste persona-entorno, hay que tener en cuenta la influencia de la radiación solar y celeste. Materiales con alta reflectividad, como los metálicos pulidos, pueden reflejar esta radiación e indicar que están a una temperatura mayor a la que realmente están, pudiendo dar conflicto con la detección de personas si aparentan estar a una temperatura similar a la de una persona, de en torno a los 25 °C. También es objetivo de estudio ver la influencia de este factor sobre las detecciones en el dataset de día.

6.2. Detección de diferentes partes del cuerpo de víctimas

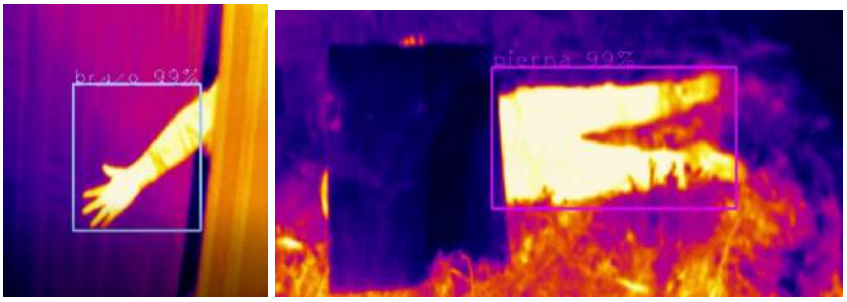
Otra prueba que se quiere realizar está relacionada con la detección de diferentes partes del cuerpo. Debido a que en accidentes o desastres las víctimas pueden quedar encerradas o sepultadas en escombros, la cámara puede llegar a ver algún miembro o extremidad, pero no la silueta del cuerpo entero.

Es por ello que se quiere entrenar a los modelos para que, aparte de detectar personas, detecten diferentes partes del cuerpo. Por ello en cada imagen de los diferentes datasets se utilizarán, a parte de la etiqueta 'persona', etiquetas adicionales, siendo estas 'cabeza', 'brazo', 'pierna' y 'torso'.

Se ha desarrollado un fichero de Python que emplea la librería OpenCV para editar el color de las detecciones según la clase detectada, cambiando el color de los cuadros delimitadores, nombres de las clases y la confianza de la detección en la imagen. Además, se ha mejorado el apartado visual con el que se representaban dichos parámetros.

Se planteará también cuál de estos miembros es más fácil de detectar y si alguno de estos puede llegar a detectar mejor personas que la propia etiqueta de persona.

A continuación se mostrarán ejemplos de detección de un brazo y dos piernas en la figura 72.



(a) Brazo (b) Piernas

Figura 72: Detección de distintas partes del cuerpo. Fuente: Autor

6.3. Distinción entre persona normal y víctimas

Hasta ahora se ha mencionado la detección de clase 'persona' pero no de 'víctima', que realmente es lo que se pretende detectar con este trabajo. Distinguir a una persona de una víctima puede llegar a ser una tarea muy complicada. El método más lógico para distinguir personas de víctimas sería etiquetar en los datasets las personas como 'persona' y las víctimas como 'víctimas'. Sin embargo, debido a la similitud que tendrían entre sí estas clases se tendría un error de clasificación muy elevado.

Un método más sencillo y que puede llegar a ser efectivo es analizar la forma del cuadro delimitador. Las víctimas de accidentes normalmente estarán tumbadas en el suelo o en posiciones similares. Este hecho puede aprovecharse midiendo el ancho y largo de los cuadros delimitadores de las imágenes. Al recuadrar una víctima en posición tumbada o derivados, la anchura de dichos cuadros generalmente va a ser mayor a la altura de estos. Al contrario, al recuadrar a personas, de pie o sentadas, la altura de dichos cuadros normalmente va a ser significativamente mayor al ancho de estos.

Se han analizado varias imágenes que muestran tanto transeuntes (detallar) como víctimas y se ha llegado a la conclusión de que las víctimas tienen una relación largo-ancho menor a 0.75.

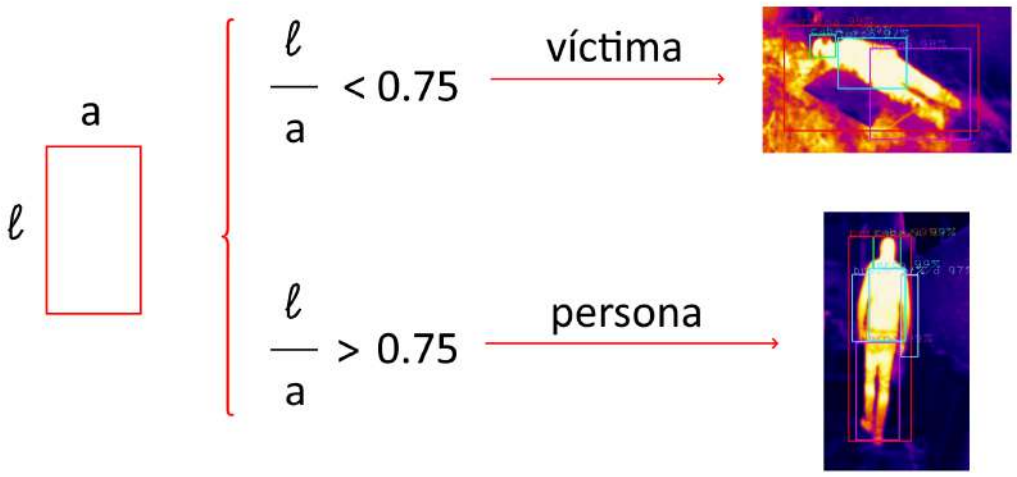


Figura 73: Relación largo-ancho para detectar víctimas. Fuente: Autor

Se ha implementado este método en un fichero de Python, de manera que cada vez que se detecte una clase 'persona' con una tasa l/a menor a 0.75, se imprima en la consola que se ha detectado una posible víctima junto a las coordenadas de la imagen de esta.

Se pretende comprobar la validez de este último método y su eficacia a la hora de detectar y distinguir víctimas.

6.4. Comprobación de la detección en vídeos grabados por robot de campo

La ultima prueba pretende comprobar la eficacia de la detección de víctimas utilizando el modelo YOLOv3 entrenado con dataset combinado evaluándolo con videos grabados con el robot de campo Unitree A1, que se muestra en la figura 74.

Este TFG forma parte del proyecto TASAR del grupo red Robótica y Cibernética - ROBCIB, uno de sus objetivos recae en la detección de víctimas en entornos de desastre utilizando robots móviles y fuentes sensoriales. Es por ello que las redes entrenadas en este TFG para la detección de víctimas serán validadas para la aplicación que se está desarrollando en el proyecto TASAR usando videos que el Robot Unitree A1 ha grabado previamente.

Figura 74: Robot Unitree A1. Fuente: Autor

Otro aspecto que se ha de comprobar es la eficacia de las redes entrenadas en interiores. Se ha enfocado la mayor parte del dataset a escenarios de exteriores, con pocas imágenes de interiores. Por ello se probará la eficacia del modelo en interiores con videos grabados por el robot Unitree A1. La figura 75 muestra un escenario de interior donde se han grabado videos con el robot Unitree A1.

Figura 75: Escenario de interior. Fuente: Autor

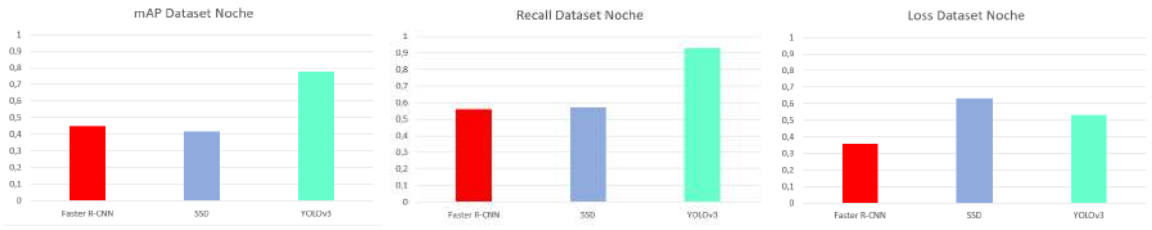
7. RESULTADOS Y DISCUSIÓN

En este apartado se analizarán y discutirán los resultados de las pruebas propuestas.

Se hará referencia en numerosas ocasiones en este apartado al Anexo II: Figuras y tablas adicionales, donde se encuentran las diferentes tablas que recogen todos los datos brutos obtenidos durante el entrenamiento y evaluación de las distintas redes con cada dataset.

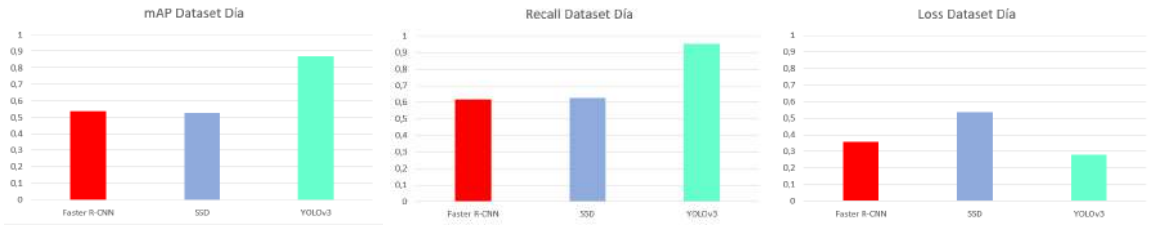
7.1. Análisis y discusión de los resultados de la prueba 6.1

La primera prueba era medir la influencia del contraste de temperaturas a la detección de víctimas, con el objetivo encontrar la combinación de red y dataset idónea. A continuación, se muestran las figuras 76, 77 y 78, que muestran los valores de precisión (mAP), recall y loss para de las redes para cada dataset, obtenidos de las tablas del Anexo II.



(a) mAP (b) Recall (c) Loss

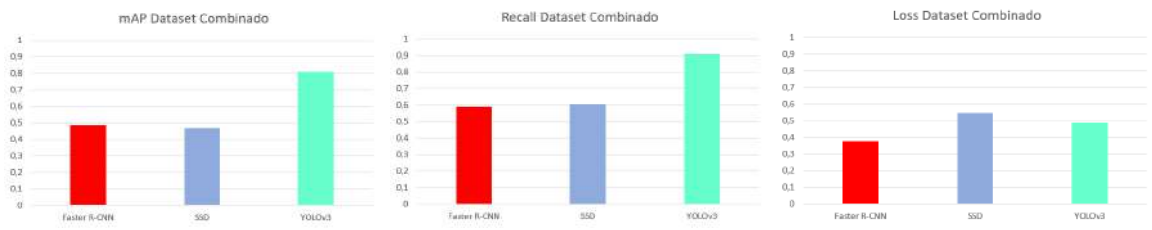
Figura 76: mAP, Recall y Loss para las redes con dataset de noche. Fuente: Autor



(a) mAP (b) Recall (c) Loss

Figura 77: mAP, Recall y Loss para las redes con dataset de día. Fuente: Autor

Analizando estos diagramas se puede ver que la red YOLOv3 es muy superior en mAP y recall para todos los datasets. También cuenta con los valores de loss más bajos de todos, excepto en el caso del dataset de noche, donde el menor loss es el de la red Faster R-CNN. De aquí se extrae que, en base a resultados de evaluación, sin ninguna duda la red YOLOv3 es la mejor de las tres.



