UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Implementación de un sistema de detección de objetos dual en el espectro visible y en el infrarrojo

Luis Hernández Casado

2021

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN TRABAJO FIN DE GRADO

Título: Implementación de un sistema de detección de objetos dual en el espectro visible y en el infrarrojo.

Autor: D. Luis Hernández Casado

Tutor: D. Juan Antonio Rodrigo Ferrán

Ponente: D. José Manuel Menéndez García

Departamento: Señales Sistemas y Radiocomunicaciones (SSR)

MIEMBROS DEL TRIBUNAL

Presidente: D. ……

Vocal: D. ……..

Secretario: D. ……..

Suplente: D. ……..

Los miembros del tribunal arriba nombrados acuerdan otorgar la calificación de: …….

Madrid, a de de 20…

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Implementación de un sistema de detección de objetos dual en el espectro visible y en el infrarrojo

LUIS HERNÁNDEZ CASADO

2021

RESUMEN

Actualmente los sistemas de detección y reconocimiento de imágenes automatizados están generalmente basados en el tratamiento de imágenes obtenidas en el espectro visible. Sin embargo, en determinadas ocasiones, especialmente en situaciones de baja iluminación o visibilidad, es imposible obtener imágenes con la suficiente calidad en el espectro visible. En estos casos suele ser posible conseguir imágenes basadas en el espectro infrarrojo. Desafortunadamente los sistemas más habituales de reconocimiento de imágenes fallan al reconocer los diferentes objetos en las imágenes cuando estas son infrarrojas ya que han sido únicamente entrenados con imágenes del espectro visible.

Por otro lado, las bases de datos de imágenes disponibles están, en su mayor parte, compuestas de imágenes del espectro visible y es difícil conseguir bases de datos de imágenes infrarrojas. Esto dificulta entrenar nuevos sistemas para reconocer este tipo de imágenes.

Este trabajo tiene como objetivo resolver este problema, desarrollando un sistema dual, capaz de realizar detección y reconocimiento en las imágenes infrarrojas y al mismo tiempo creando un procedimiento para automatizar la generación de bases de datos de imágenes térmicas listas para usar como conjuntos de entrenamiento de nuevos sistemas.

Para ello, se estudiaron diferentes redes neuronales y métodos de adquisición, detección y tratamiento de imágenes, seleccionando el más conveniente para usar en nuestro sistema.

A continuación, utilizando un dispositivo capaz de captar las imágenes en ambos espectros a la vez, se generó un conjunto doble de imágenes tanto en el espectro visible como en el infrarrojo. Sobre el conjunto de imágenes de espectro visible se aplicaron las técnicas de reconocimiento del sistema elegido, detectando e identificando los objetos de la imagen, así como sus posiciones dentro de la misma.

Los datos obtenidos de cada imagen se utilizaron para el tratamiento de la misma imagen en el espectro infrarrojo, generando automáticamente un Conjunto de Datos de entrenamiento que nos permitió conseguir el objetivo planteado en el proyecto, un nuevo sistema capaz de reconocer, con un alto grado de fiabilidad, los objetos contenidos en las imágenes infrarrojas.

SUMMARY

Currently, automated image detection and recognition systems are generally based on the treatment of images obtained in the visible spectrum. However, on many occasions as with low light or low visibility conditions, it is not possible to get images with enough quality in the visible spectrum, but it is easy to get infrared images. Unfortunately, the most common image recognition systems available right now fail when trying to detect and recognize objects in infrared images because they have been trained exclusively with visible spectrum images.

On the other hand, the available image databases are, for the most part, composed of images of the visible spectrum and it is difficult to obtain databases of infrared images. This makes it difficult to train new systems to recognize these types of images

This work aims to solve this problem, developing a dual system, capable of performing detection and recognition on infrared images and at the same time creating a procedure to automate the generation of ready-to-use thermal imaging databases as training sets of new systems.

To do this, different neural networks and methods of acquisition, detection, and processing of images were studied, selecting the most convenient to use in our system.

Then, using a device capable of capturing the images in both spectra at once, a double set of images was generated in both the visible and infrared spectrums. On the subset of visible spectrum images, the

recognition techniques of the chosen system were applied, detecting, and identifying the objects of the image, as well as their positions within it.

The data obtained from each image were used for the treatment of the corresponding image in the infrared spectrum, automatically generating a Training Data Set that allowed us to achieve the objective set out in the project, a new system capable of recognizing, with a high degree of reliability, the objects contained in the infrared images.

PALABRAS CLAVE

Redes neuronales, reconocimiento de imágenes, imágenes infrarrojas, redes convolucionales, CNN, conjunto de datos, espectro visible, espectro infrarrojo, generación automática de conjuntos de datos, Detectron2.

KEYWORDS

Neural networks, image recognition, infrared images, convolutional networks, CNN, Dataset, visible spectrum, infrared spectrum, automatic dataset generation, Detectron2.

ÍNDICE DEL CONTENIDO

1. INTRODUCCIÓN Y OBJETIVOS

1.1. INTRODUCCIÓN

Los sistemas de detección y reconocimiento de imágenes automatizados están generalmente basados en el tratamiento de imágenes obtenidas en el espectro visible. Sin embargo, en determinadas ocasiones, especialmente en situaciones de baja iluminación o visibilidad, es imposible obtener imágenes con la suficiente calidad en el espectro visible.

Por otro lado, incluso en casos que es posible obtener imágenes en el espectro visible, el contraste de la iluminación es tal que muchos detalles se pierden en las zonas oscuras de las imágenes y a veces ni siquiera el ojo humano, y tampoco los sistemas automatizados de reconocimiento de imágenes, son capaces de reconocer determinados objetos en las mismas. En estos casos suele ser posible conseguir imágenes basadas en el espectro infrarrojo que aportan información sobre los objetos en la imagen, información que el espectro visible es incapaz de aportar.

Desafortunadamente los sistemas más habituales de reconocimiento de imágenes fallan al reconocer los diferentes objetos en las imágenes cuando estas son infrarrojas ya que el entrenamiento de estos sistemas normalmente se ha realizado únicamente con imágenes del espectro visible.

Un ejemplo de esto puede verse en la siguiente figura:

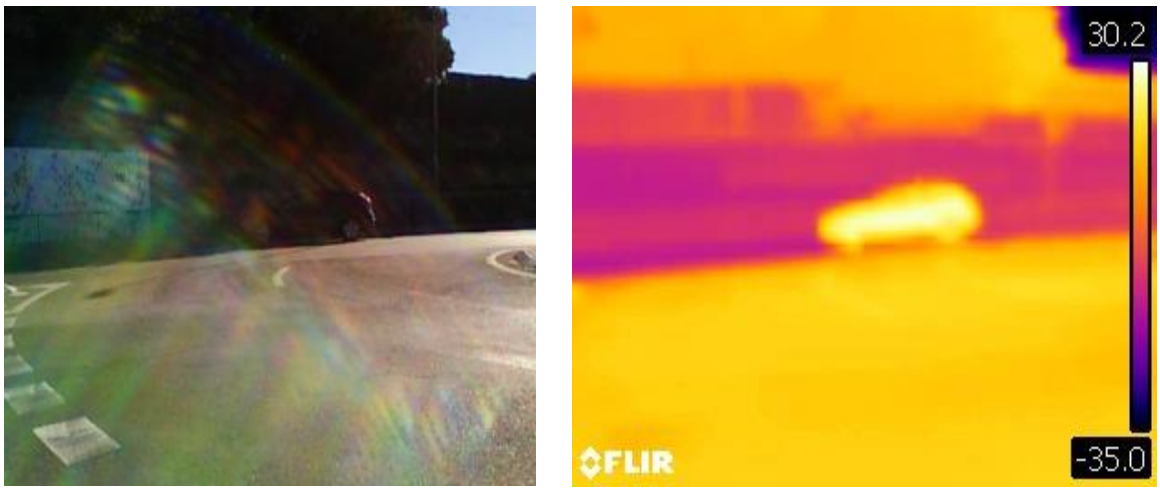

Figura 1: Misma imagen en ambos espectros

Ambas imágenes han sido obtenidas simultáneamente de la misma escena, en la primera imagen visualmente se aprecia en el centro una sombra que podría ser un vehículo, sin embargo, al pasar la imagen por un sistema actual de reconocimiento de imágenes basado en redes neuronales, éste no es capaz de detectar ningún objeto ya que la información contenida en la imagen visual es insuficiente.

La segunda imagen corresponde a la misma situación sólo que representando la información que obtenemos del espectro infrarrojo de la imagen. En este caso visualmente se puede apreciar la forma de un vehículo. Sin embargo, los sistemas tradicionales de reconocimiento de imágenes no son capaces de reconocerlo ya que no están entrenados para reconocer imágenes infrarrojas.

Para empezar el trabajo se estudiarán diferentes redes neuronales y métodos de adquisición, detección y tratamiento de imágenes, seleccionando el más conveniente para usar en nuestro sistema. Lo que se pretende es escoger, entre las tecnologías disponibles, un sistema que permita automatizar los procesos de reconocimiento de imágenes y que estos puedan ser usados con fiabilidad con imágenes infrarrojas.

A continuación, utilizando una cámara fotográfica capaz de captar las imágenes en ambos espectros a la vez, se conseguirá un conjunto doble de imágenes tanto en el espectro visible como en el infrarrojo. Sobre el conjunto de imágenes de espectro visible se aplicarán las técnicas de reconocimiento del

sistema elegido, detectando e identificando los objetos de la imagen, así como sus posiciones dentro de la misma.

Los datos obtenidos de cada imagen se utilizarán para el tratamiento de la misma imagen en el espectro infrarrojo. Nuestro objetivo es la generación automática de un nuevo conjunto de datos que será usado para entrenar un nuevo sistema capaz de reconocer, con un alto grado de fiabilidad, los objetos contenidos en las imágenes infrarrojas.

1.2. OBJETIVOS

Este trabajo tiene como objetivo resolver este problema, desarrollando un sistema dual, capaz de realizar detección y reconocimiento en las imágenes infrarrojas y al mismo tiempo creando un procedimiento para automatizar la generación de bases de datos de imágenes térmicas listas para usar como conjuntos de entrenamiento de nuevos sistemas.

Para ello se han definido los siguientes subobjetivos:

1.- Preparación de un entorno informático hardware y software, instalando un entorno de trabajo que incluye tanto librerías como módulos y utilidades, en las versiones adecuadas para evitar incompatibilidades.

2.- Generación de un amplio conjunto de imágenes, en formatos visual e infrarrojo. Este conjunto de datos, en bruto, será organizado y estructurado para disponer de conjuntos diferenciados que permitan desarrollar y probar el nuevo sistema de reconocimiento infrarrojo.

3.- Creación de un proceso para la generación automática de los conjuntos de datos (datasets) anotados necesarios. Estos nuevos datasets incluyen imágenes infrarrojas, así como la información correspondiente a los objetos contenidos en dichas imágenes.

4.- Procesado de los conjuntos de datos seleccionados para entrenar un nuevo sistema de reconocimiento, válido para imágenes infrarrojas.

5.- Prueba del nuevo sistema de reconocimiento infrarrojo.

1.3. ESTRUCTURA

El desarrollo de este Trabajo de Fin de Grado está estructurado en cinco capítulos diferenciados.

En el primero se realiza la introducción, se comentan los objetivos y el fin que se busca con su realización.

En el segundo capítulo se expone el estado del arte, poniendo en contexto la teoría necesaria referente a este proyecto. Se explican las bases teóricas de la Inteligencia Artificial (IA) y del *Machine Learning*, así como conceptos básicos sobre las imágenes infrarrojas y las herramientas utilizadas durante el desarrollo.

La tercera sección está dedicada al desarrollo del proyecto, y describe todos los procesos realizados para completar este TFG, desde la adquisición de imágenes mediante una cámara dual en los espectros

visible e infrarrojo, hasta los métodos para conseguir un conjunto de datos estructurados que se usan como base para el entrenamiento de un nuevo sistema de reconocimiento especializado para su uso sobre imágenes térmicas.

El cuarto capítulo se centra en el análisis de los resultados obtenidos.

El quinto capítulo se dedica a la exposición de las conclusiones que se extraen de la realización de este proyecto, así como a las potenciales líneas de desarrollo futuras tomando como base el mismo.

Adicionalmente se han incluido un apartado dedicado a la Bibliografía empleada en la elaboración del proyecto, así como dos Anexos, uno dedicado a los aspectos éticos, económicos, sociales y ambientales del proyecto y el otro analizando el presupuesto económico.

2. ESTADO DEL ARTE

2.1. IMÁGENES TÉRMICAS

Las imágenes térmicas son una parte importante de este trabajo. Los objetos desprenden cierta cantidad de radiación infrarroja, la cual es mayor cuanta más alta sea la temperatura del objeto.

El ojo humano es capaz de ver en el espectro visible, el cual se sitúa entre las longitudes de onda 380 y 750 nm, pero para ser capaz de captar la radiación que desprenden los objetos según su temperatura es necesario usar cámaras que sean capaces de captar las longitudes de onda infrarrojas, las que corresponden a las longitudes de onda entre 3 µm y 14 µm.

Las imágenes que se captan con estas cámaras se suelen colorear de manera artificial para mejorar la visibilidad de las imágenes.

En el caso de la cámara que se usará para nuestro proyecto, las imágenes térmicas obtenidas estarán coloreadas de tal forma que las zonas más frías de la imagen se verán de un color azul oscuro y según sube la temperatura el color va tomando tonos naranjas hasta llegar a colores amarillos claro, que es el color que representa a las temperaturas más altas. A estos colores se les suele llamara falsos colores o pseudocolores.

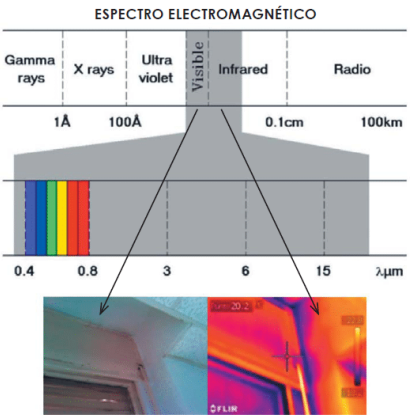

Figura 2: división del espectro infrarrojo

La cámara también es capaz de identificar la temperatura de los objetos captados por su objetivo, siendo posible calibrar la misma para obtener una correspondencia entre las temperaturas en diferentes partes de la imagen y el matiz del color representado.

2.2. RECONOCIMIENTO DE IMÁGENES

Ya que el proyecto está basado en el reconocimiento de imágenes, a continuación, se revisarán las diferentes tecnologías de inteligencia artificial y *machine learning* que vamos a utilizar como herramientas de apoyo a la consecución de los objetivos.

2.2.1. INTELIGENCIA ARTIFICIAL

Una buena definición de lo que es la inteligencia artificial (IA) es: la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano. Por lo general estas máquinas no solo igualan el rendimiento del ser humano, sino que lo mejoran considerablemente al no necesitar descansar y tener la capacidad de analizar grandes cantidades de información en un corto espacio de tiempo. Además, la proporción de errores es significativamente menor en las maquinas que en sus contrapartes humanas. [1]

Las tecnologías basadas en la IA ya están siendo utilizadas para ayudar a los humanos y simplificar sus vidas, estas tecnologías se pueden aplicar a multitud de situaciones, como la mejora del desempeño de la estrategia algorítmica comercial, el procesamiento eficiente y escalable de datos de pacientes, la distribución de contenido por redes sociales, la protección contra amenazas de ciberseguridad…

2.2.2. BIG DATA

El Big Data es la gestión y el análisis masivo, a un coste accesible, de un conjunto de datos estructurados, semiestructurados y no estructurados, de gran volumen, complejos y de diferente índole que requieren una serie de aplicaciones informáticas concretas y específicas para su procesamiento y uso. [2][3]

El Big Data precisa herramientas especiales que almacenan y ordenan los datos, les dan sentido y hacen que sean útiles, esto es, los convierte en información valiosa.

El Big Data se fundamenta en las llamadas 6 Vs que son:

1. Volumen: Cantidad de datos que son originados y almacenados con el objetivo de procesarlos y transformarlos en acciones.
2. Velocidad: Rapidez con la que los datos son creados, almacenados y procesados.
3. Variedad: Formas, tipos y fuentes en las que se registran los datos.
4. Veracidad: La calidad de los datos, es decir, el grado de fiabilidad de la información recibida.
5. Valor: Datos que se transforman en información, que a su vez se convierte en conocimiento y a su vez en una acción o decisión.
6. Variabilidad: Utilizar los datos con distintos fines.

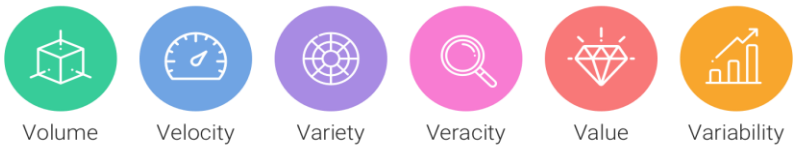

Figura 3: Las 6Vs del Big Data

2.2.3. *MACHINE LEARNING*

Machine learning es una rama de la inteligencia artificial (IA) que se centra en el uso de datos y algoritmos para imitar la forma en la que aprenden los seres humanos, con una mejora gradual de su precisión.

El concepto *machine learning* clásico depende de la intervención humana para aprender. Los expertos humanos determinan el conjunto de características necesarias para comprender las diferencias entre las entradas de datos y, por lo general, requieren muchos datos estructurados para aprender.

El concepto *machine learning* "profundo" (*Deep Learning*) puede aprovechar los conjuntos de datos etiquetados, lo que se conoce también como aprendizaje supervisado, para informar a su algoritmo. Puede ingerir datos no estructurados en formato bruto (por ejemplo, texto, imágenes) y puede determinar automáticamente el conjunto de características que distinguen las distintas categorías de datos entre sí. A diferencia de *machine learning* clásico no requiere intervención humana para procesar los datos, lo que nos permite escalar *machine learning* de maneras más interesantes. [4][5]

2.2.4. REDES NEURONALES ARTIFICIALES

Para entender bien cómo funciona una red neuronal artificial primero debemos explicar cómo funciona y que es una neurona.

La neurona es el nivel más básico de la inteligencia artificial, ésta realiza una serie de operaciones dentro de ella a partir de los estímulos externos que le llegan por las entradas y genera un valor en la salida, es decir realiza una regresión lineal. Se puede ilustrar de la siguiente forma: [6]

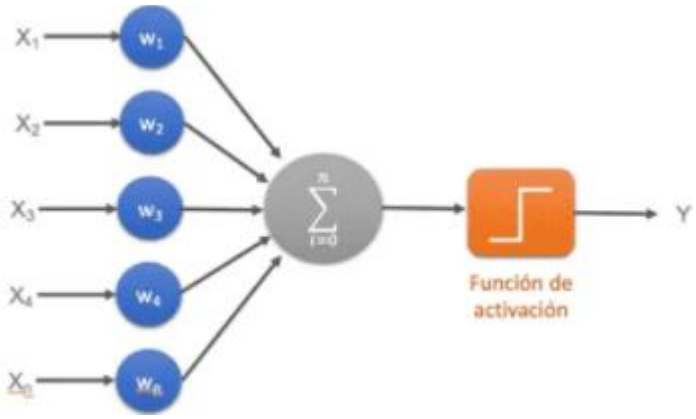

Figura 4: Representación de una neurona

Estas neuronas no se suelen usar individualmente, si no que se pueden unir varias en paralelo para tener más líneas y tener más capacidad de decisión. Al unirlas ya no se decide entre 1 o 0, sí o no, sino que se puede elegir cualquier cosa. Por esta razón las neuronas se usan en grandes cantidades dando lugar a lo que se conoce como redes neuronales.

La gran cantidad de neuronas que componen las redes neuronales se suelen organizar en capas, cada una de las cuales hace una tarea determinada.

Por lo general encontramos tres tipos de capas, una primera llamada capa de entrada, la última llamada capa de salida y las capas intermedias a las que se le llaman capas ocultas.

Esta disposición de las capas permite que nuestra red pueda aprender un conocimiento jerarquizado. Por lo general las decisiones que deben tomar estas redes son complejas por lo que necesita una gran cantidad de parámetros y cada parámetro tiene su importancia. Esto lo conseguimos al tener cada capa de neuronas colocadas en paralelo, lo que nos permite que el resultado de una capa se lo pasemos a la siguiente para que pueda volver a tomar otra decisión y así sucesivamente.

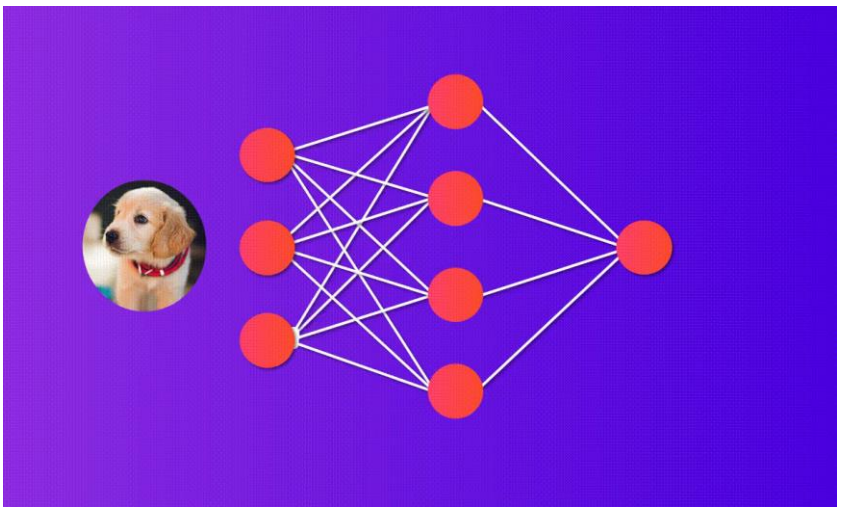

Figura 5: Ejemplo de representación de una red neuronal

2.2.5. FUNCIÓN DE ACTIVACIÓN

En algebra elemental la suma de elementos lineales produce un resultado lineal.

Por esta razón si se unifican varias neuronas en varias capas, como en la figura 5, uniendo la salida de una neurona con la entrada de otra y no aplicamos nada más, no conseguiremos aumentar nuestro poder de decisión.