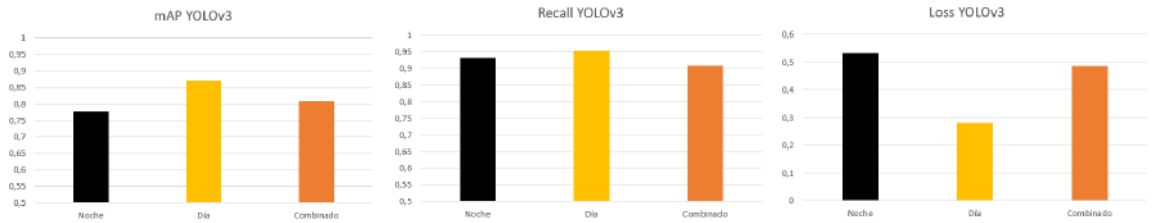
(a) mAP (b) Recall (c) Loss

Figura 78: mAP, Recall y Loss para las redes con dataset combinado. Fuente: Autor

Otro hecho que hay que tener en cuenta a la hora de escoger una red sobre otra es la velocidad en inferencia. Hay una gran diferencia en los frames por segundo en inferencia para cada red. La red Faster R-CNN, a pesar de tener unos buenos valores de precisión y recall y poco loss, es sumamente lenta en inferencia, trabajando a muy pocos frames por segundo. Pasa al contrario para la red SSD, donde sí trabaja a una tasa mayor de fps pero cuenta con una mayor pérdida. YOLOv3 tiene tanto buenos valores en los parámetros de detección como velocidad de inferencia.

Hay estudios previos que realizaron comparativas de la velocidad de ejecución en inferencia de las redes, como el de S A Sanchez et al [61], en el que utilizaba una GPU de 12 GB de RAM para comparar el rendimiento y velocidad en inferencia de múltiples redes. Los resultados obtenidos fueron una tasa media de fps para Faster R-CNN de 7, para SSD de 19 y para YOLOv3 de 45.

En la figura 79 se muestra el análisis de los resultados de YOLOv3 comparando los datasets.



(a) mAP (b) Recall (c) Loss

Figura 79: Comparación del mAP, Recall y Loss de los datasets para YOLOv3. Fuente: Autor

Observando la figura 79, se puede observar que los mayores valores de precisión y recall y los menores de loss se obtuvieron para el dataset de día, seguido del combinado y finalmente el de noche. Esto se opone totalmente a los resultados esperados, que esperaban que, al haber mayor contraste y menor influencia de otras condiciones

ambientales como la radiación solar, el dataset de noche sería el idóneo. Las redes entrenadas con dataset combinado ofrecen valores de precisión, recall y loss no muy por debajo de los del dataset de día, con la ventaja de que son más polivalentes, funcionando correctamente tanto de día como de noche.

7.2. Análisis y discusión de los resultados de la prueba 6.2

Un hecho que destaca de los diagramas 76, 77 y 78 es el alto valor de loss que tiene la red SSD en comparación con el resto de redes para todos los datasets. Esto puede suceder por múltiples motivos, pero el más probable es que esté relacionado con los resultados de la segunda prueba, que es la detección de diferentes partes del cuerpo de víctimas. Para la realización de esta prueba se etiquetaron los datasets con 5 clases: 'persona', 'cabeza', 'brazo', 'pierna' y 'torso'. Debido a la similitud de algunas de estas clases entre sí, como pueden ser los brazos y piernas en las imágenes térmicas, se produce un gran aumento de las pérdidas de clasificación. Esto se puede ver en las tablas 8, 9 y 10 del Anexo II.

Este mismo problema lo presenta en menor medida la red Faster R-CNN, como se puede ver en 8, 9 y 10, pero no la red YOLOv3, donde se puede ver en 11, 13 y 15 que los valores de pérdidas de clasificación son prácticamente despreciables. Este hecho se pone de manifiesto a la hora de aplicar los detectores a videos previamente grabados.

Al observar los resultados obtenidos al utilizar la red SSD, puede verse claramente que la red no etiqueta correctamente. Como ya se vio en el apartado teórico, esta red tiene complicaciones a la hora de utilizar imágenes con mala calidad debido a que SSD comienza a utilizar la imagen para la detección cuando esta ya se ha reducido mucho de tamaño por las sucesivas capas convolucionales y de pooling.

Esto provoca que clases parecidas, al reducir su tamaño, se vean prácticamente idénticas. Esto dificulta enormemente la clasificación, ya que en la capa softmax se obtendrán probabilidades parecidas de pertenecer a una clase u otra. Por ejemplo, al analizar un brazo en una imagen podrían obtenerse probabilidades de que sea un brazo del 50 % y de que sea una pierna del 50 %. Al exigirse en inferencia un cierto umbral de certeza de clase, de por ejemplo el 60-70 %, estas clases no se mostrarán por pantalla.

Otro problema relacionado que existe para esta red es que se trata al ambiente como clase, la clase '0'. Es por ello que en múltiples ocasiones aparecen en el fondo detecciones erróneas de personas, como la que se puede observar en la figura 80. Esto puede ser debido a que los cuadros de etiquetado de la clase persona tengan una gran proporción

de fondo en ellos, por lo que puede haber problemas de clasificación entre la clase 'persona' y la '0'.

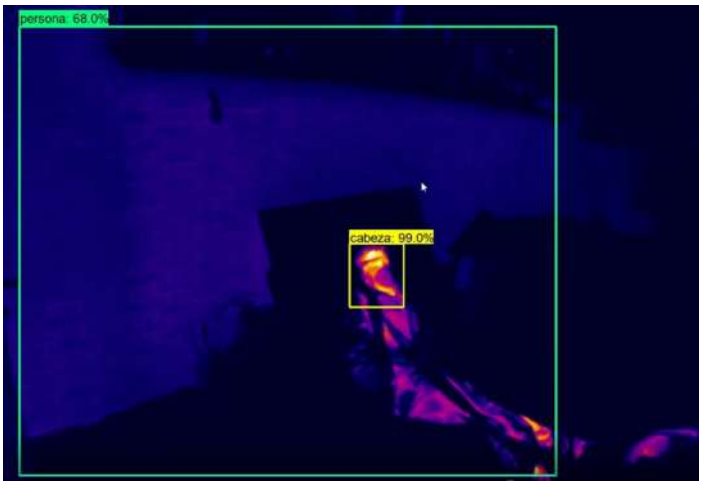


Figura 80: Error en la detección de víctimas con red SSD. Fuente: Autor

Las única clase que se etiqueta regularmente bien en esta red es 'cabeza', por ser una clase bien diferenciada de las demás. Se ha observado además que se detectan correctamente las clases 'brazo', 'pierna' y 'torso' siempre y cuando estén a una escala muy grande, es decir, cuando estos están muy próximos a la cámara. Es por ello que se concluye afirmando que, en base al gran valor de loss debido a errores de clasificación, esta red no apropiada para la detección de víctimas.

Para Faster R-CNN el error de clasificación sigue siendo apreciable, pero es mucho menor al de SSD. Los resultados en inferencia son en su mayoría correctos, pudiendo afirmarse en base a los parámetros de precisión, recall y loss que esta red es viable para la detección de víctimas, pero no la mejor opción. En la figura 81 se ve un ejemplo de detección de víctima con Faster R-CNN.

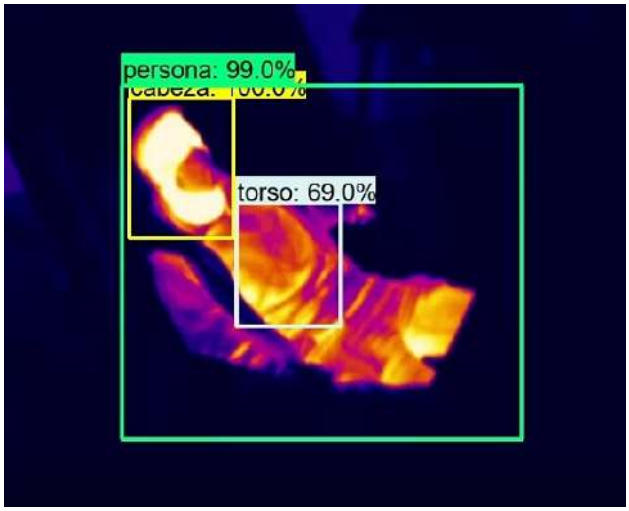


Figura 81: Ejemplo detección con Faster R-CNN. Fuente: Autor

YOLOv3 no presenta este tipo de problemas, probablemente debido a que no se usa una capa sofmax para la clasificación, sino que utiliza regresión logística, lo que hace que dos clases parecidas no se excluyan mutuamente a la hora de asignar la probabilidad de clase. Los resultados en inferencia son muy satisfactorios, mostrando una gran precisión y recall y bajo loss en las predicciones, siendo YOLOv3 una red viable para la detección de víctimas. En la figura 82 se ve un ejemplo de detección de víctima con YOLOv3.

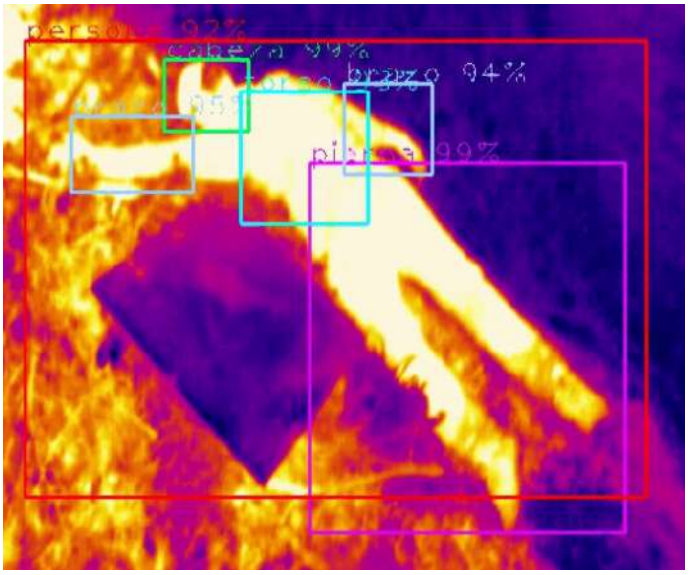


Figura 82: Ejemplo detección con YOLOv3. Fuente: Autor

Un dato para destacar es que SSD utiliza un sistema para evitar predicciones de clase duplicadas llamado non-maximum suppression (nms), como se vió en teoría, por el cual únicamente toma la predicción de cuadro con mayor IoU. La figura 83 muestra un etiquetado duplicado de la clase persona en la red Faster R-CNN.

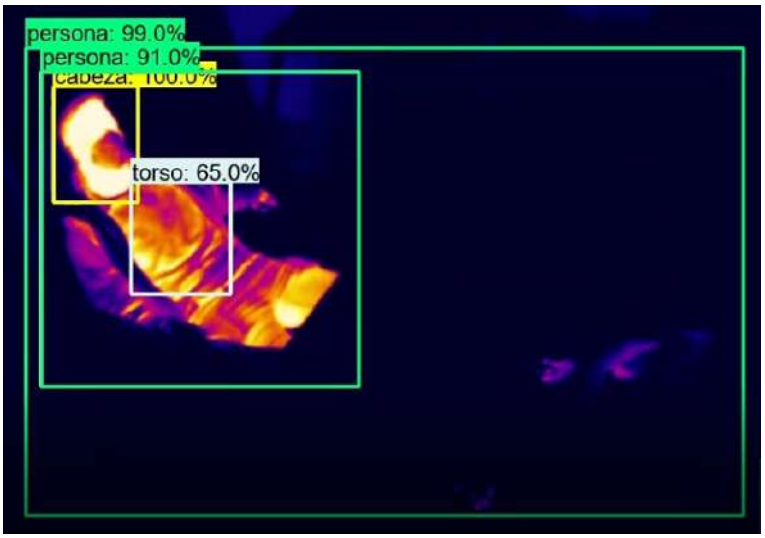


Figura 83: Error predicciones duplicadas en Faster R-CNN. Fuente: Autor

Para saber qué partes del cuerpo son más fáciles de detectar, se pueden analizar los resultados obtenidos en la precisión de clase de la evaluación de la red YOLOv3 de las tablas 12,14 y 16, representando en la figura 84 la precisión media de cada clase para los tres datasets.

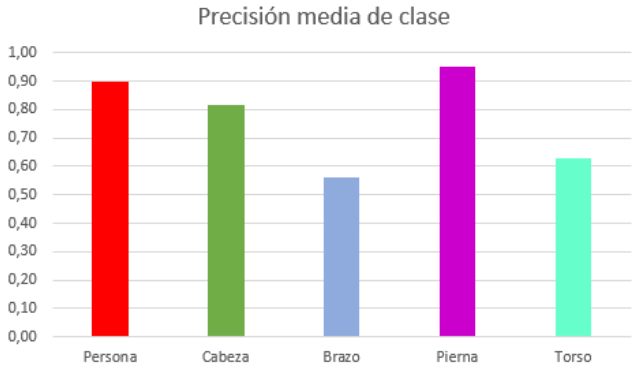


Figura 84: Precisión media de clase para YOLOv3. Fuente: Autor

Como se puede ver en la figura 84, todas las clases tienen una buena precisión media. Las clases 'persona', 'cabeza' y 'pierna' tienen una precisión muy alta, llegando a 0.95 para el caso de 'pierna', mientras que las clases 'brazo' y 'torso' tienen una precisión algo menor, pero dentro de los márgenes aceptables.

7.3. Análisis y discusión de los resultados de la prueba 6.3

La prueba que busca evaluar la distinción entre personas normales y víctimas se ha llevado a cabo evaluando distintos vídeos en los que aparecían tanto transeúntes como supuestas víctimas en la red YOLOv3. Se evaluó la eficacia del método de analizar la relación largo-ancho de la persona detectada para distinguir ambos casos, y los resultados obtenidos fueron más que satisfactorios. Se mostraba por pantalla, cada vez que se detectaba en la imagen una posible víctima, un mensaje que alertaba de la situación y las coordenadas de la imagen en donde se encontraba la víctima, como se ve en la figura 85.

Una posible mejora de este sistema sería que se alertase cada vez que identifica una nueva víctima, y no cada vez que se detecta esa misma víctima en las sucesivas imágenes. Esto se podría hacer de diferentes maneras, como por ejemplo alertar cuando las coordenadas de la nueva víctima estén lo suficientemente alejadas de las coordenadas de la víctima inicial. Sin embargo, esto no ha sido implementado en este trabajo.

Otra posible mejora sería utilizar el mismo método aplicado a otras clases, es decir, detectar víctimas a partir de la relación largo-ancho de los cuadros delimitadores de las

clases 'brazo', 'pierna' y 'torso'. Esta mejora tampoco ha sido implementada.

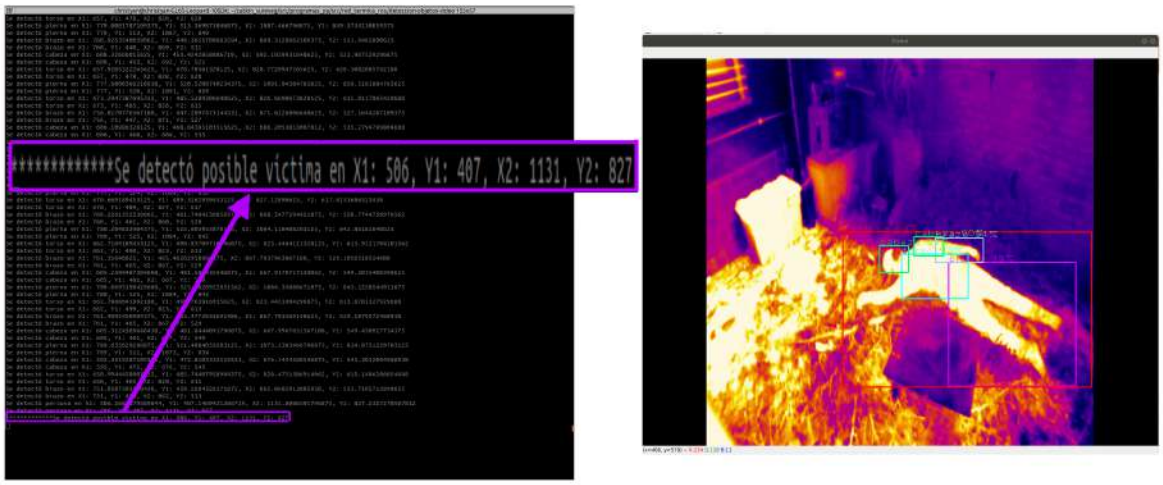


Figura 85: Ejemplo de alerta por pantalla de una posible víctima. Fuente: Autor

7.4. Análisis y discusión de los resultados de la prueba 6.4

Esta prueba buscaba evaluar el funcionamiento del modelo YOLOv3 entrenado con dataset de día en vídeos grabados por el robot Unitreee. El resultado fue bueno, aunque presentaba un error de clasificación entre las clases de piernas y brazos ligeramente mayor a las pruebas anteriores. A pesar de ello, el modelo es perfectamente válido, detectándose todo correctamente en la mayoría de las ocasiones. De esta manera también se ha probado la eficacia de YOLOv3 en ambientes de interior. A continuación, en la figura 86 se muestra uno de los resultados obtenidos en esta prueba.

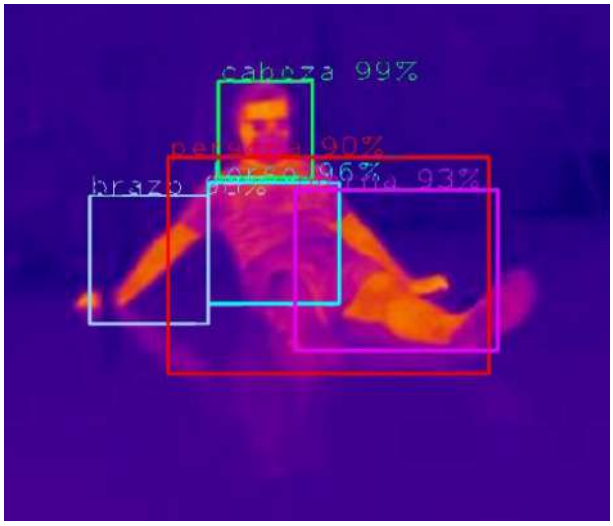


Figura 86: Resultado de la inferencia en vídeo grabado por Robot Unitree A1. Fuente: Autor

8. CONCLUSIONES

Los grandes avances en Visión Artificial que se han dado en los últimos años, debido a la incorporación de redes neuronales a los modelos de visión, han supuesto una revolución en el sector. Son múltiples las áreas de aplicación, pero la más relevante es el procesamiento de imágenes con redes neuronales para clasificar, identificar o detectar objetos en imágenes.

Existen multitud de arquitecturas formadas por redes neuronales, como pueden ser Faster R-CNN, SSD o YOLO, que implementan una serie de algoritmos capaces de llevar a cabo la detección de objetos, teniendo cada una de ellas una serie de ventajas e inconvenientes.

El procesamiento de imágenes térmicas mediante el uso de estas redes neuronales ha permitido llevar a cabo con éxito la tarea de detección de víctimas.

Para lograrlo, se necesitó configurar entornos virtuales, donde se instaló python y las distintas librerías de Deep Learning y Visión Artificial; se generaron una serie de datasets, consistentes en imágenes térmicas con un etiquetado que indicaba la presencia o no de una víctima; se entrenaron las redes con dichos datasets, con el objetivo de adaptar dichas redes a la tarea de detección y minimizar las pérdidas; para terminar evaluando las redes obtenidas, midiendo distintos parámetros como la precisión para ver si el funcionamiento es correcto.

Finalmente se realizaron una serie de pruebas durante el desarrollo de este proyecto, que tenían el objetivo de comprobar la viabilidad de las redes neuronales para detectar víctimas en distintas condiciones, analizando y discutiendo posteriormente los resultados.

A continuación, se muestran un resumen de las conclusiones extraídas del análisis de los datos obtenidos en las distintas pruebas que se han llevado a cabo en este proyecto:

- La red SSD presenta numerosos inconvenientes para la tarea de detección de víctimas, que ocasionan que tenga un gran valor de loss, lo cual imposibilita su correcta aplicación.
- La red Faster R-CNN mejora considerablemente los resultados obtenidos respecto a SSD, pero su lenta velocidad en inferencia hace prácticamente imposible su ejecución a tiempo real.

- La red YOLOv3 cuenta con mAP y recall mucho mayoresa las otras dos redes, que rondan los valores de 85 % y 95 % respectivamente, y un loss ligeramente menor, de en torno al 35 %. Además, YOLOv3 tiene la mayor velocidad en inferencia a las demás redes. Por ello se ha llegado a la conclusión de que esta red es la adecuada para la detección de víctimas.
- En cuanto a las condiciones en las que mejor se detecta, las redes entrenadas con el dataset de día han mostrado mejores valores en los parámetros de detección. Por ejemplo para el caso de YOLOv3, esta presenta un mAP un 12 % mayor comparado con el mAp de noche y un 8 % mayor al combinado, presenta recall prácticamente iguales a los otros datasets y tiene un loss 88 % menor comparado al de la noche y un 72 % menor al combinado. Aunque las redes entrenadas con el dataset combinado presentan valores inferiores de dichos parámetros, su mayor robustez, por poder trabajar tanto en ambiente de noche como de día, las convierten en una buenas alternativas.
- Las clases que mejor se detectan son 'persona','cabeza' y 'pierna', con una precisión de clase en torno al 90 %, aunque 'brazo' y 'torso', con precisión de clase que ronda 60 %, se detectan correctamente la gran mayoría de las veces.
- El método que mide la relación largo-ancho de la clase 'persona' para distinguir transeúntes de víctimas es un método viable que proporciona resultados satisfactorios.
- Las evaluaciones en videos grabados por el robot Unitree A1, aun teniendo un error ligeramente superior a las pruebas anteriores, sigue mostrando que es perfectamente viable la detección de víctimas en robots móviles. Además, se ha comprobado la eficacia de la red en escenarios de interior.

9. IMPACTO DEL TRABAJO

9.1. Aplicación y beneficios

Como se ha mencionado anteriormente, el objetivo principal de este trabajo es detectar víctimas en entornos de desastre. Se ha podido estudiar la efectividad de los distintos modelos de redes neuronales para desarrollar dicha tarea y se han desarrollado múltiples pruebas que evalúan la eficacia de estos modelos en distintas condiciones.

Las aportaciones de este trabajo pueden llegar a ser un pequeño avance en las tareas de rescate y detección de víctimas. La investigación en este campo puede lograr que, en un futuro próximo, se dispongan de medios efectivos para paliar y reducir los daños en accidentes y catástrofes al mínimo.

Por otro lado, algunas de las otras conclusiones que se han extraído al analizar los resultados de las distintas pruebas pueden servir de apoyo para futuros proyectos que involucren trabajar con imágenes térmicas en tareas de detección.