La red neuronal quedaría reducida a un modelo sustituible por una única neurona. Como es lógico esto no nos interesa pues estamos derrochando cientos, miles de cálculos para un trabajo totalmente desaprovechado.

La función de activación se aplica para evitar el problema que se acaba de comentar, dependiendo de cada caso utilizaremos una función u otra, pero pueden ser funciones de tipo sigmoide, umbral, hiperbólica, etc. Sobre estas funciones matemáticas no se entrará en detalle pues se considera que la explicación sobre el uso de una función u otra se considera fuera del estudio de este trabajo de fin de grado.

Una vez deslinealizada la función de la neurona, el resultado del uso de neuronas en varias capas sí tiene sentido y conseguimos lo que hemos querido siempre, aumentar el poder de decisión. Ahora que se ha realizado esto, podemos formar una red neuronal completa.

2.2.6. CONJUNTOS DE DATOS: *DATASETS*

Un *dataset* es un conjunto de datos estructurado y etiquetado que puede ser utilizado como entrada para una red neuronal, la cual es capaz de aprender y tomar decisiones conforme a lo aprendido. Este proceso se llama Entrenamiento.

El tipo de información que contienen estos datos es distinto para cada caso, por ejemplo, en nuestro caso particular requerimos de un gran número de pares de fotografías (cada par de fotografías corresponderán a la misma escena, pero una de ellas representará la imagen en el espectro visible y la otra en el espectro infrarrojo.

Una de las formas más comunes de etiquetar un *dataset* de imágenes es con recuadros que marquen los objetos que requiramos para el futuro entrenamiento de la red. Pero cuando se habla de imágenes recuadradas no es tan trivial como pintar un cuadro en el objeto, sino que debe crearse un archivo o varios aparte que contendrán las coordenadas de cada objeto recuadrado en cada una de las imágenes.

Dependiendo del tipo de red que vayamos a usar este archivo será de una forma u otra, por ejemplo, en nuestro caso, que usamos Detectron2, es un único archivo ***json*** con la información estructurada de todas las imágenes, sin embrago, en una red tipo YOLO se utiliza un archivo TXT para cada imagen.

En un *dataset* no solo importa la cantidad de información, la calidad de la misma también es muy importante. Es por esto por lo que encontrar un *dataset* adecuado es uno de los procesos más importantes y complejos en el desarrollo de una red neuronal. Para ayudar en esta etapa hay iniciativas de empresas como Google que facilitan la búsqueda de *datasets*.

2.2.7. ENTRENAMIENTO DE UNA RED

Una red neuronal artificial está basada en el funcionamiento de las redes de neuronas biológicas y como tal, antes de tomar decisiones debe tener un proceso de aprendizaje. En el caso de las redes neuronales artificiales este aprendizaje se realiza durante el proceso de entrenamiento.

Para realizar el proceso de entrenamiento de una red neuronal, previamente debemos haber escogido el dataset adecuado, puesto que será sobre el cual se basará la red neuronal para aprender. Lo que sucede al entrenar una red neuronal es que se ajustan los pesos de las entradas de cada neurona para que a la salida de la red el resultado sea lo más ajustado posible al resultado deseado.

El hecho de usar un dataset amplio nos ayuda a poder generalizar y así poder tomar la decisión correcta sea cual sea la entrada. Otra característica importante en el dataset que usemos es que, a parte del gran número de datos etiquetados con el objetivo al que queremos llegar, haya datos sin ninguna relación al objetivo final y etiquetados como 0, esto ayuda a la red neuronal a aprender que no siempre va a encontrar lo que busca.

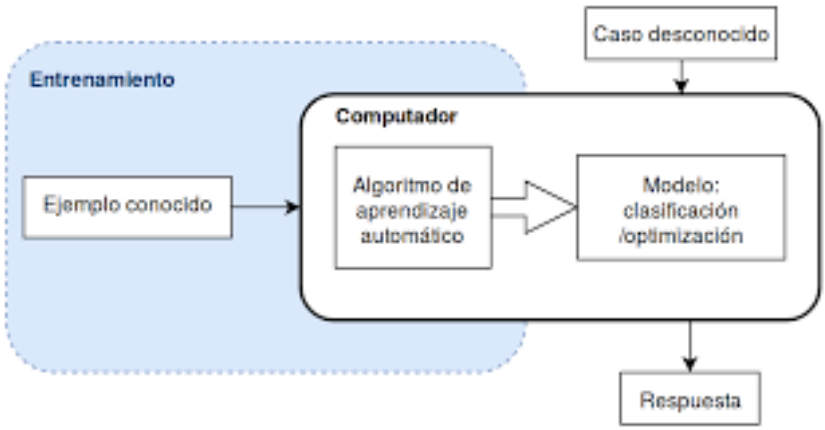

Figura 6: Diagrama del entrenamiento en una red neuronal

2.2.8. FUNCIÓN DE PÉRDIDA Y EL DESCENSO DEL GRADIENTE

Cuando entrenamos una red necesitamos comprobar que el resultado es el esperado, para esto usamos lo que se conoce como función de pérdida. Esta función mide la desviación que hay entre las predicciones de nuestra red neuronal entrenada y los valores reales de las observaciones utilizadas durante el aprendizaje. Cuanto menor es el resultado de esta función, más eficiente es la red neuronal.

No solo existe una única función capaz de medir esto, se pueden usar funciones como el error cuadrático medio o la entropía cruzada.

Para poder reducir esta pérdida, es necesario modificar los pesos y recalcular el modelo. Durante los primeros años de las redes neuronales, para obtener la menor pérdida posible se calculaba la pérdida de cada parámetro posible y luego se determinaba el menor peso. Aunque eficaz, esta técnica se descartó puesto que era muy lenta y en su lugar se empezó a usar la técnica del descenso del gradiente. Esta técnica consiste en: desde un punto inicial calcular el gradiente de la curva de pérdida en este punto, lo

que nos indicara en qué dirección se deben ajustar los pesos para minimizar la pérdida.

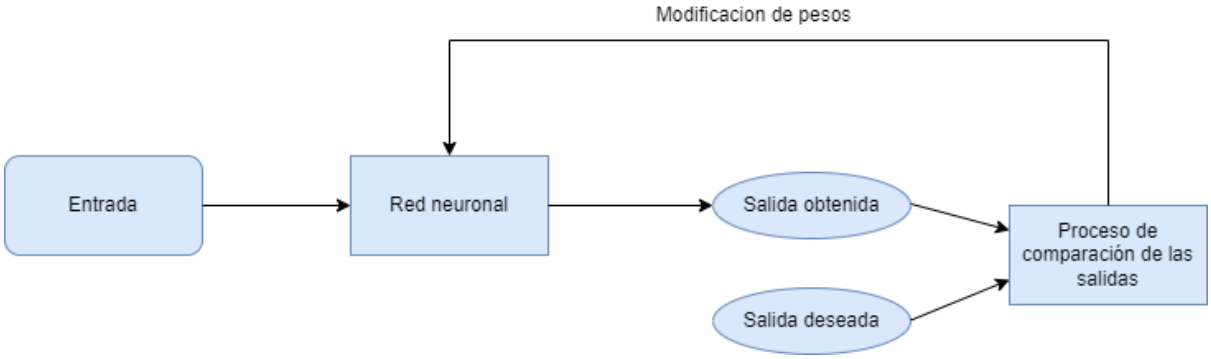

Figura 7: Diagrama de aprendizaje por corrección de error

Aunque es cierto que por lo general siempre se busca minimizar la pérdida lo máximo posible, siempre existirá algo de pérdida. Esto se debe a que, si la pérdida de nuestra red es 0, no será capaz de generalizar, sufriendo lo que se llama un sobre-entrenamiento de la red neuronal.

2.2.9. REDES NEURONALES CONVOLUCIONALES

Existen varios tipos de redes neuronales, por lo que según los objetivos que se busquen habrá que usar uno u otro. Por ejemplo, cuando se trata de trabajar con imágenes, como nuestro caso, es habitual utilizar las redes neuronales convolucionales.

Estas redes neuronales convolucionales (CNN) procesan sus capas imitando al córtex visual del ojo humano para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos.

Para ello, la CNN contiene varias capas ocultas especializadas y con una jerarquía: esto quiere decir que las primeras capas pueden detectar líneas, curvas y se van especializando hasta llegar a capas más profundas que reconocen formas complejas como un rostro o un vehículo.

Las redes convolucionales toman como entrada cada píxel de la imagen, en el caso de que las imágenes tuviesen 320x240 pixeles equivaldría a un total de 76800 neuronas si la imagen fuese en blanco y negro y el triple si es a color.

Una vez determinada la entrada la red realiza las llamadas convoluciones, que consisten en tomar "grupos de pixeles cercanos" de la imagen de entrada e ir operando matemáticamente (producto escalar) contra una pequeña matriz que se llama *kernel*. Este *kernel* se puede considerar como un filtro con el que se extraen ciertas características importantes o patrones de la imagen.

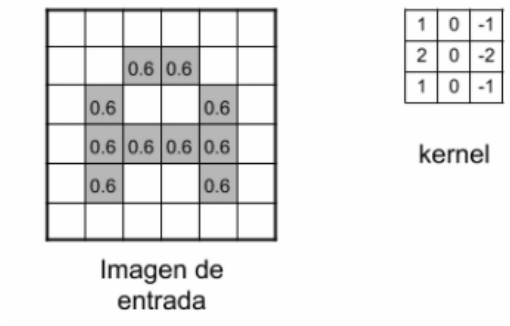

Figura 8: Representación del *kernel*

Este proceso se genera en las capas de convolución. Una vez el *kernel* ha recorrido todas las neuronas de la entrada se obtiene una nueva matriz a la que se le aplica la función de activación. Esta función suele ser la sigmoide ReLu (Rectified Linear Unit) pero pueden utilizarse otras.

Antes de pasar a la siguiente convolución se debe reducir el número de neuronas, puesto que si no se hace estas van creciendo exponencialmente. Para este proceso se usa un proceso de submuestreo en el que reduciremos el tamaño de nuestras imágenes filtradas, pero en donde deberán prevalecer las características más importantes que detectó cada filtro.

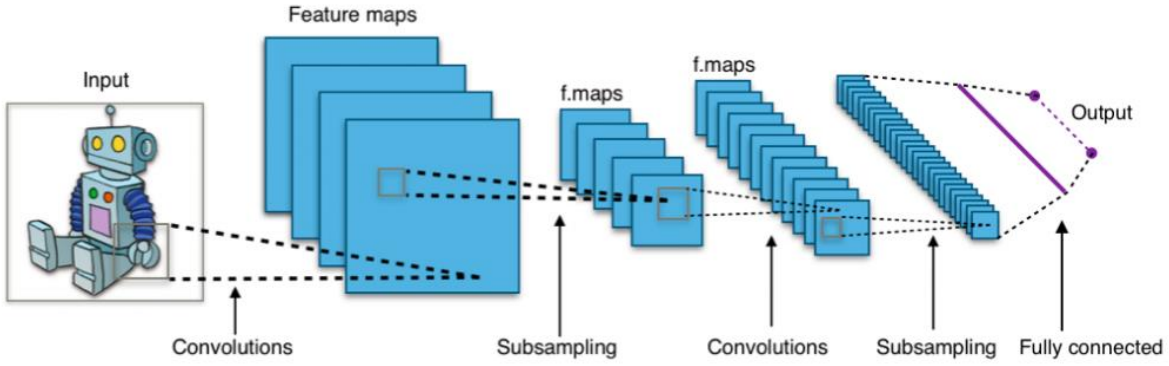

Figura 9: Esquema gráfico de una red convolucional

2.2.10. REDES RCNN, DETECTRON2

Dentro de las redes convolucionales existen las redes RCNN (Region Based Convolutional Neural Networks). Estas redes primero seleccionan varias regiones para la imagen y anotan su cuadro delimitador. Luego, la red convolucional se usa para realizar cálculos directos en cada región propuesta para extraer características. Después de eso, introduce su salida en lo que se conoce como un SVM (*Support Vector Machine*). Este algoritmo observa si el objeto que queremos clasificar está dentro de lo que hemos buscado.

Otra característica de estas redes es que usan ciertos *offset* para ajustar mejor el recuadro de la detección.

Aunque en teoría esta red mejora mucho los resultados de las redes CNN, lo cierto es que hardware actual no está a la altura para poder mantener un ratio de imágenes por segundo decente para su uso en aplicaciones de reconocimiento de videos. [30]

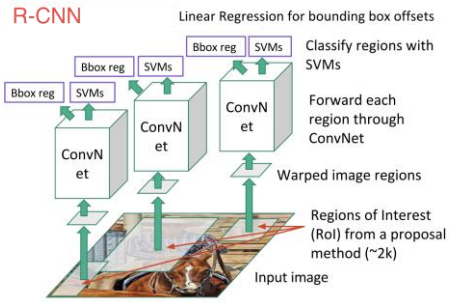

Figura 10: Ejemplo de la detección de una red RCNN

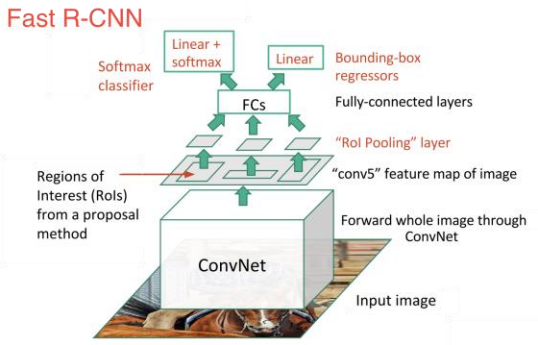

Figura 11: Ejemplo de la detección de una red Fast RCNN

Los autores originales del algoritmo, He Kaiming y RG Dashen entre otros, adaptaron ciertas características del modelo para solventar este problema y así crear una versión más rápida, creando así las redes Fast-RCNN. En esta red cuando extraen las características y los candidatos a regiones propuestas no se introducen a la red directamente, sino que se utiliza lo que se llama un RoI pooling vector (*Region of interest pooling vector*).

Con esto conseguimos unos vectores que nos indican si una zona debe ser tenida en cuenta y ser introducida en la red o simplemente es parte el fondo de la imagen y se puede descartar. Gracias a esto podemos reducir considerablemente los candidatos y así reducir en gran medida el tiempo de procesamiento necesario para cada imagen.

Dentro de las redes neuronales convolucionales Fast-RCNN se encuentra la red que usaremos durante este proyecto, la red Detectron2, la segunda versión de la red Detectron. Esta red nace de la investigación de un equipo de desarrollo de la empresa Facebook, actualmente conocida como Meta, sobre las redes Fast RCNN. [11]

Figura 12: Logo de Detectron2

2.3. HERRAMIENTAS DE TRABAJO

2.3.1. AMBIENTE DE TRABAJO

Como primer paso para manejar datasets, así como para entrenar y probar los sistemas de redes neuronales necesarios para el proyecto, es básico utilizar un ambiente de trabajo para la ciencia de datos, como es Anaconda Navigator.

Este ambiente de trabajo nos permite trabajar con el lenguaje Python y además gracias a Jupyter Notebook, podremos ir creando bloques de programas en Python e irlos comprobando bloque por bloque de forma interactiva, así podremos ir haciendo paso a paso nuestros ejercicios y nuestros propios proyectos de aprendizaje automático e inteligencia artificial. [18]

Figura 13: Logo de Anaconda Navigator

Dentro de Anaconda Navigator podremos crear diferentes entornos de trabajo de Python donde importar las librerías e implementar la red neuronal sin afectar al resto de sistemas.

Tal y como he mencionado antes, gracias a usar este ambiente de trabajo podremos trabajar con Jupyter Notebook, que es una aplicación cliente-servidor lanzada en 2015. Permite crear y compartir documentos web en formato JSON que siguen un esquema versionado y una lista ordenada de celdas de entrada y de salida. Estas celdas albergan, entre otras cosas, código, texto, fórmulas matemáticas y ecuaciones, o también contenido multimedia.

Una gran ventaja que tiene es que el código se organiza en celdas independientes, lo que hace posible probar bloques concretos de código de forma individual. El programa se ejecuta desde la aplicación web cliente que funciona en cualquier navegador estándar, haciendo un uso local del ordenador sobre el que está instalado, permitiendo usar su tarjeta gráfica para acelerar los cálculos. [17]

Figura 14: Logo de Jupyter Notebook

2.3.2. GOOGLE COLABORATORY

Google Collaboratory, comúnmente conocido como Colab, es un servicio gratuito en la nube, ofrecido por Google para fomentar la investigación sobre inteligencia artificial.

Al igual que Jupyter Notebook, tiene un sistema de celdas que pueden contener textos explicativos o códigos ejecutables. Estas celdas también tienen la posibilidad de ejecutarse independientemente para poder probar nuestro código por bloques y no todo de golpe.

Una gran ventaja que tiene Colab con respecto a Jupyter es la posibilidad de trabajar con el entorno sin tener que instalar nada en nuestro ordenador, ya que basta con acceder a través de la web a un notebook y empezar a trabajar con el mismo.

Figura 15: Logo de Google Collaboratory

La principal característica de Colab es que el código no se ejecuta en tu terminal, si no que se ejecuta de forma remota en unas GPUs proporcionadas por Google y cuyo uso es gratuito.

Gracias a esto podemos liberar a nuestra máquina de los costosos cálculos que conlleva el *machine learning*, además de poder acceder a este mundo sin la necesidad de tener un equipo excesivamente potente.

Aunque es verdad que el uso de las GPUs remotas es gratuito, este acceso está restringido a un máximo de 12 horas, las cuales se recargan a las 48 horas. Para aquellos usuarios que requieran un uso más prolongado existe la versión de pago llamada Colab pro.

Esta versión de pago cuesta 9,99€ al mes y permite la utilización de GPU,s más potentes así como un acceso más prolongado a las GPU remotas.

Además de esta versión existe una versión Colab Pro+ con un coste de 49,99€ al mes, la cual permite la posibilidad de ejecutar los cuadernos en segundo plano y el acceso a GPUs aún más rápidas. [20]

3. DESARROLLO

3.1. ELECCIÓN DE LA CAMARA

Figura 16: Cámara Flir C3

Como apuntábamos en la introducción, uno de los primeros pasos del proyecto es conseguir una gran cantidad de imágenes, tanto en espectro visual como en espectro infrarrojo, que nos sirvan de base para la creación de los datasets necesarios para el entrenamiento, validación y prueba de la red neuronal.

Por tanto, había que elegir la cámara adecuada para captar las imágenes tanto del espectro visible como del térmico. Finalmente, la cámara elegida fue la Flir C3, capaz de obtener imágenes simultaneas en ambos espectros. El rango espectral infrarrojo que capta esta cámara es desde los 7,5 hasta los 14µm.

Otra de las características de esta cámara es que las imágenes térmicas las capta a una resolución de 320x240 con una relación de aspecto de 4:3. En cuanto a las temperaturas que es capaz de obtener oscilan entre –10 y +100°C con un error de ±2℃. Por otro lado, las imágenes del espectro visible las capta a una resolución de 640x480 con la misma relación de aspecto que antes.

La cámara es capaz de almacenar en su memoria interna unos 500 grupos de imágenes (una en el espectro térmico y otra en el visible), lo cual es muy útil para la creación del nuevo dataset. [21]

El precio de la cámara ronda unos 600 euros lo que la convierte en una gran opción en cuanto a la relación calidad/precio.

3.2. ELECCIÓN DEL RESTO DEL HARDWARE A UTILIZAR

Ya elegida la cámara, hay que elegir el resto del hardware que vamos a utilizar para el desarrollo del código y del entrenamiento de la red neuronal.

Cuando tenemos que elegir un hardware que va a ser usado en el ámbito de la inteligencia artificial es importante tener en cuenta la potencia de éste, puesto que es un factor que afecta mucho en este ámbito. En especial es importante la elección de la GPU, que es la que más trabajo de cómputo realiza en temas de inteligencia artificial.

En nuestro caso, el hardware elegido será un ordenador personal que cuenta con una tarjeta gráfica Nvidia RTX 3070, con una gran capacidad de cómputo al tener núcleos especializados para la IA. Además de esta GPU, el ordenador elegido cuenta también con una CPU Intel i5 de decima generación.

3.3.ELECCIÓN DEL PROGRAMA PARA PRE-PROCESAR LAS IMÁGENESDE AMBOS ESPECTROS

La cámara elegida capta imágenes tanto en el espectro visible como en el térmico, pero las capta a diferentes resoluciones y por defecto sólo guarda las imágenes en uno de los dos formatos.

Para simplificar nuestro sistema es conveniente que las imágenes de ambos espectros tengan las mismas dimensiones y la misma perspectiva.

Para conseguir esto, hemos usado un software desarrollado por FLIR, la misma empresa que fabrica la cámara, llamado FLIR Tools.

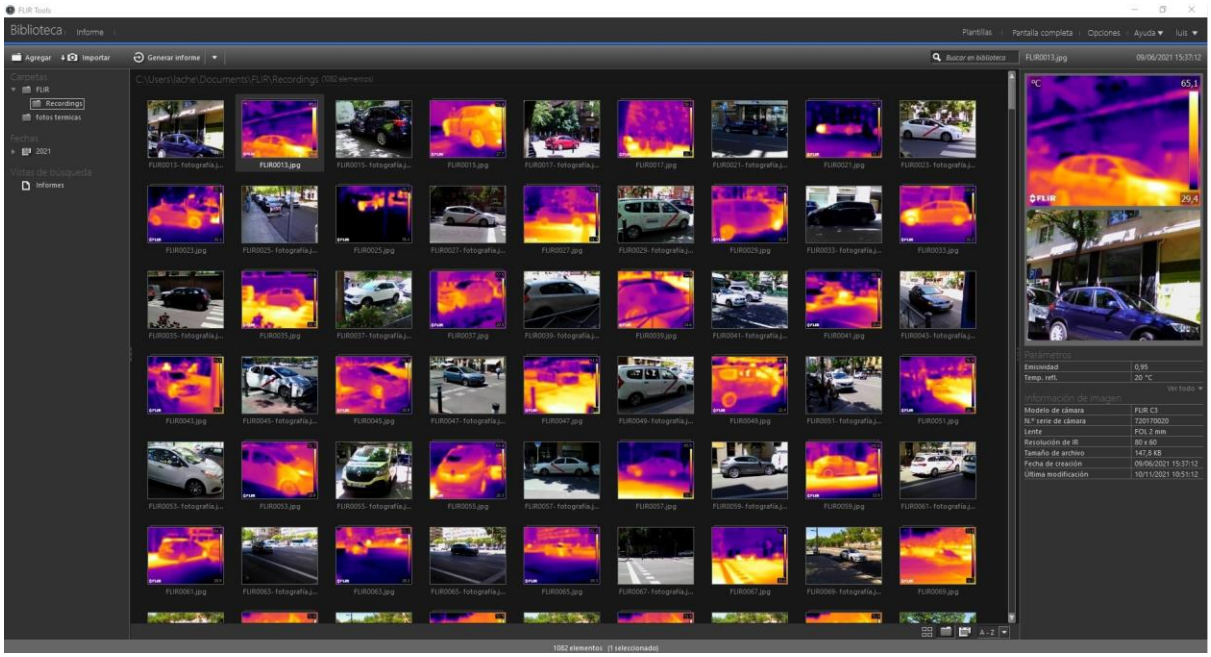

Figura 17: FLIR Tools

Este programa nos permite tratar las imágenes captadas por la cámara, podemos recorrer cada una de las imágenes captadas y separar en archivos diferentes las imágenes de un espectro y otro, igualando las resoluciones de ambas modalidades. [22]

Gracias a esto podremos obtener una carpeta con todas las imágenes del espectro visible y otra con todas las del espectro térmico, lo que nos será muy útil en el desarrollo de las fases de reconocimiento y entrenamiento del sistema.