9.2. Impacto social, medioambiental y económico del trabajo

La motivación de cualquier proyecto de ingeniería es aplicar conocimientos técnicos con el fin de mejorar y facilitar la vida de las personas. Si bien se busca un impacto positivo en la sociedad y en el medio ambiente, es necesario conocer los posibles consecuencias negativas que se puedan llevar a cabo.

Por ello, debe de garantizarse la sostenibilidad de los proyectos tal y como se indica en el Informe Brundtland [62] presentado por la ONU, de manera que se garantice un equilibrio entre la parte económica, social y medioambiental de un proyecto.

Este proyecto tiene los siguientes impactos sociales, ambientales y económicos:

En cuanto al impacto social, el proyecto está destinado a la detección de víctimas en entornos de desastre usando cámaras térmicas montadas en robots de campo. Esto puede llegar a suponer un gran impacto a la hora de rescatar personas, al optimizar y agilizar los procesos de búsqueda, complementando la tarea de los operarios. De esta forma se podría incrementar la tasa de éxito durante las operaciones de rescate.

Respecto al impacto medioambiental, para la implementación de los modelos de detección de víctimas es necesario el uso de equipos o robots dotados con cámaras térmicas. La fabricación, mantenimiento y posterior desecho de estos equipos puede llegar a suponer un impacto medioambiental negativo. Sin embargo, hay que tener en cuenta que el uso de estos equipos podría tener un beneficio en medio ambiente al servir de apoyo a la maquinaria pesada en tareas como la de descombro al dar información de la localización de las víctimas, lo que permite el uso más eficiente de la maquinaria pesada, reduciendo el área de trabajo. De esta forma se puede reducir la emisión de contaminantes.

En lo referente al impacto económico, el desarrollo del proyecto no ha supuesto una gran inversión económica. El único gran desembolso de este proyecto viene dado por la adquisición de la cámara térmica. Si la implementación del proyecto se realizara en robots móviles, como en el caso del proyecto TASAR, la inversión económica sería mucho más elevada.

Teniendo en cuenta los beneficios sociales que pueda aportar este proyecto, el posible impacto negativo económico y ambiental es perfectamente asumible. De esta forma puede afirmarse que este proyecto es compatible con el desarrollo sostenible.

9.3. Futuras líneas de investigación

Una posible línea de investigación futura podría ser la utilización de un dataset mucho mayor y más variado, que muestren diferentes condiciones climáticas, distintas humedades, etc., para entrenar redes que detecten víctimas.

Debido a que en multitud de aplicaciones se requieren sistemas con menor consumo energético o instrumentos de menor tamaño o coste, una posible investigación puede ser la aplicación de redes neuronales para la detección de víctimas utilizando sistemas e instrumentos limitados o con pocos recursos, como cámaras con muy baja resolución o el uso de sistemas embebidos como Jetson nano para hacerlos portables.

Conforme pasan los años, van surgiendo nuevas y mejoradas arquitecturas de redes neuronales en visión, por lo que un trabajo futuro podría ser realizar un procedimiento similar al realizado en este TFG y obtener los resultados correspondientes a dichas redes.

Como se ha mencionado con anterioridad, este TFG está incluido dentro del proyecto TASAR del grupo de Robótica y Cibernética del CAR-CSIC. Este proyecto utilizará los modelos entrenados en este TFG para la detección de víctimas a tiempo real con un robot móvil que dispone de una cámara infrarroja.

A título informativo y de documentación, se describen en el Anexo III los avances que se están realizando en este momento para lograr la detección a tiempo real.

10. BIBLIOGRAFÍA

Referencias

[1] Cámara infrarroja optris PI 640. https://www.optris.es/camara-infrarroja-optris-pi-640. Accessed: 2021-6-23.

[2] Noticias onu: Pese al aumento de las amenazas de origen natural en el siglo XXI, los países siguen "sembrando las semillas de su destrucción". https://news.un.org/es/story/2020/10/1482242, October 2020. Accessed: 2021-6-23.

[3] Búsqueda y rescate. https://es.wikipedia.org/wiki/B Acceso: 22-6-2021.

[4] B. siciliano y o. khatib. springer handbook of robotics. springer handbooks. springer international publishing, 2016.

[5] Obras Urbanas. Auxdron lifeguard: el dron diseñado para salvamento y rescate. https://www.rpas-drones.com/auxdron-lifeguard-salvamento-rescate/, July 2018. Accessed: 2021-6-22.

[6] Eugene Demaitre. UGV vendors take national security dispute to court. https://www.roboticsbusinessreview.com/public-safety/ugv-vendors-take-national-security-dispute-court/, July 2017. Accessed: 2021-6-22.

[7] Jim Ollhoff. *Search & Rescue*. ABDO & Daughters, 2012.

[8] AUV. https://www.ecagroup.com/en/solutions/auv-search-and. Acceso: 22-6-2021.

[9] Jisoo Park, Jingdao Chen, Yong K Cho, Dae Y Kang, and Byung J Son. Cnn-based person detection using infrared images for night-time intrusion warning systems. *Sensors*, 20(1):34, 2020.

[10] Jan Portmann, Simon Lynen, Margarita Chli, and Roland Siegwart. People detection and tracking from aerial thermal views. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 1794–1800. IEEE, 2014.

[11] Burak Ilikci, Lei Chen, Hyuk Cho, and Qingzhong Liu. Heat-map based emotion and face recognition from thermal images. In *2019 Computing, Communications and IoT Applications (ComComAp)*, pages 449–453. IEEE, 2019.

[12] Achim Königs and Dirk Schulz. Evaluation of thermal imaging for people detection in outdoor scenarios. In *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6. IEEE, 2012.

[13] Mate Krišto, Marina Ivasic-Kos, and Miran Pobar. Thermal object detection in difficult weather conditions using yolo. *IEEE Access*, 2020.

[14] Gianmarco Cerutti, Rahul Prasad, and Elisabetta Farella. Convolutional neural network on embedded platform for people presence detection in low resolution thermal images. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.

[15] Andres Gomez, Francesco Conti, and Luca Benini. Thermal image-based cnn's for ultra-low power people recognition. In *Proceedings of the 15th ACM International Conference on Computing Frontiers*, pages 326–331, 2018.

[16] Gianmarco Cerutti, Bojan Milosevic, and Elisabetta Farella. Outdoor people detection in low resolution thermal images. In *2018 3rd International Conference on Smart and Sustainable Technologies (SpliTech)*, 2018.

[17] Diego M Jiménez-Bravo, Pierre Masala Mutombo, Bart Braem, and Johann M Marquez-Barja. Applying faster r-cnn in extremely low-resolution thermal images for people detection. In *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–4. IEEE, 2020.

[18] Muhammad Ilham Perdana, Anhar Risnumawan, and Indra Adji Sulistijono. Automatic aerial victim detection on low-cost thermal camera using convolutional neural network. In *2020 International Symposium on Community-centric Systems (CcS)*, pages 1–5. IEEE, 2020.

[19] Termografía infrarroja, ùqué es y para que se aplica? - ebuilding. https://ebuilding.es/termografia-infrarroja/, March 2017. Acceso: 1 - 5 -2021.

[20] Rafael Royo Pastor. Termografía infrarroja. fundamentos, investigación y aplicaciones. *Colección Manual de referencia*, 2013.

[21] Wikipedia contributors. Thermography. https://en.wikipedia.org/w/index.php?title =Thermographyoldid=1021245667, May 2021. Acceso: 1 - 5 -2021.

[22] Wikipedia contributors. Reflectividad. https://es.wikipedia.org/w/index.php?title =Reflectividadoldid=129655659. Acceso: 1 - 5 -2021.

[23] http://www.ait-orsenor.com/testo_guia.pdf. Acceso: 1 - 5 -2021.

[24] Tablas de emisividad. http://www.academiatesto.com.ar/cms/tablas-de-emisividad. Acces0: 2021-5-11.

[25] Ventanas atmosféricas. http://www.academiatesto.com.ar/cms/ventanas-atmosfericas. Acceso: 1 - 5 -2021.

[26] Wikipedia contributors. Radiación infrarroja. https://es.wikipedia.org/w/index.php?title=Radiaci%C3%B3n_infrarroja&oldid=135230377. *Acceso* : 1 – 5 – 2021.

[27] Mate Krišto, Marina Ivasic-Kos, and Miran Pobar. Thermal object detection in difficult weather conditions using yolo. *IEEE Access*, 8, 2020.

[28] Fluke. De qué forma las paletas de color, las alarmas y los marcadores mejoran las inspecciones infrarrojas. https://www.fluke.com/es-es/informacion/blog/termografia/de-que-forma-las-paletas-de-color-las-alarmas-y-los-marcadores-mejoran-las-inspecciones-infrarrojas, December 2020. Acceso: 1 - 5 -2021.

[29] Wikipedia contributors. Fotografía. https://es.wikipedia.org/w/index.php?title=Fotograf%C3%ADa&oldid=135375066. Acceso: 1 - 5 -2021.

[30] Andres Gomez, Francesco Conti, and Luca Benini. Thermal image-based cnn's for ultra-low power people recognition. In *Proceedings of the 15th ACM International Conference on Computing Frontiers*, 2018.

[31] Lady Ada. PIR motion sensor. https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor, January 2014. Acceso: 1 - 5 -2021.

[32] Detector de movimiento/presencia infrarrojo pasivo cableado. https://alarmas-para-casa.com.es/inicio/80-detector-de-movimientopresencia-infrarrojo-pasivo-cableado.html?gclid=CjwKCAjwm7mEBhBsEiwA$_of$ – $THFEoSLYvWoqNqxjq9Xeuggs3EGURsIgy6rEJ24wvsU6UO6wpVwS2BoCo$ – $8QAvD_BwE, January 2020$. *Acceso* : 1 – 5 – 2021.

[33] Jairo Rojas Campo. Cámaras térmicas o termográficas: cómo funcionan, tipos y marcas de seguridad. https://www.tecnoseguro.com/analisis/pro/camaras-termicas-como-funcionan-tipos-marcas-seguridad. Acceso: 1 - 5 -2021.

[34] Qué son las redes neuronales y sus funciones. https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/, October 2019. Accessed: 2021-6-17.

[35] Michael A Nielsen. Neural networks and deep learning. 2015.

[36] Deep learning crash course for beginners. https://www.youtube.com/watch?v=VyWAvY2CF9c. Accessed: 2021-6-17.

[37] Michael A Nielsen. Neural networks and deep learning. 2015.

[38] Daniel Lerch. Deep learning con redes pre-entrenadas en ImageNet. https://medium.com/neuron4/redes-pre-entrenadas-en-imagenet-30d858c37b1f, February 2018. Accessed: 2021-6-17.

[39] CS231n convolutional neural networks for visual recognition. https://cs231n.github.io/convolutional-networks/. Accessed: 2021-6-17.

[40] Sumit Saha. A comprehensive guide to convolutional neural networks — the ELI5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, December 2018. Accessed: 2021-6-17.

[41] Ilija Mihajlovic. Everything you ever wanted to know about computer vision. https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e, April 2019. Accessed: 2021-6-17.

[42] Joyce Xu. Deep learning for object detection: A comprehensive review. https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9, September 2017. Accessed: 2021-6-17.

[43] Ahmed Fawzy Gad. Faster R-CNN explained for object detection tasks. https://blog.paperspace.com/faster-r-cnn-explained-object-detection/, November 2020. Accessed: 2021-6-17.

[44] Tomasz Grel. Region of interest pooling explained. https://deepsense.ai/region-of-interest-pooling-explained/, February 2017. Accessed: 2021-6-17.