3.4. ELECCIÓN DE LENGUAJE DE PROGRAMACIÓN Y LIBRERIAS FUNDAMENTALES

A la hora de elegir lenguaje de programación no se tuvo ninguna duda que el lenguaje más adecuado para el proyecto era Python. Se trata del lenguaje más estandarizado en el mundo de la inteligencia artificial, con infinidad de módulos y utilidades creados que simplifican el desarrollo de proyectos complejos.

La versión de **Python** elegida es la **3.6.13**. No hemos utilizado una versión posterior ya que vimos que ésta es compatible con todas las librerías y módulos necesarios para el desarrollo. Las pruebas realizadas para utilizar versiones posteriores ocasionaban errores en la ejecución de los procesos.

Al final, conseguir un entorno software estable es una tarea tediosa y minuciosa en la que hay que alcanzar un punto de equilibrio estable entre las funcionalidades aportadas por cada módulo o librería y la compatibilidad entre todos los necesarios.

Entre los módulos descargados podemos destacar como críticos:

pytorch, v1.8.1 y torchvision, v0.9.1, estos módulos consisten en librerías usadas para la ejecución de programas de *machine learning*, realizando multitud de cálculos simultáneos, al hacer uso de la programación de tensores. Además, son requeridos por Detectron 2 que se basa en ellos para la ejecución de los cálculos necesarios.

cudatoolkit, v11.1.74, esta librería nos permite utilizar la capacidad de proceso de la GPU Nvidia del ordenador para acelerar el tiempo de los procesos utilizando computación paralela en las unidades de proceso gráfico de la tarjeta. La librería CUDA (*Compute Unified Device Architecture - Arquitectura Unificada de Dispositivos de Cómputo*) está pensada para lenguajes de programación como C o C++, aunque posee un port para poder ser usada en Python y es compatible también con la librería Pytorch.

opencv-python, v4.5.2.54, OpenCV es la librería imprescindible para realizar tanto el tratamiento digital de las imágenes como gestionar su entrada y salida, presentación en pantalla, escritura al disco, etc.

Adicionalmente, en el entorno creado se instalaron otros módulos que son necesarios para diferentes apartados de la ejecución de los procesos, podríamos destacar:

numpy y **pandas** para proceso numérico

json para tratamiento de los ficheros de tipo json empleados en la creación de los datasets

path para simplificar el acceso a los ficheros en disco

random para aleatorizar determinados procesos

3.5. ELECCIÓN E INSTALACIÓN DE LA RED NEURONAL

Tras analizar las diferentes redes neuronales disponibles, se llegó a la conclusión que las que mejor se adaptaban a nuestras necesidades eran las redes YOLO (You Only Look Once) y la red Detectron2 .

Estudiando más a fondo ambas opciones, haciendo pruebas básicas de instalación y utilización con conjuntos de datos reducidos, tras revisar diferentes experimentos comparativos en internet, se llegó a la conclusión que la mejor opción para el desarrollo de este proyecto era Detectron2.

Figura 18: Logo Detectron2

Esta decisión se tomó en base a las características principales que ofrecen estas redes. Por un lado, tenemos las redes YOLO que se caracterizan por ser redes simples de usar y que realizan una detección muy rápida. Por su parte, la red Detectron2 tiene como gran ventaja una gran precisión en sus resultados. Esta fue la principal característica por la que se tomó la decisión, puesto que en nuestro trabajo no nos importa tanto que se realice una detección rápida, sino que sea precisa.

Para implementar la red, primero creamos un nuevo entorno en el ambiente de trabajo de Anaconda Navigator. El primer paso necesario para la preparación del entorno de programación adecuado es la instalación tanto de Python como de los módulos adicionales necesarios, cada uno de ellos en las versiones adecuadas para poder ejecutar los programas sin errores ni incompatibilidades. Algunos de los módulos que han sido cargados en el entorno son los representados en la siguiente figura.

```
In [1]: conda list
# packages in environment at /home/luis/anaconda3/envs/PFG_env:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                         main
_openmp_mutex             4.5                       1_gnu
absl-py                   0.13.0                    pypi_0    pypi
alabaster                 0.7.12                    py36_0
antlr4-python3-runtime    4.8                       pypi_0    pypi
argon2-cffi               20.1.0           py36h27cfd23_1
astroid                   2.5              py36h06a4308_1
async_generator           1.10             py36h28b3542_0
attrs                     21.2.0             pyhd3eb1b0_0
babel                     2.9.1              pyhd3eb1b0_0
backcall                  0.2.0              pyhd3eb1b0_0
blas                      1.0                         mkl
bleach                    3.3.0              pyhd3eb1b0_0
brotlipy                  0.7.0            py36h27cfd23_1003
bzip2                     1.0.8                h7b6447c_0
ca-certificates           2021.5.25            h06a4308_1
cachetools                4.2.2                     pypi_0    pypi
certifi                   2021.5.30        py36h06a4308_0
cffi                      1.14.5           py36h261ae71_0
chardet                   3.0.4                     pypi_0    pypi
cloudpickle               1.6.0                       py_0
colorama                  0.4.4              pyhd3eb1b0_0
cryptography              3.4.7            py36hd23ed53_0
cudatoolkit               11.1.74              h6bb024c_0    nvidia
cycler                    0.10.0                    pypi_0    pypi
cython                    0.29.23                   pypi_0    pypi
dataclasses               0.8                pyh4f3eec9_6
```

Figura 19: Captura de algunas de las librerías importadas en el entorno de trabajo

Tras la importación de estos módulos necesarios para el correcto funcionamiento de la red, el siguiente paso fue la instalación del paquete Detectron2, así como de los módulos encargados de la ejecución del código, además de un conjunto de modelos pre-entrenados, llamado *model_zoo*.

3.6. CREACIÓN DEL *DATASET*

La primera necesidad que surgió en el desarrollo del proyecto fue la disponibilidad de un dataset específico para el entrenamiento de un sistema adaptado para el reconocimiento de imágenes infrarrojas.

En un principio se pensó en usar algún dataset ya creado, pero no se encontró ninguno que cumpliese las necesidades que requiere nuestro sistema. Esto es porque nuestro sistema requiere de dos grupos de imágenes que sean iguales solo que unas en el espectro visible y otras en el térmico.

Puesto que no se podía utilizar ningún dataset ya creado se decidió crear uno propio con la ayuda de la cámara térmica previamente seleccionada, la FlirC3.

La creación manual de un dataset es un proceso complejo, laborioso y lento ya que exige una alta carga de trabajo manual: capturando un alto número de imágenes, reconociendo visualmente uno a uno los objetos en cada imagen y marcando exactamente para cada imagen la identificación y localización de los objetos reconocidos visualmente.

Uno de los principales objetivos de este trabajo es reducir considerablemente esta alta carga de trabajo a través de automatizar el proceso haciendo uso de Detectron2, con un modelo estándar de reconocimiento, para identificar y localizar los objetos en una serie de imágenes de espectro visual. Posteriormente realizamos un proceso de asignación de la información recogida a la serie correspondiente de imágenes térmicas. Con esto reducimos considerablemente el trabajo de creación del dataset de imágenes infrarrojas y a la vez conseguimos desarrollar unos procedimientos y el código correspondiente para duplicar el proceso para cualquier otro tipo de tareas que exijan el reconocimiento de imágenes infrarrojas de cualquier tipo.

3.6.1. OBTENCIÓN DE LAS IMÁGENES DEL *DATASET*

Una vez decidido que íbamos a crear nuestro propio dataset, había que decidir el tema principal del dataset. Las principales ideas fueron: personas, coches y animales de compañía. Estas opciones parecían adecuadas ya sus imágenes tienen una marca de calor característica y son abundantes para facilitar la creación sencilla un gran dataset.

Finalmente, la opción escogida para este proyecto fue la de imágenes de coches. En el caso de personas sí que existen datasets de imágenes térmicas, pero pretendíamos hacer algo nuevo y explorar el tema de la generación automática de datasets anotados. En el caso de animales era algo más complicado conseguir un dataset amplio en comparación con los coches.

Con el tema ya escogido, había que iniciar el proceso de obtención de las imágenes del dataset. La primera recogida de imágenes fue a mediados de julio de 2021, a mediodía en el centro de Madrid, al iniciar la captura de imágenes se encontró el primer problema, al captar imágenes de los vehículos en la carretera, estos casi no se aprecian puesto que la alta temperatura del asfalto (llegando hasta los 80ºC) hace que, en el espectro infrarrojo, casi ni se distinguen los vehículos que circulan por ella.

Viendo lo que ocurría, ese día se trató de tomar únicamente fotos en zonas de sombra, donde la temperatura del asfalto fuese menor.

Figura 20: Ejemplo imagen térmica en Julio

Como solución a este problema los siguientes días que se fue a obtener imágenes se decidió ir a primera hora de la mañana para evitar las altas temperaturas en la calzada. Esta solución fue bastante efectiva, aunque acarreó otro problema, y es que algunos coches acababan de ser arrancados y todavía no habían entrado en calor. Aunque esto en un principio parecía un gran problema, en realidad al final no ha supuesto tanto, puesto que, aunque la marca térmica característica de los coches (alta temperatura en motor, ruedas delanteras y tubo de escape) no esta tan marcada, sí que es posible apreciar la silueta del coche, por lo que siguen siendo útiles para nuestro *dataset*.

Durante esta primera fase de recogida de imágenes se tomaron unos 100-150 pares de imágenes, las cuales eran suficientes para iniciar a desarrollar el código, pero no para desarrollar y entrenar la red neuronal completa. Es por esto por lo que durante finales de octubre y principios de noviembre se volvió a salir a tomar nuevas imágenes para completar el dataset. Puesto que las temperaturas en estas fechas eran bastante más bajas que en verano, no hubo ningún problema sobre la hora del día a la que sacar las imágenes.