[45] Redes neuronales de clases múltiples: Softmax. https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax?hl=es. Accessed: 2021-6-17.

[46] Eddie Forson. Understanding SSD MultiBox — real-time object detection in deep learning. https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab, November 2017. Accessed: 2021-6-17.

[47] Jonathan Hui. SSD object detection: Single shot MultiBox detector for real-time processing. https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06, March 2018. Accessed: 2021-6-17.

[48] Uri Almog. YOLO V3 explained - towards data science. https://towardsdatascience.com/yolo-v3-explained-ff5b850390f, October 2020. Accessed: 2021-6-17.

[49] Jonathan Hui. Real-time object detection with YOLO, YOLOv2 and now YOLOv3. https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088, March 2018. Accessed: 2021-6-17.

[50] Welcome to python.org. https://www.python.org/. Accessed: 2021-6-23.

[51] Horacio Quiroga. *Anaconda*. Createspace Independent Publishing Platform, North Charleston, SC, 2017.

[52] Najmeh Foroozani. env_conda. https://github.com/Foroozani/env_conda.

[53] Gerardus Blokdyk. *Tensorflow: A Complete Guide*. 5starcooks, 2018.

[54] PyTorch. https://pytorch.org/. Accessed: 2021-6-23.

[55] Home - OpenCV. https://opencv.org/, February 2021. Accessed: 2021-6-23.

[56] Wikipedia contributors. CUDA. https://es.wikipedia.org/w/index.php?title=CUDAoldid=135154579. Accessed: 2021-6-23.

[57] Tf2 object detection zoo. https://github.com/tensorflow/models/blob/master/research/object_detection, /g3doc/tf2_detection$_zoo.md$.

[58] Armaan Priyadarshan. Training-a-custom-tensorflow-2.x-object-detector. https://github.com/armaanpriyadarshan/Training -a-CustomTensorFlow-2.x-Object-Detector.

[59] Alejandro Puig. deteccion-objetos-video. https://github.com/puigalex/deteccion-objetos-video.

[60] Rafael Padilla. Object-detection-metrics. https://github.com/rafaelpadilla/Object-Detection-Metrics different-competitions-different-metrics.

[61] S a sanchez et al 2020 iop conf. ser.: Mater. sci. eng. 844 012024.

[62] Brundtland H. et. al. Informe brundland. comisiión mundial para el medio ambiente y desarrollo, onu, nueva york, 1987.

11. Anexo I: Planicación y costes

11.1. EDP

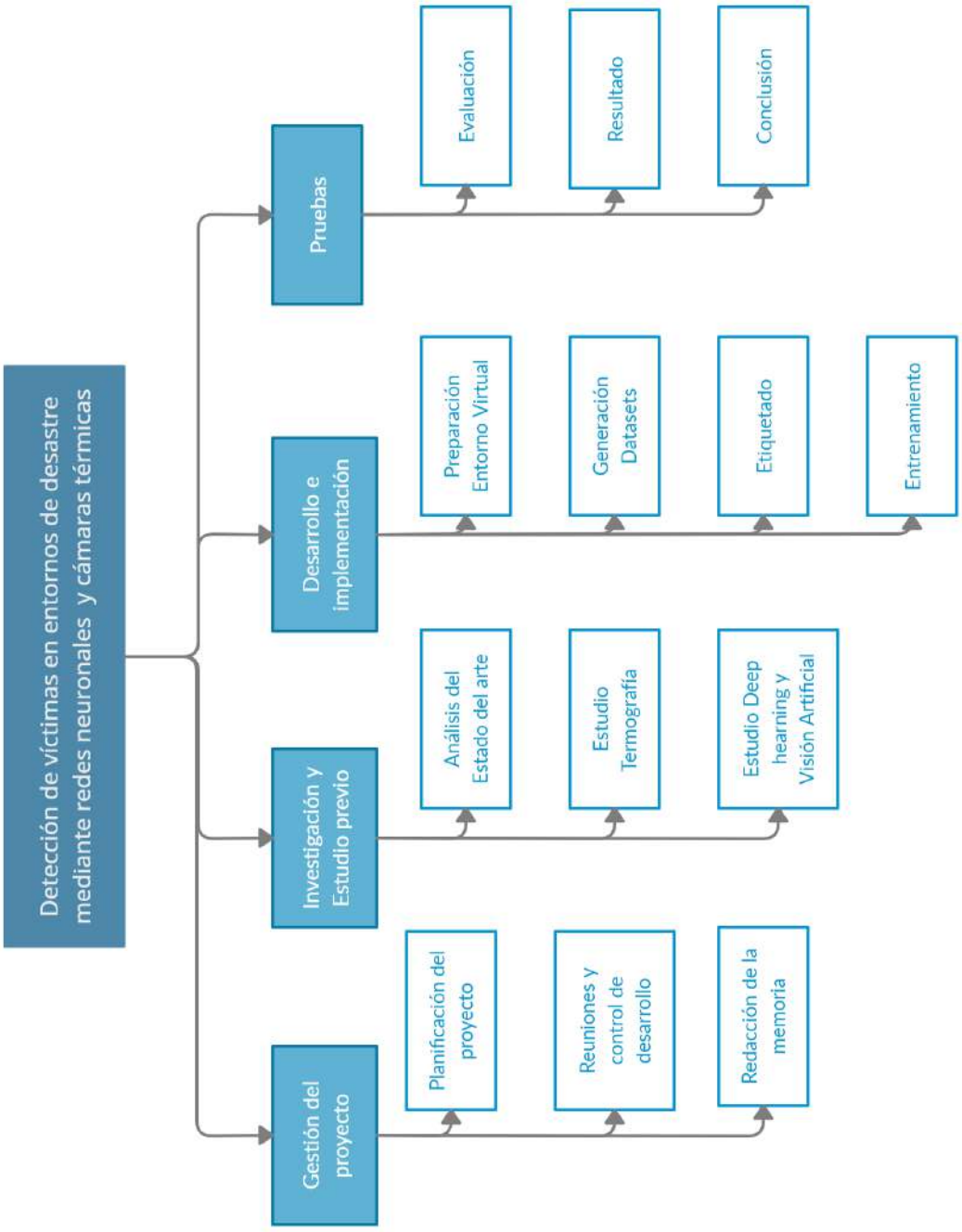


Figura 87: EDP. Fuente:Autor

11.2. Diagrama de Gantt

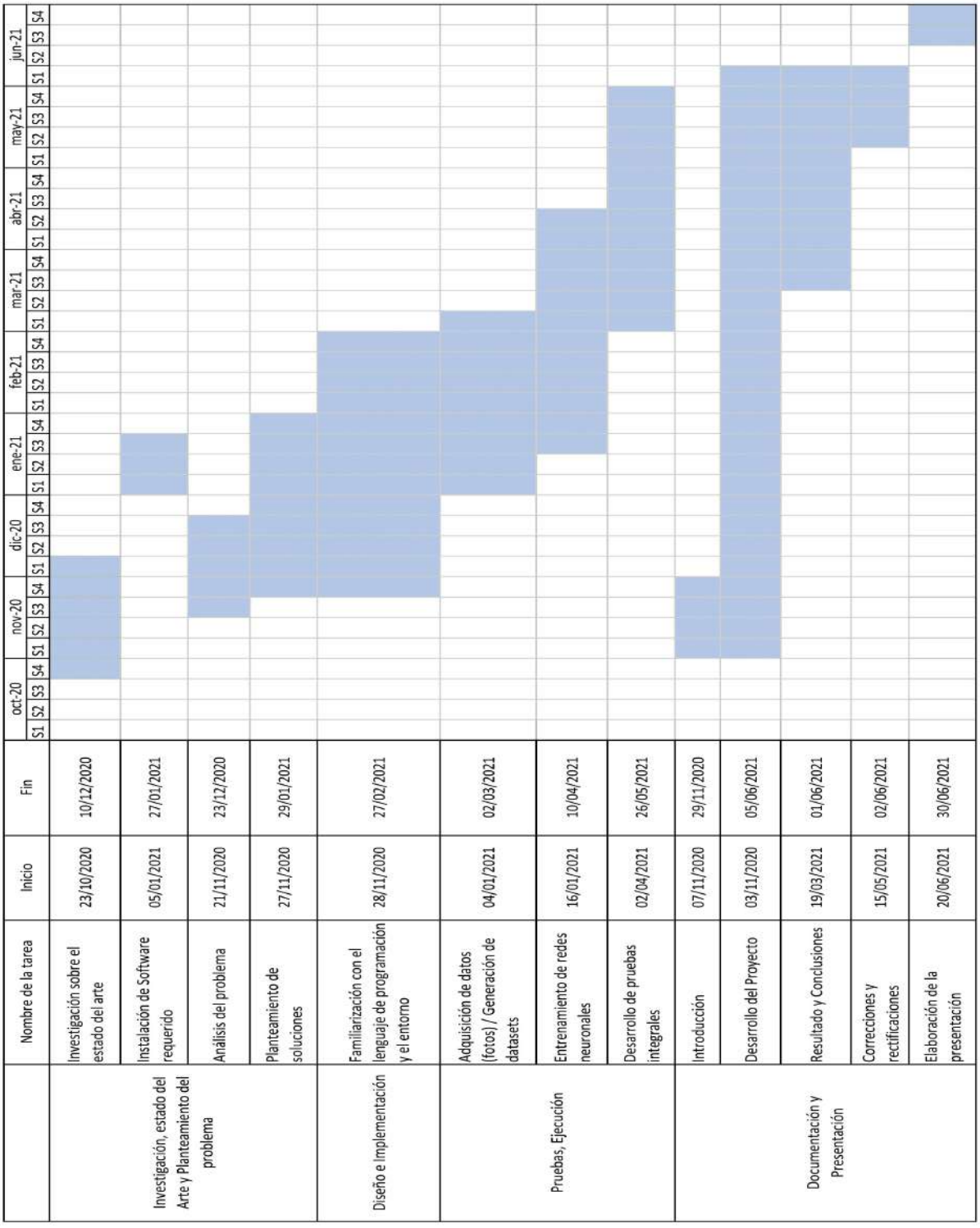


Figura 88: Diagrama de Gannt. Fuente:Autor

11.3. Estudio económico

A continuación se presenta el estudio económico realizada para este trabajo.

Costes personal

El coste personal se obtiene del coste del trabajo realizado por el alumno y del coste del trabajo realizado por el tutor.

El coste del alumno se obtiene a partir de las horas empleadas en este proyecto y del coste por horas de un ingeniero recién egresado. El Trabajo Fin de Grado en el Grado en Ingeniería en Tecnologías Industriales de la UPM son 12 créditos ECTS. Cada crédito equivale a 30 horas de trabajo, luego se tiene una dedicación de 360 horas. El coste anual de un ingeniero recién titulado es de unos 37900€/año. En este coste se incluyen los costes directos e indirectos. Hay 217 días laborales al año, y para la jornada completa de 8 horas da un total de 1736 horas anuales. Así se obtiene un coste por hora de 21,83 €/h. Multiplicando el coste por horas por las horas empleadas en el proyecto se obtiene un coste personal de 7858,80€.

El coste del tutor se obtiene a partir del coste anual que supone el tutor a la Universidad, siendo de unos 40000€/año, dando un valor de 23,04€/hora; y que se ha dedicado unos 30 minutos por semana a la revisión del proyecto durante un total de 33 semanas, dando un total de 16,5 horas. Multiplicando estos valores se obtiene un coste del tutor de 380,18€.

Costes del software

El mayoría del software empleado en este proyecto es software libre, luego no genera ningún coste en el proyecto.

El sistema operativo Windows 10 tiene un precio de 145€ al año y el coste de Microsoft Office 365 es de 69€ al año.

Costes de amortización del material

En cuanto al coste material, los materiales usados para este proyecto fueron la cámara infrarroja y un ordenador de sobremesa.

La cámara tiene un coste de 2299€ y suponiendo un periodo de vida útil estimado de 10 años, al aplicar el coeficiente de amortización lineal se obtiene que el coste de amortización del mismo en un año es de 229,90€.

Para el caso del ordenador, con un coste de 6400€ y una vida útil de 5 años, al aplicar el coeficiente de amortización lineal se obtiene que el coste de amortización del mismo en un año es de 1280€.

De esta manera se obtiene el presupuesto global del proyecto.

Tabla 4: Presupuesto global del proyecto

Partida	**Concepto**	**Gasto**
Personal	Coste del trabajo del alumno	7858,80€
	Coste del trabajo del tutor	380,18€
	Total Personal	**8238,98€**
Software	Sistema operativo Windows 10	145€
	Microsoft Office 365	69€
	Total Software	**214€**
Amortización	Cámara infrarroja	229,90€
	Ordenador	1280€
	Total Amortización	**1509,90€**
Total	**TOTAL PROYECTO**	**9962,88€**

12. Anexo II:Figuras y tablas adicionales

12.1. Figuras

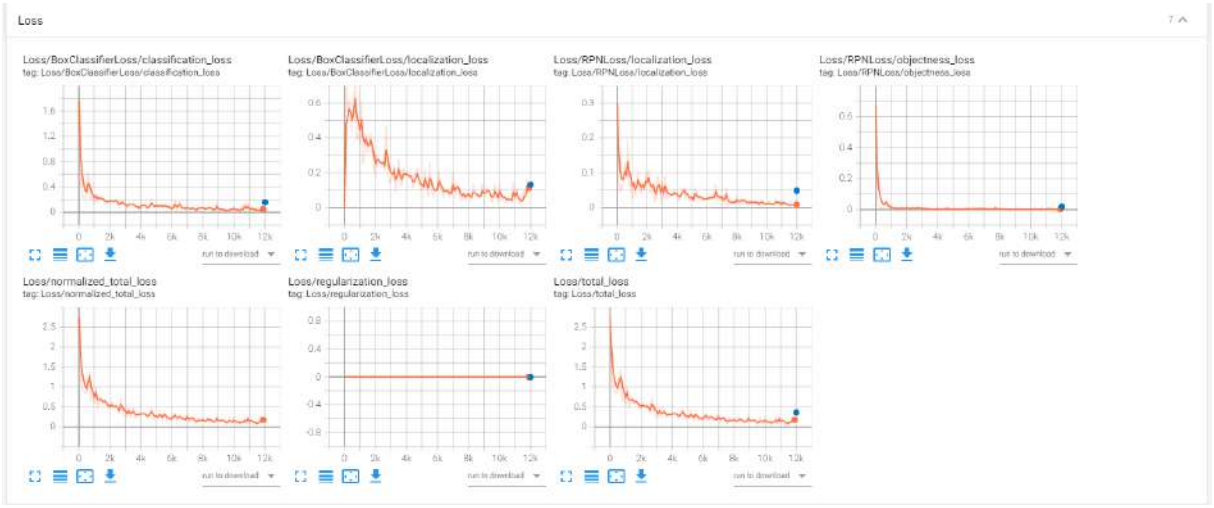


Figura 89: Resultado de loss entrenamiento Faster R-CNN noche. Fuente: Autor

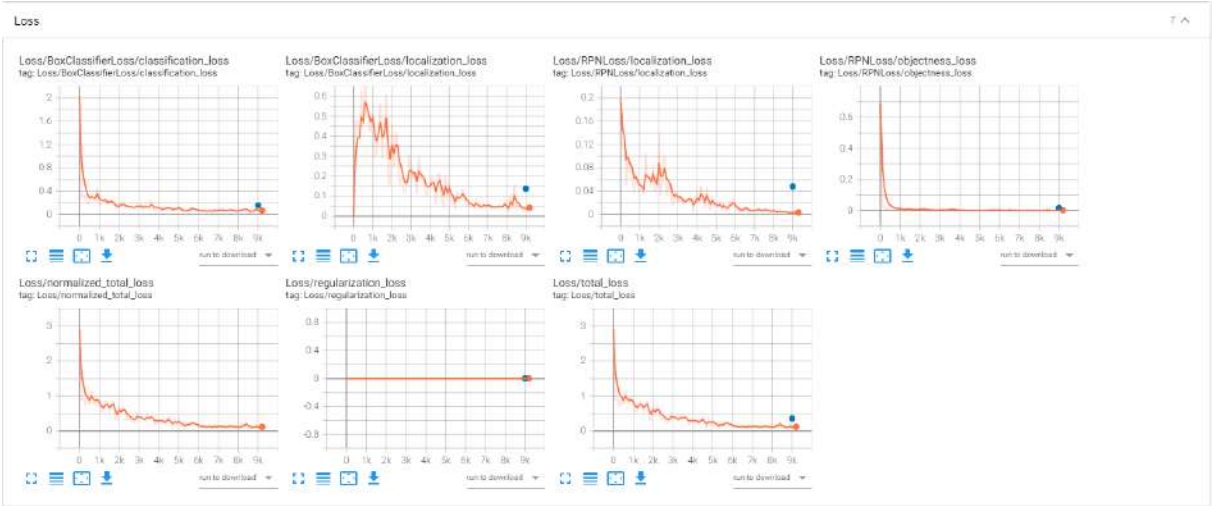


Figura 90: Resultado de loss entrenamiento Faster R-CNN día. Fuente: Autor

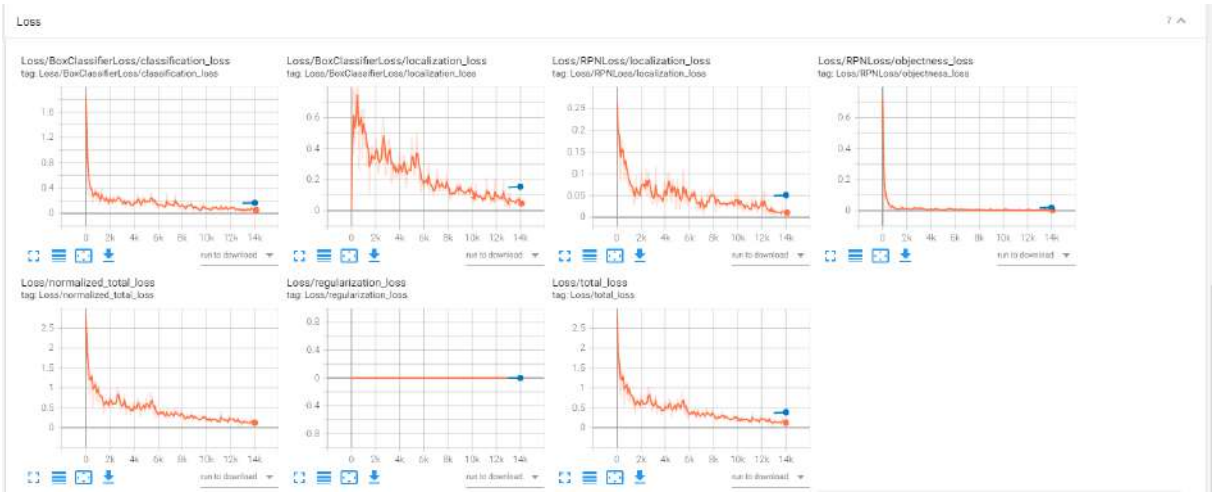


Figura 91: Resultado de loss entrenamiento Faster R-CNN combinado. Fuente: Autor

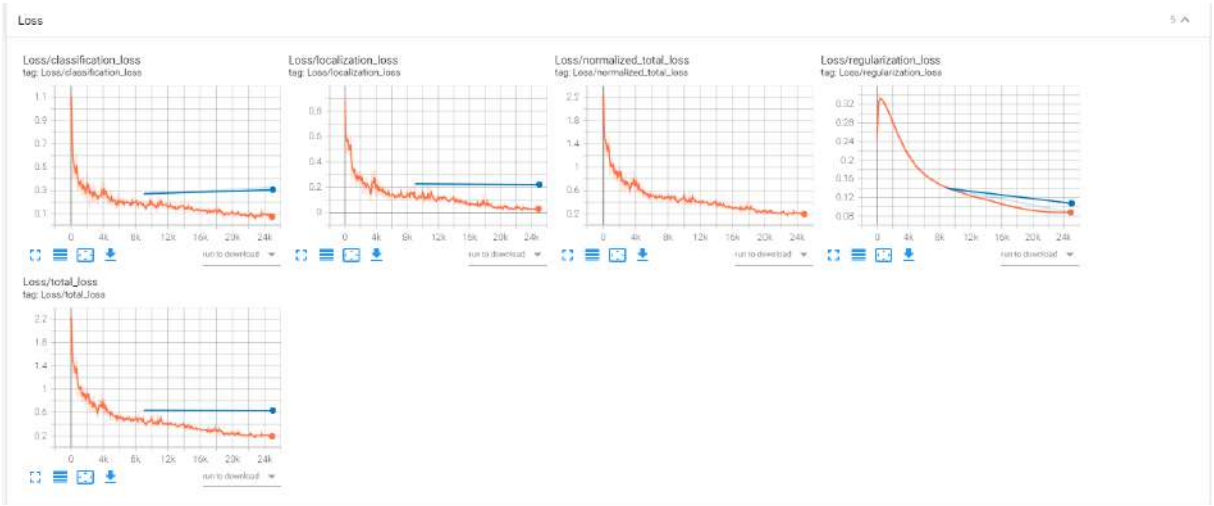


Figura 92: Resultado de loss entrenamiento SSD noche. Fuente: Autor

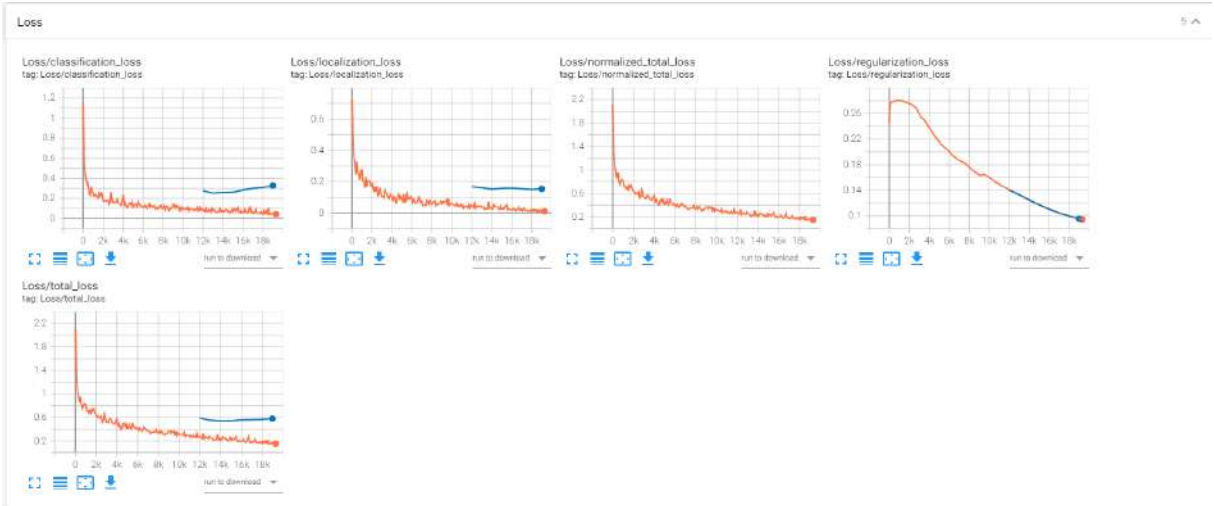


Figura 93: Resultado de loss entrenamiento SSD noche. Fuente: Autor

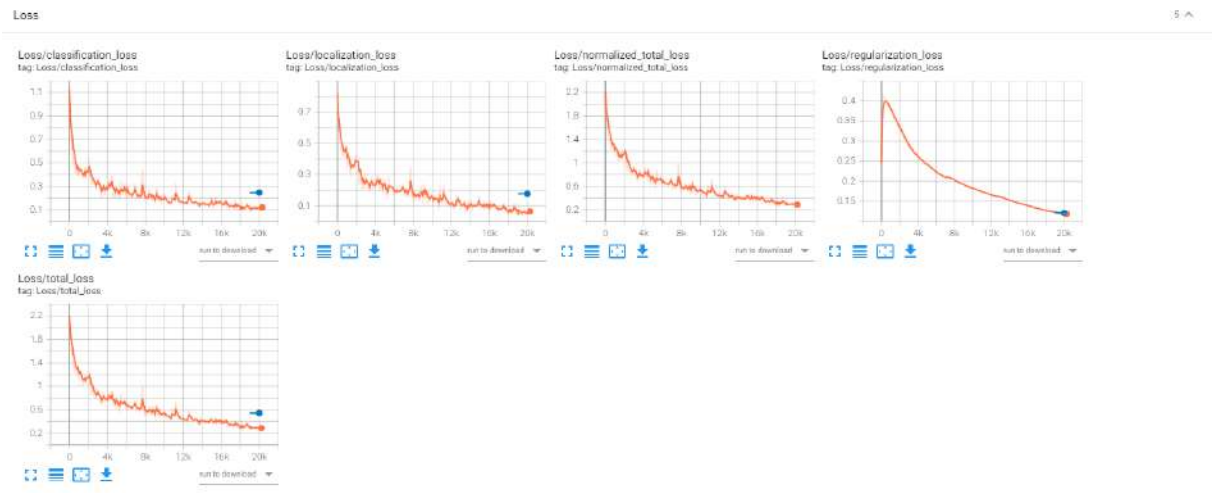
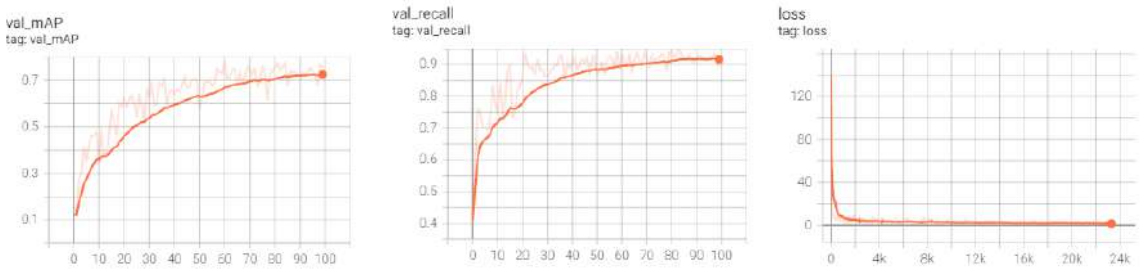
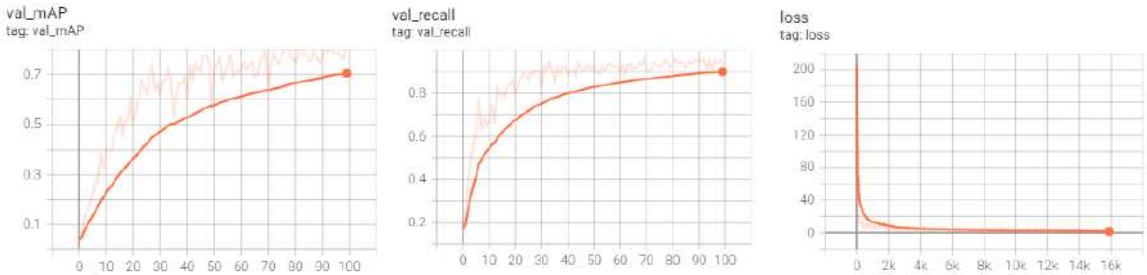


Figura 94: Resultado de loss entrenamiento SSD noche. Fuente: Autor



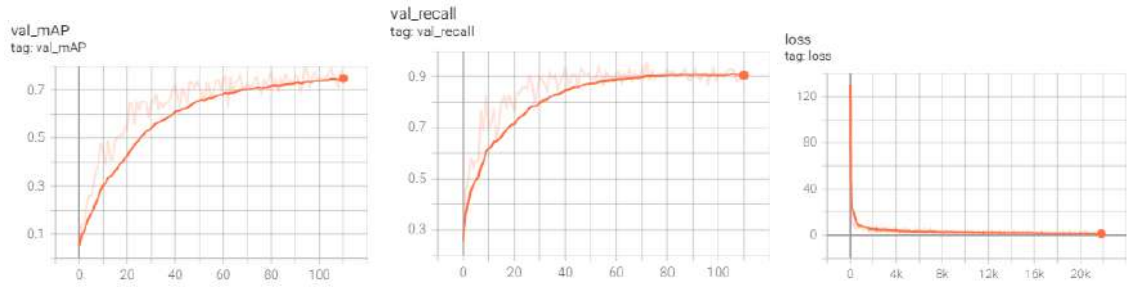
(a) mAP (b) Recall (c) loss

Figura 95: Resultados del entrenamiento y evaluación YOLOv3 noche. Fuente: Autor



(a) mAP (b) Recall (c) loss

Figura 96: Resultados del entrenamiento y evaluación YOLOv3 día. Fuente: Autor



(a) mAP (b) Recall (c) loss

Figura 97: Resultados del entrenamiento y evaluación YOLOv3 combinado. Fuente: Autor

12.2. Tablas

Resultados de la evaluación de las tres redes con cada dataset

Resultados Faster R-CNN

Tabla 5: Resultados de la evaluación Faster R-CNN dataset noche

Average Prec.	IoU=0.50:0.95	area=all	maxDets=100	0.451
Average Prec.	IoU=0.50	area=all	maxDets=100	0.874
Average Prec.	IoU=0.75	area=all	maxDets=100	0.391
Average Prec.	IoU=0.50:0.95	area=small	maxDets=100	0.000
Average Prec.	IoU=0.50:0.95	area=medium	maxDets=100	0.380
Average Prec.	IoU=0.50:0.95	area=large	maxDets=100	0.486
Average Recall	IoU=0.50:0.95	area=all	maxDets=1	0.466
Average Recall	IoU=0.50:0.95	area=all	maxDets=10	0.554
Average Recall	IoU=0.50:0.95	area=all	maxDets=100	0.562