Una ventaja que todo esto proporciona al proyecto es que la alta variabilidad de las circunstancias en que fueron captadas las imágenes permite alimentar al sistema con un conjunto de datos muy diverso y hacer que su entrenamiento sea más rico.

Por otro lado, la variabilidad hace que existan una serie de imágenes en el espectro visible que desafían a los sistemas de reconocimiento tradicionales haciendo que a menudo no puedan ser reconocidas. Un ejemplo es el de la siguiente figura, que es la misma del apartado de Introducción, en este caso la imagen del vehículo en el espectro visible está en una zona de sombra y no puede ser reconocido en el sistema tradicional.

Figura 21: Misma imagen en ambos espectros

El nuevo sistema creado en este proyecto sí que es capaz de reconocer el vehículo cuando actúa sobre la imagen de infrarrojos de este.

Finalmente, el *dataset* cuenta con 540 pares de imágenes de 320x240 pixeles.

3.6.2. ETIQUETADO DEL *DATASET*

Como ya se ha explicado anteriormente, tener un *dataset* bien etiquetado es una de las tareas más importantes en el uso de redes neuronales. Es muy común que el etiquetado se realice a mano imagen por imagen. Sin embargo, esto significa un gran coste en cuanto a tiempo y esfuerzo. Por esta razón, este proyecto automatiza el proceso de etiquetado de *datasets* de imágenes en el espectro térmico a partir del reconocimiento de objetos de las mismas imágenes en el espectro visible.

Para realizar este paso usaremos la red neuronal Detectron2 previamente instalada. Gracias a que se ha instalado un modelo ya entrenado para detectar hasta 80 tipos de objetos diferentes, entre los cuales se encuentran los coches que es lo que nos interesa, se puede usar la red para que detecte la posición de los coches en nuestras imágenes del espectro visible.

Hay que recordar que no solo se necesita que la red detecte los objetos y los enseñe, necesitamos también generar un código que por cada imagen que se introduce a la red nos devuelva un archivo con las coordenadas y características del o los objetos encontrados.

Este archivo será un .txt que constará de una primera línea que nos indica el número de objetos identificados y las claves de los mismos: números del 0 (persona) al 79 (cepillo de dientes). En nuestro caso concreto la clave que nos interesa es la número 2 (coche).

En la segunda línea del fichero se guardan las coordenadas x e y de los vértices que identifican cada una de las cajas en las que se enmarcan estos objetos.

```
tensor([2, 2, 2], device='cuda:0')
Boxes(tensor([[300.2196,  45.1572, 320.0000,  72.0875],
        [  0.0000,  24.2089, 256.3420, 145.0836],
        [202.6482,  45.4186, 239.6194,  62.9462]], device='cuda:0'))
```

Figura 22: Ejemplo del contenido de un archivo TXT

Una vez que se obtienen los archivos con las coordenadas de los objetos se procede a comprobar visualmente que el proceso ha funcionado. Para ello, se procede a representar las coordenadas extraídas de las imágenes del espectro visible en las mismas imágenes, pero del espectro térmico y ver si corresponden también a los objetos.

```
tensor([2], device='cuda:0')
Boxes(tensor([[ 68.8082,  74.9540, 180.7619, 118.9766]], device='cuda:0'))
```

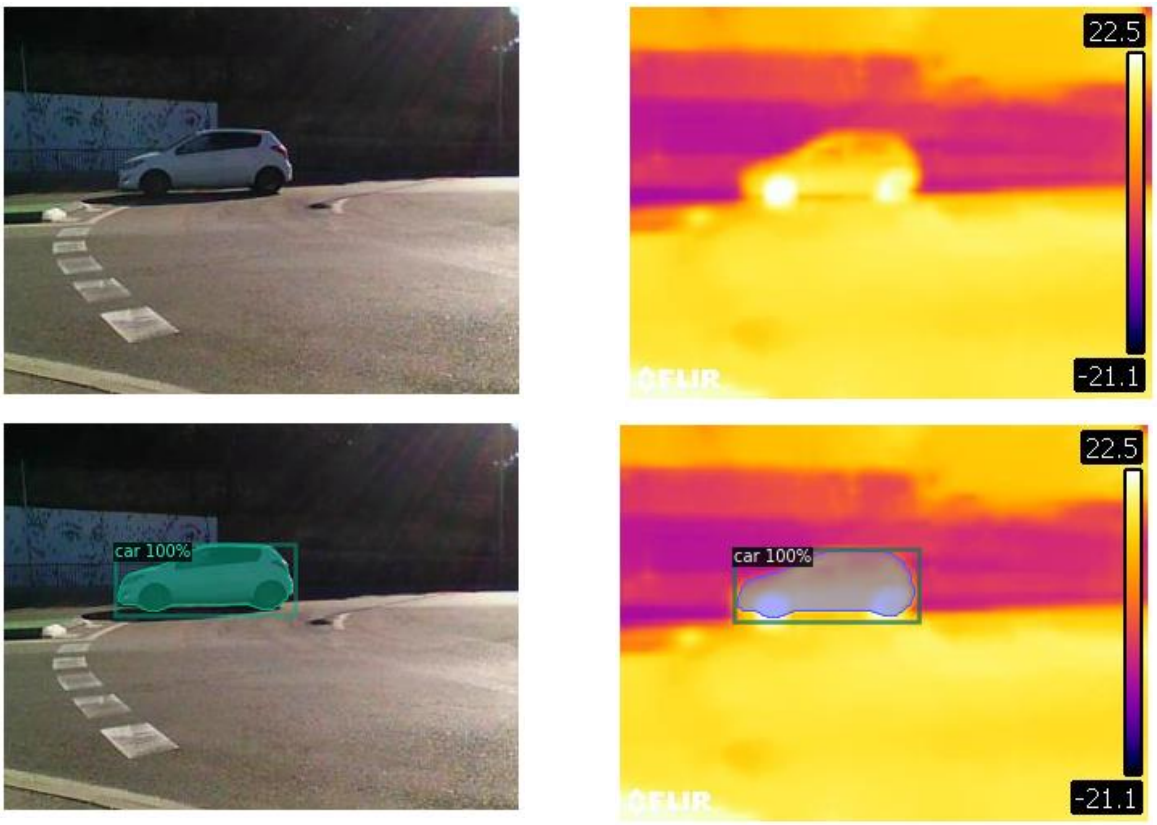

Figura 23: Asignación del reconocimiento a la imagen térmica

Al realizar la comprobación visual del resultado obtenido se pudo observar que en la gran mayoría de imágenes se detectaban sin problemas los vehículos, además de otros objetos que podían aparecer en la imagen. En las ocasiones que el sistema no era capaz de detectar un coche que realmente estaba en la imagen se trataba de casos específicos en los que la baja calidad de la imagen sumado a una mala iluminación hacía que apreciar la existencia de un vehículo en la escena fuese difícil incluso para el ojo humano. Un buen ejemplo de esto está representado en la figura 21.

Curiosamente en las imágenes en el espectro térmico de estas imágenes sí que se podía apreciar la figura del coche fotografiado, por lo que, aunque fueron descartadas para el entrenamiento de la red al no estar etiquetadas, se usaron con gran éxito para probar el funcionamiento final del sistema.

Como se mencionó en el apartado 2.2.6. los *datasets* que se usan en la red Detectron2 tienen toda la información de etiquetado de las imágenes en un único archivo del tipo ***.json***, adaptado al formato COCO. Por esta razón el siguiente paso era transformar y combinar todos los archivos **.txt** creados anteriormente en un único archivo *json*.

Para llevar a cabo esta transformación había que leer consecutivamente cada línea de cada archivo **.txt** para extraer su información, cambiar el formato y añadirla al archivo json estandarizado con la estructura definida por el estándar COCO con el que trabaja la red Detectron2.

El estándar COCO está estructurado como un diccionario que contiene una primera lista que identifica todas las imágenes que componen el dataset, detallando el nombre del fichero en el disco que contiene la imagen, así como sus dimensiones (320x240 en nuestro caso).


```
{
  "images": [
   {
    "file_name": "FLIR0375.jpg",
    "height": 240,
    "width": 320,
    "id": "FLIR0375"
   },
   {
    "file_name": "FLIR0497.jpg",
    …
```


La siguiente lista de la que consta el archivo *json* en formato COCO contiene la información o anotaciones de los objetos detectados: las categorías o claves y las coordenadas de localización. Como en nuestro proyecto nos vamos a entrar en los coches detectados, en este paso procedemos a solo introducir en el archivo *json* la información de la categoría que nos interesa.

En el caso de las coordenadas, el formato COCO estipula que se expresen como las coordenadas del vértice de menor **x** y menor **y**, junto con la altura y anchura de la caja, mientras que la información que tenemos en los archivos .txt corresponde a los vértices de (x, y) mínimos y de (x, y) máximos. Por esta razón antes de escribir las coordenadas en el archivo *json* es necesario calcular la altura y anchura de la caja. Además de las coordenadas de las cajas, también hay que calcular el área de las cajas para introducirlo como parámetro en el archivo.


```
],
  "type": "instances",
  "annotations": [
   {
    "area": 2233,
    "iscrowd": 0,
    "image_id": "FLIR0375",
    "bbox": [
     93,
     88,
     77,
     29
    ],
    "category_id": 0,
    "id": 1,
    "ignore": 0,
    "segmentation": []
   },
   {
    "area": 7215,
    "iscrowd": 0,
    "image_id": "FLIR0497",
            …
```


Una vez que el código ha recorrido todas las imágenes y ha escrito toda la información extraída de los txt en el nuevo archivo *json* ya sólo queda añadir el apartado de la lista de las categorías o claves de los objetos detectados, que en nuestro caso es únicamente la del objeto coche.


```
  ],
  "categories": [
   {
    "supercategory": "none",
    "id": 0,
```

```
    "name": "car"
  }
```


Con esto tenemos el fichero *json* que junto a las imágenes conforma un *dataset* completo y listo para ser usado en el entrenamiento de una nueva red Detectron2.

3.7. DIVISIÓN DEL *DATASET*

Para el correcto funcionamiento de cualquier red neuronal es necesario dividir su *dataset* para poder tanto entrenar como probar la red.

La división más común es usar un 80% del *dataset* para el entrenamiento de la red y el resto para las fases de validación y test.

Figura 24: División típica de un dataset

La división usada para el *dataset* creado para este proyecto ha sido el uso de 54 imágenes para test, otras 54 para validación y el resto (432) para entrenamiento.

3.7.1. *DATASET* DE ENTRENAMIENTO

El conjunto de entrenamiento cuenta con 432 imágenes térmicas etiquetadas. Los datos de este conjunto serán los usados para que el sistema aprenda durante el entrenamiento de la red. Por esta razón este conjunto es mucho más numeroso que el resto.

El objetivo del dataset es que el sistema, después de la fase de entrenamiento, sea capaz de detectar coches en imágenes térmicas con un alto grado de fiabilidad.

3.7.2. *DATASET* DE VALIDACIÓN

Este conjunto se usa para ajustar algunos hiper-parametros del modelo entrenado previamente. Queremos entrenar a la red neuronal con ciertos hiper-parametros, ver el resultado en el set de validación, cambiar los valores de los hiper-parametros e ir comparando resultados hasta encontrar los hiper-parametros que sean buenos resolviendo el problema.