Average Recall	IoU=0.50:0.95	area=small	maxDets=100	0.000
Average Recall	IoU=0.50:0.95	area=medium	maxDets=100	0.522
Average Recall	IoU=0.50:0.95	area=large	maxDets=100	0.581
Loss	RPN	Localization		0.048
Loss	RPN	Objectness		0.019
Loss	BoxClassifier	Localization		0.132
Loss	BoxClassifier	Classification		0.158
Loss	Regularization			0.000
Loss	Total			0.357

Tabla 6: Resultados de la evaluación Faster R-CNN dataset dia

Average Prec.	IoU=0.50:0.95	area=all	maxDets=100	0.538
Average Prec.	IoU=0.50	area=all	maxDets=100	0.935
Average Prec.	IoU=0.75	area=all	maxDets=100	0.558
Average Prec.	IoU=0.50:0.95	area=small	maxDets=100	0.200
Average Prec.	IoU=0.50:0.95	area=medium	maxDets=100	0.415
Average Prec.	IoU=0.50:0.95	area=large	maxDets=100	0.571
Average Recall	IoU=0.50:0.95	area=all	maxDets=1	0.496
Average Recall	IoU=0.50:0.95	area=all	maxDets=10	0.606
Average Recall	IoU=0.50:0.95	area=all	maxDets=100	0.616
Average Recall	IoU=0.50:0.95	area=small	maxDets=100	0.200
Average Recall	IoU=0.50:0.95	area=medium	maxDets=100	0.487
Average Recall	IoU=0.50:0.95	area=large	maxDets=100	0.648
Loss	RPN	Localization		0.048
Loss	RPN	Objectness		0.016
Loss	BoxClassifier	Localization		0.137
Loss	BoxClassifier	Classification		0.157
Loss	Regularization			0.000
Loss	Total			0.359

Tabla 7: Resultados de la evaluación Faster R-CNN dataset combinado

Average Prec.	IoU=0.50:0.95	area=all	maxDets=100	0.487
Average Prec.	IoU=0.50	area=all	maxDets=100	0.902
Average Prec.	IoU=0.75	area=all	maxDets=100	0.467
Average Prec.	IoU=0.50:0.95	area=small	maxDets=100	0.000
Average Prec.	IoU=0.50:0.95	area=medium	maxDets=100	0.350
Average Prec.	IoU=0.50:0.95	area=large	maxDets=100	0.524
Average Recall	IoU=0.50:0.95	area=all	maxDets=1	0.480
Average Recall	IoU=0.50:0.95	area=all	maxDets=10	0.582
Average Recall	IoU=0.50:0.95	area=all	maxDets=100	0.592
Average Recall	IoU=0.50:0.95	area=small	maxDets=100	0.000
Average Recall	IoU=0.50:0.95	area=medium	maxDets=100	0.511
Average Recall	IoU=0.50:0.95	area=large	maxDets=100	0.612
Loss	RPN	Localization		0.050
Loss	RPN	Objectness		0.017
Loss	BoxClassifier	Localization		0.150
Loss	BoxClassifier	Classification		0.159
Loss	Regularization			0.000
Loss	Total			0.376

Resultados SSD

Tabla 8: Resultados de la evaluación SSD dataset noche

Average Prec.	IoU=0.50:0.95	area=all	maxDets=100	0.417
Average Prec.	IoU=0.50	area=all	maxDets=100	0.81
Average Prec.	IoU=0.75	area=all	maxDets=100	0.367
Average Prec.	IoU=0.50:0.95	area=small	maxDets=100	0.000
Average Prec.	IoU=0.50:0.95	area=medium	maxDets=100	0.373
Average Prec.	IoU=0.50:0.95	area=large	maxDets=100	0.441
Average Recall	IoU=0.50:0.95	area=all	maxDets=1	0.459
Average Recall	IoU=0.50:0.95	area=all	maxDets=10	0.567
Average Recall	IoU=0.50:0.95	area=all	maxDets=100	0.575
Average Recall	IoU=0.50:0.95	area=small	maxDets=100	0.000
Average Recall	IoU=0.50:0.95	area=medium	maxDets=100	0.560
Average Recall	IoU=0.50:0.95	area=large	maxDets=100	0.580
Loss	Localization			0.217
Loss	Classification			0.327
Loss	Regularization			0.088
Loss	Total			0.631