Los datos de validación no sirven para el aprendizaje de la red como tal, pero sí que ayudan a que el aprendizaje sea mejor, por lo que es un conjunto muy importante para un correcto aprendizaje de la red.

3.7.3. *DATASET* DE TEST

Este conjunto de datos se utiliza para comprobar el funcionamiento del modelo previamente entrenado. Nos ayuda a sacar conclusiones sobre la precisión y el rendimiento de nuestra red.

Este conjunto no es necesario que sea excesivamente numeroso, pero tiene que tener la extensión necesaria para probar la red en diferentes condiciones y así poder comprobar cómo se comporta.

3.8. ENTRENAMIENTO DE LA RED NEURONAL

Con el dataset ya etiquetado el siguiente paso era entrenar la nueva red neuronal para que pueda aprender a detectar coches en imágenes en el espectro térmico.

Tras programar y preparar todo para el entrenamiento de la red, al iniciarlo surgió un problema. La librería CUDA funcionaba perfectamente en las fases anteriores de reconocimiento y tratamiento de creación del dataset. Sin embargo, daba un error durante la ejecución de la fase de entrenamiento.

Analizando el problema se descubrió como un problema de incompatibilidad relacionado con la combinación exacta de versiones de los módulos y librerías empleados en el entorno, junto al hecho de que el modelo de entrenamiento contaba con 80 posibles clases de objetos mientras que nuestro dataset sólo proporcionaba un tipo de objeto, el tipo “coche”.

El sistema podía realizar el entrenamiento, pero sólo si se desactivaban los cálculos en la GPU, pasando estos a ser realizados únicamente con la CPU.

El hecho de no poder usar la GPU para el entrenamiento era un gran contratiempo, ya que el uso de la CPU es mucho más lento. Una sesión de entrenamiento de menos de dos horas haciendo uso de la GPU se convertía en una sesión de más de 12 horas usando únicamente la CPU para los cálculos.

Se realizaron pruebas de entrenamiento con versiones reducidas del dataset, con un número bajo imágenes y el proceso funcionaba, validando la funcionalidad del código desarrollado. Pero claramente, al tener un dataset tan corto la fiabilidad del reconocimiento posterior era reducida.

Otra opción probada para reducir este tiempo de entrenamiento con la CPU fue decrementar el número de iteraciones que hace el sistema para el entrenamiento, pero esto, lógicamente, también afectaba negativamente a la precisión de la detección de la red.

La tercera solución que se barajó y finalmente se eligió fue la de usar una herramienta como Google Colab, con la que poder realizar el entrenamiento con una GPU en sus servidores de forma remota.

Puesto que no podíamos permitirnos la limitación de 12 horas de uso de las GPUs remotas cada 48 horas para el desarrollo del proyecto, se procedió a contratar un mes de Colab pro, el cual da acceso ilimitado a las GPU además de acceso a GPUs más potentes.

Gracias a esto también pudimos recortar el tiempo de entrenamiento en 1 hora con respecto a la versión gratuita, pasamos de 2 horas y media a 1 hora y media. Por supuesto mucho mejor que el entrenamiento sobre la CPU local.

Para proceder con el entrenamiento el primer paso fue registrar, en el sistema Detectron2, los datasets de entrenamiento y validación. Este registro básicamente consiste en desarrollar el código necesario para hacerle llegar al sistema neuronal, de una manera estructurada, donde están los ficheros a los que necesitará acceder en la fase de entrenamiento, tanto los ficheros de imágenes como el fichero *json* que contiene toda la información estructurada de los objetos contenidos en las imágenes.

Con estos datasets registrados el siguiente paso es definir los parámetros de entrenamiento. El parámetro más importante que hay que ajustar es el número de iteraciones que hará la red en su entrenamiento. Es

importante puesto que si te quedas corto la red no habrá aprendido suficiente para desempeñar su trabajo, pero si nos pasamos aparecerá el sobreentrenamiento u *overfit*.

El principal problema del *overfit* es que, aunque aparentemente las métricas indican una tasa baja de errores, el sistema se acostumbra y ajusta tanto a las imágenes del dataset que se le hace imposible generalizar y por lo tanto en un entorno real no funciona.

En nuestro caso para la elegir el número de iteraciones a usar se probaron diferentes casos y el que mejor resultados dio fue usar 1500 iteraciones, de las cuales las primeras 1000 serán iteraciones de *warmup*. Este tipo de iteraciones sirven para crear el esqueleto del aprendizaje, mientras que las ultimas 500 son para purificar los detalles. [24]

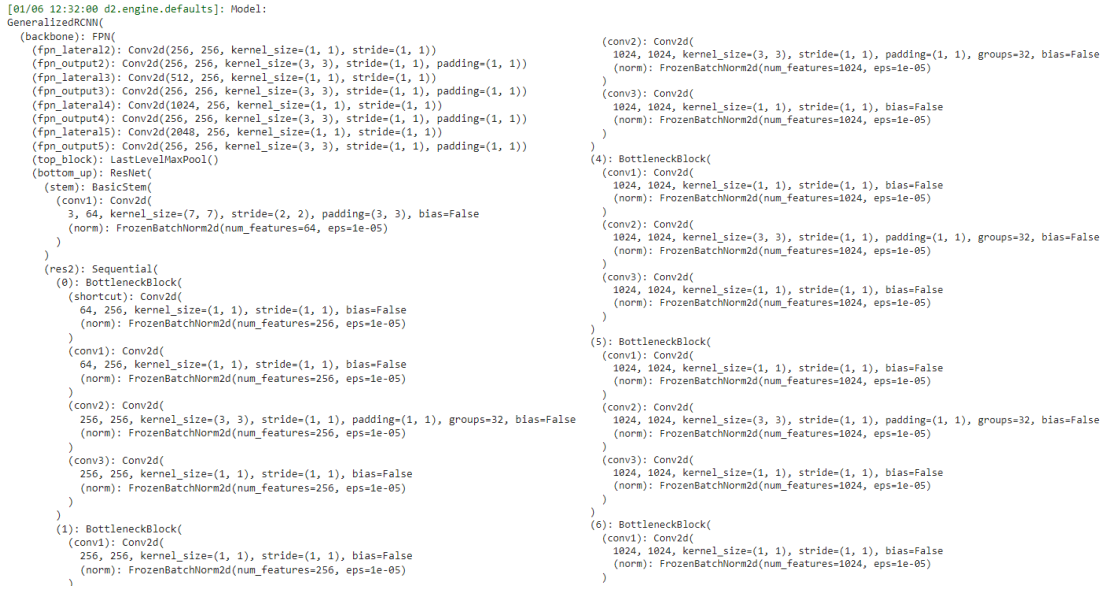

Figura 25: Captura de parte del modelo de la red

Como se ve en la figura anterior, cuando iniciamos el entrenamiento se nos muestra el modelo de la red con sus capas y las convoluciones de cada una de estas. Además de esto, se nos muestra tanto el tamaño del *kernel* de cada convolución como su *padding*.

Como ya quedó explicado en el apartado 2.2.8., el *kernel* es el filtro con el que se obtienen las características de las imágenes y por su parte el *padding* se usa cuando hay que pasar el *kernel* por las esquinas y bordes de las imágenes, haciendo que las zonas del *kernel* que quedan por fuera de la imagen pasen a valer 0.

El entrenamiento de la red tuvo una duración de 1h y 36min.

```
[01/06 13:38:30 d2.utils.events]:  eta: 0:28:38  iter: 1039  total_loss: 0.2945  loss_cls: 0.06956  loss_box_reg: 0.2125  loss_rpn_cls: 0.008621  loss_rpn_loc: 0.005379  time: 3.8339  data_tim
[01/06 13:39:44 d2.utils.events]:  eta: 0:27:23  iter: 1059  total_loss: 0.3211  loss_cls: 0.09356  loss_box_reg: 0.21  loss_rpn_cls: 0.00638  loss_rpn_loc: 0.004108  time: 3.8314  data_time: 
[01/06 13:41:02 d2.utils.events]:  eta: 0:26:09  iter: 1079  total_loss: 0.3594  loss_cls: 0.1009  loss_box_reg: 0.2376  loss_rpn_cls: 0.003822  loss_rpn_loc: 0.003549  time: 3.8327  data_time
[01/06 13:42:20 d2.utils.events]:  eta: 0:24:54  iter: 1099  total_loss: 0.3278  loss_cls: 0.07874  loss_box_reg: 0.2193  loss_rpn_cls: 0.005559  loss_rpn_loc: 0.004522  time: 3.8332  data_tim
[01/06 13:43:36 d2.utils.events]:  eta: 0:23:40  iter: 1119  total_loss: 0.2454  loss_cls: 0.05794  loss_box_reg: 0.1953  loss_rpn_cls: 0.003811  loss_rpn_loc: 0.003786  time: 3.8329  data_tim
[01/06 13:44:51 d2.utils.events]:  eta: 0:22:25  iter: 1139  total_loss: 0.3332  loss_cls: 0.1018  loss_box_reg: 0.2197  loss_rpn_cls: 0.008321  loss_rpn_loc: 0.006104  time: 3.8312  data_time
[01/06 13:46:03 d2.utils.events]:  eta: 0:21:10  iter: 1159  total_loss: 0.2995  loss_cls: 0.07135  loss_box_reg: 0.2246  loss_rpn_cls: 0.006104  loss_rpn_loc: 0.003891  time: 3.8273  data_tim
[01/06 13:47:20 d2.utils.events]:  eta: 0:19:56  iter: 1179  total_loss: 0.3325  loss_cls: 0.1029  loss_box_reg: 0.206  loss_rpn_cls: 0.01173  loss_rpn_loc: 0.00604  time: 3.8276  data_time: 0
[01/06 13:48:37 d2.utils.events]:  eta: 0:18:41  iter: 1199  total_loss: 0.3411  loss_cls: 0.09062  loss_box_reg: 0.2554  loss_rpn_cls: 0.00833  loss_rpn_loc: 0.005497  time: 3.8282  data_time
[01/06 13:49:53 d2.utils.events]:  eta: 0:17:27  iter: 1219  total_loss: 0.296  loss_cls: 0.07815  loss_box_reg: 0.2113  loss_rpn_cls: 0.003355  loss_rpn_loc: 0.005178  time: 3.8279  data_time
[01/06 13:51:12 d2.utils.events]:  eta: 0:16:12  iter: 1239  total_loss: 0.3503  loss_cls: 0.1005  loss_box_reg: 0.2134  loss_rpn_cls: 0.009043  loss_rpn_loc: 0.004429  time: 3.8294  data_time
[01/06 13:52:29 d2.utils.events]:  eta: 0:14:58  iter: 1259  total_loss: 0.3268  loss_cls: 0.07561  loss_box_reg: 0.2439  loss_rpn_cls: 0.005315  loss_rpn_loc: 0.004677  time: 3.8296  data_tim
[01/06 13:53:47 d2.utils.events]:  eta: 0:13:43  iter: 1279  total_loss: 0.3143  loss_cls: 0.08625  loss_box_reg: 0.2205  loss_rpn_cls: 0.005384  loss_rpn_loc: 0.004134  time: 3.8308  data_tim
[01/06 13:55:03 d2.utils.events]:  eta: 0:12:29  iter: 1299  total_loss: 0.2606  loss_cls: 0.0548  loss_box_reg: 0.1953  loss_rpn_cls: 0.002584  loss_rpn_loc: 0.003718  time: 3.8308  data_time
[01/06 13:56:18 d2.utils.events]:  eta: 0:11:14  iter: 1319  total_loss: 0.2962  loss_cls: 0.0695  loss_box_reg: 0.2163  loss_rpn_cls: 0.005211  loss_rpn_loc: 0.00443  time: 3.8292  data_time:
[01/06 13:57:34 d2.utils.events]:  eta: 0:10:00  iter: 1339  total_loss: 0.3122  loss_cls: 0.0987  loss_box_reg: 0.22  loss_rpn_cls: 0.005268  loss_rpn_loc: 0.00479  time: 3.8287  data_time: 0
[01/06 13:58:50 d2.utils.events]:  eta: 0:08:45  iter: 1359  total_loss: 0.3362  loss_cls: 0.09779  loss_box_reg: 0.2189  loss_rpn_cls: 0.01209  loss_rpn_loc: 0.004655  time: 3.8284  data_time
[01/06 14:00:08 d2.utils.events]:  eta: 0:07:31  iter: 1379  total_loss: 0.345  loss_cls: 0.1031  loss_box_reg: 0.2414  loss_rpn_cls: 0.0119  loss_rpn_loc: 0.006086  time: 3.8294  data_time: 0
[01/06 14:01:25 d2.utils.events]:  eta: 0:06:16  iter: 1399  total_loss: 0.3615  loss_cls: 0.07849  loss_box_reg: 0.2432  loss_rpn_cls: 0.00398  loss_rpn_loc: 0.004405  time: 3.8296  data_time
[01/06 14:02:44 d2.utils.events]:  eta: 0:05:01  iter: 1419  total_loss: 0.4852  loss_cls: 0.1459  loss_box_reg: 0.2623  loss_rpn_cls: 0.02072  loss_rpn_loc: 0.007913  time: 3.8315  data_time:
[01/06 14:04:04 d2.utils.events]:  eta: 0:03:47  iter: 1439  total_loss: 0.2832  loss_cls: 0.06745  loss_box_reg: 0.2118  loss_rpn_cls: 0.00737  loss_rpn_loc: 0.005077  time: 3.8338  data_time
[01/06 14:05:22 d2.utils.events]:  eta: 0:02:32  iter: 1459  total_loss: 0.2747  loss_cls: 0.05713  loss_box_reg: 0.1998  loss_rpn_cls: 0.002071  loss_rpn_loc: 0.003005  time: 3.8346  data_tim
[01/06 14:06:36 d2.utils.events]:  eta: 0:01:18  iter: 1479  total_loss: 0.3566  loss_cls: 0.06926  loss_box_reg: 0.2477  loss_rpn_cls: 0.004237  loss_rpn_loc: 0.004424  time: 3.8327  data_tim
```