Tabla 9: Resultados de la evaluación SSD dataset día

Average Prec.	IoU=0.50:0.95	area=all	maxDets=100	0.524
Average Prec.	IoU=0.50	area=all	maxDets=100	0.924
Average Prec.	IoU=0.75	area=all	maxDets=100	0.528
Average Prec.	IoU=0.50:0.95	area=small	maxDets=100	0.000
Average Prec.	IoU=0.50:0.95	area=medium	maxDets=100	0.458
Average Prec.	IoU=0.50:0.95	area=large	maxDets=100	0.550
Average Recall	IoU=0.50:0.95	area=all	maxDets=1	0.498
Average Recall	IoU=0.50:0.95	area=all	maxDets=10	0.617
Average Recall	IoU=0.50:0.95	area=all	maxDets=100	0.627
Average Recall	IoU=0.50:0.95	area=small	maxDets=100	0.000
Average Recall	IoU=0.50:0.95	area=medium	maxDets=100	0.559
Average Recall	IoU=0.50:0.95	area=large	maxDets=100	0.651
Loss	Localization			0.157
Loss	Classification			0.264
Loss	Regularization			0.117
Loss	Total			0.538

Tabla 10: Resultados de la evaluación SSD dataset combinado

Average Prec.	IoU=0.50:0.95	area=all	maxDets=100	0.466
Average Prec.	IoU=0.50	area=all	maxDets=100	0.886
Average Prec.	IoU=0.75	area=all	maxDets=100	0.426
Average Prec.	IoU=0.50:0.95	area=small	maxDets=100	0.202
Average Prec.	IoU=0.50:0.95	area=medium	maxDets=100	0.376
Average Prec.	IoU=0.50:0.95	area=large	maxDets=100	0.494
Average Recall	IoU=0.50:0.95	area=all	maxDets=1	0.468
Average Recall	IoU=0.50:0.95	area=all	maxDets=10	0.591
Average Recall	IoU=0.50:0.95	area=all	maxDets=100	0.605
Average Recall	IoU=0.50:0.95	area=small	maxDets=100	0.200
Average Recall	IoU=0.50:0.95	area=medium	maxDets=100	0.550
Average Recall	IoU=0.50:0.95	area=large	maxDets=100	0.624
Loss	Localization			0.178
Loss	Classification			0.246
Loss	Regularization			0.123
Loss	Total			0.546

Resultados YOLOv3

Tabla 11: Resultados de la evaluación YOLOv3 dataset noche

mAP		0.776
Average Prec.	Class	0.958
Average Conf.		0.878
Average Recall		0.931
Loss	Classification	0.038
Loss	Confidence	0.395
Loss	Position	0.098
Loss	Total	0.531

Tabla 12: Evaluación de precisión media de clases YOLO dataset noche

persona	0.942
cabeza	0.790
brazo	0.567
pierna	0.939
torso	0.640

Tabla 13: Resultados de la evaluación YOLOv3 dataset día

mAP		0.871
Average Prec.	Class	0.944
Average Conf.		0.921
Average Recall		0.954
Loss	Classification	0.022
Loss	Confidence	0.125
Loss	Position	0.135
Loss	Total	0.282

Tabla 14: Evaluación de precisión media de classes YOLO dataset día

persona	0.843
cabeza	0.724
brazo	0.666
pierna	0.922
torso	0.693

Tabla 15: Resultados de la evaluación YOLOv3 dataset combinado

mAP		0.809
Average Prec.	Class	0.958
Average Conf.		0.929
Average Recall		0.910
Loss	Classification	0.049
Loss	Confidence	0.286
Loss	Position	0.151
Loss	Total	0.486

Tabla 16: Evaluación de precisión media de classes YOLO dataset combinado

persona	0.918
cabeza	0.931
brazo	0.449
pierna	0.988
torso	0.548

13. Anexo III:Aplicación a la detección de víctimas en tiempo real

Una de las principales aportaciones de este trabajo destaca en la aplicabilidad del modelo previamente entrenado a la detección de víctimas, usando diferentes medios para la adquisición de imágenes. El mejor modelo obtenido se ha validado utilizando un robot en campo (Unitree A1) con capacidad de transmitir imágenes hacia una estación remota (E.R.O - Estación Remota del Operador).

El robot en esta fase de experimentación del sistema es tele-operado desde una estación remota por un operador o rescatista previamente capacitado.

La transmisión de imágenes y el procesamiento se realizan en tiempo real en la E.R.O, de tal manera que el operador puede saber a priori si existe una víctima (pudiendo ser una persona totalmente inconsciente o atrapada entre escombros) que necesite de asistencia primaria y su ubicación dentro del entorno en base a la posición del robot.

La arquitectura de comunicaciones con el robot en campo y la adquisición de imágenes ha sido previamente desarrollada como parte del proyecto TASAR del ROBCIB mediante ROS (Robot Operating System), la red entrenada toma las imágenes y las procesa para evaluar si existe una víctima.

El Robot utilizado para la exploración en el entorno de desastre ha sido el Unitree A1, dotado de la cámara térmica y conectado a una red inalámbrica para la transmisión de datos. En esta red están conectados los siguientes ordenadores:

- El ordenador a bordo del Robot Unitree A1, una Jetson TX2.
- El ordenador de la E.R.O, una MSI con procesador Intel i7 10ma y Tarjeta grafica Nvidia GTX 1660Ti

El control de movimientos del robot en el entorno se ejecuta mediante la publicación de comandos de velocidad lineal y angular, mediante el tópico /cmd_vel. Por su parte, el robot retransmite su posición dentro del entorno mediante el tópico /position. El flujo de estos datos se muestra en la figura 98. La estructura de los tópicos publicados es:

- Transmisión de comandos de velocidad al robot:

/cmd_vel (geometry_msgs/Twist)

linear:
x: 0.1 (vel_lin)
y: 0.0
z: 0.0

angular:
x: 0.0
y: 0.0
z: 0.1 (vel_ang)

- Recepción de posición del robot:

/pose (geometry_msgs/Twist)

geometry_msgs/Vector3 linear
float64 x (Posición x)
float64 y (Posición y)
float64 z

geometry_msgs/Vector3 angular float64 x
float64 y
float64 z (Orientación)

Para la captación y posterior transmisión de imágenes térmicas desde la cámara integrada en el robot, se han instalado los drivers para ROS (en la Jetso TX2), desarrollados por el fabricante Optris. Estos drivers están disponibles en la página http://wiki.ros.org/optris_drivers.

La imagen se transmite mediante el tópico /image_compressed. Se ha utilizado la imagen comprimida, debido a que incrementa la tasa de transmisión de fps y la pérdida de información no es significativa para el desarrollo del proceso.

La estructura del tópico de la imagen es:

- Recepción de imagen térmica:

/image_compressed (sensor_msgs/Image Message)

uint32 height uint32 width
string encoding (Contiene los píxeles y canales de la imagen)
uint8 is_bigendian
uint32 step
uint8[] data

Para el procesamiento en tiempo real de la imagen recibida en la E.R.O, se ha instalado previamente en el entorno virtual de anaconda los paquetes de ROS, de tal manera que, con la red ya entrenada, se pueda realizar la inferencia y detección de personas dentro en la imagen recibida. Esto se puede ver en la figura 98.

Para ello se ha desarrollado un algoritmo que recibe como entrada la imagen transmitida por el robot, a través de la suscripción al tópico correspondiente de ROS y se procesa mediante OpenCV para aplicar las respectivas correcciones a la imagen, operaciones de erosión, dilatación y eliminación de ruido. Finalmente se realiza la inferencia con la red entrenada y se establece si existe o no una víctima en la zona que inspecciona el robot. Este proceso se ejecuta de manera cíclica durante toda la inspección del entorno.

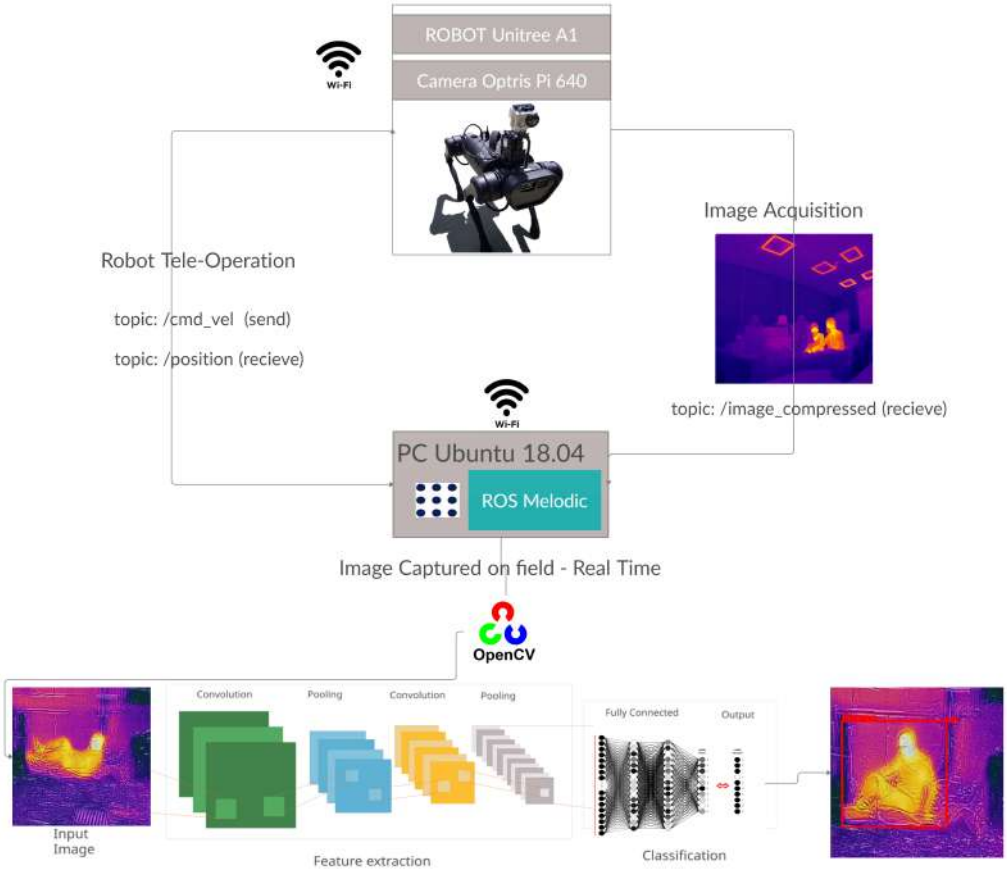


Figura 98: Esquema de conexión del sistema para para el procesamiento en tiempo real mediante un robot en el entorno SAR. Fuente: TASAR-ROBCIB

14. Anexo IV:Índice figuras y tablas

Índice de figuras

15. Anexo V: Glorario: siglas y abreviaturas

LED	Light Emitting Diode
NIR	Near-Infrared
SWIR	Short-Wavelength Infrared
MWIR	Mid-Wavelength Infrared
LWIR	Long-Wavelength Infrared
FIR	Far Infrared
RGB	Reg - Green - Blue
PIR	Passive InfraRed sensors
MNIST	Modified National Institute of Standards and Technology
ReLU	Rectified Linear Unit
NN	Neural Network
CNN	Convolutional Neural Network
R-CNN	Region-based Convolutional Neural Network
RoI	Region of Interest
RPN	Region Proposal Network
IoU	Intersection-over-Union
SSD	Single Shot MultiBox Detector
nms	non-maximum suppression
YOLO	You Only Look Once
CUDA	Compute Unified Device Architecture
CPU	Central Processing Unit
GPU	Graphics processing unit
API	Application Programming Interface
COCO	Common Objects in Context
TP	True Positive
TN	True Negative
FP	Flase Positive
FN	False Negative
mAP	mean average precision
fps	frames per second
ROS	Robot Operating System
E.R.O	Estación Remota del Operador