Figura 26: Captura de pantalla del entrenamiento

Una vez terminado el entrenamiento, Detectron2 nos proporciona toda una serie de datos que aportan información sobre la evolución del proceso de entrenamiento, así como la calidad o fiabilidad previstas.

Para analizar estos datos lo mejor es una representación gráfica. Para ello disponemos de la herramienta Tensorboard que nos crea una ventana dentro del notebook de Colab.

En esta ventana podremos visualizar las diferentes gráficas de precisión del reconocimiento, evolución de los falsos negativos a lo largo del entrenamiento, la evolución del factor de pérdida, etc. En un próximo apartado dedicado a Resultados entraremos en más detalle sobre el significado de estas gráficas.

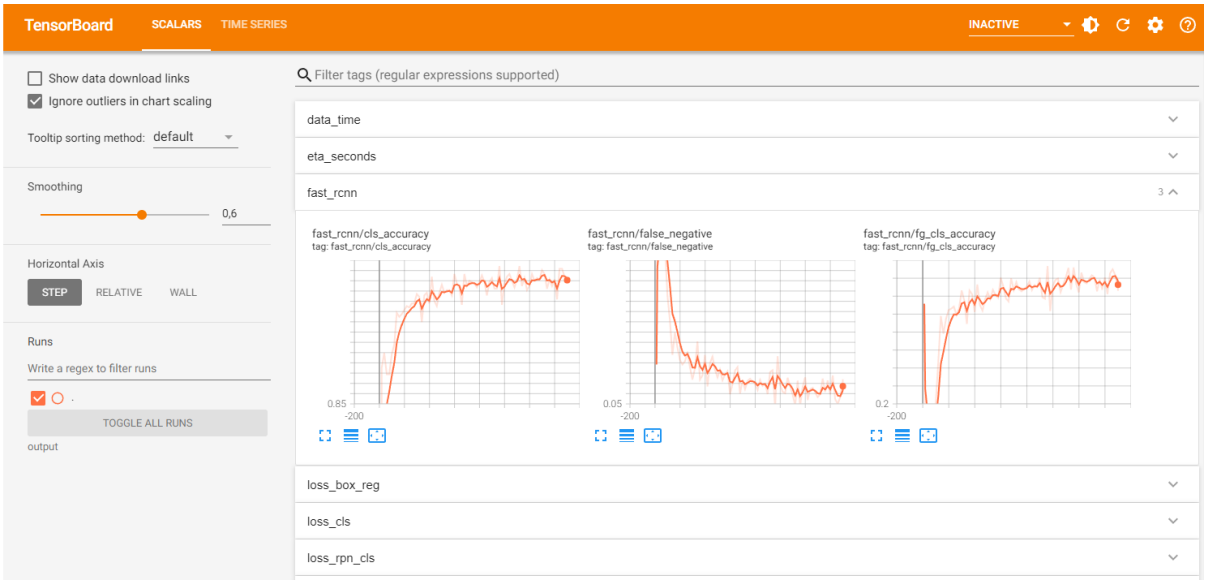

Figura 27: Captura del tensorboard

3.9. PRUEBA DE LA RED

Con la red ya entrenada el siguiente paso fue probarla con los datos del dataset de prueba. Este conjunto consiste en una serie de 54 imágenes que no habían sido procesadas por el sistema, a las que se añadieron unas 20 imágenes más en las que no se habían detectado coches en el espectro visible.

Para ello simplemente introducimos las imágenes térmicas reservadas para las pruebas en la red y analizamos los resultados de su reconocimiento con el nuevo entorno creado en el apartado anterior.

Para poder visualizar mejor los resultados se decidió pintar la caja predicha por el sistema para las imágenes térmicas, también en las imágenes del espectro visible.

Gracias a esto es mucho más fácil comprobar si el resultado obtenido es bueno o no, ya que el ojo humano está más acostumbrado a este espectro que al térmico.

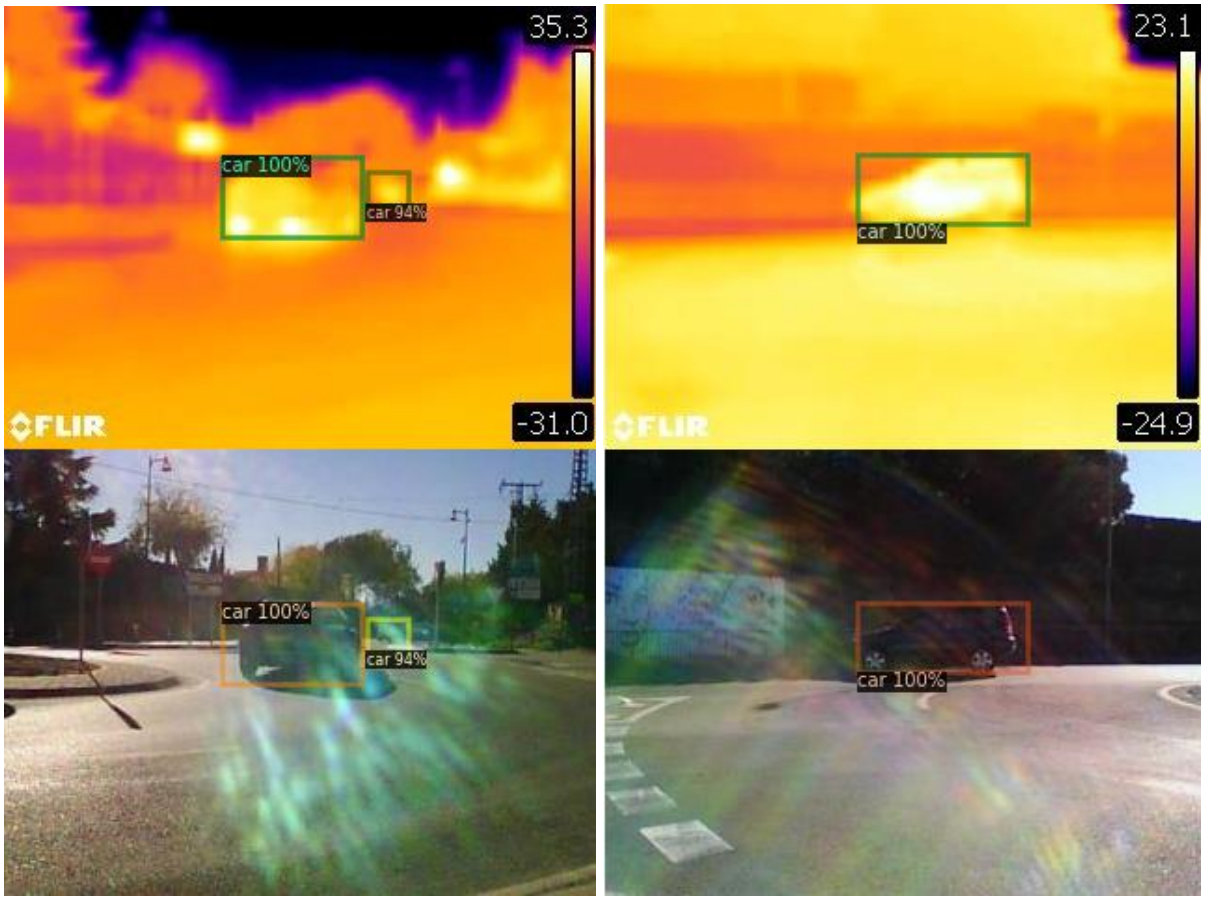

Figura 28: 2 ejemplos del test del sistema

El sistema tardó unos 3 minutos en procesar todas las imágenes del conjunto. La revisión visual del reconocimiento de imágenes fue muy satisfactoria, logrando reconocer en las imágenes térmicas la práctica totalidad de los vehículos, incluso aquellos que no habían sido detectados en el espectro visible por la mala calidad de la imagen original.

4. RESULTADOS

4.1. RESULTADOS DE EL ETIQUETADO DE IMÁGENES

El primer punto crítico de nuestro sistema es el proceso de etiquetado de las imágenes en el espectro térmico del *dataset*. Como ya hemos explicado con anterioridad, un buen etiquetado de las imágenes es crucial para un correcto funcionamiento de la red.

Como en nuestro caso este etiquetado lo realizamos de forma automática, a partir del reconocimiento de objetos en las imágenes en el espectro visible, es muy importante que los resultados de este reconocimiento sean buenos.

Para comprobar los resultados del etiquetado se ha realizado una revisión minuciosa de todas las imágenes etiquetadas.

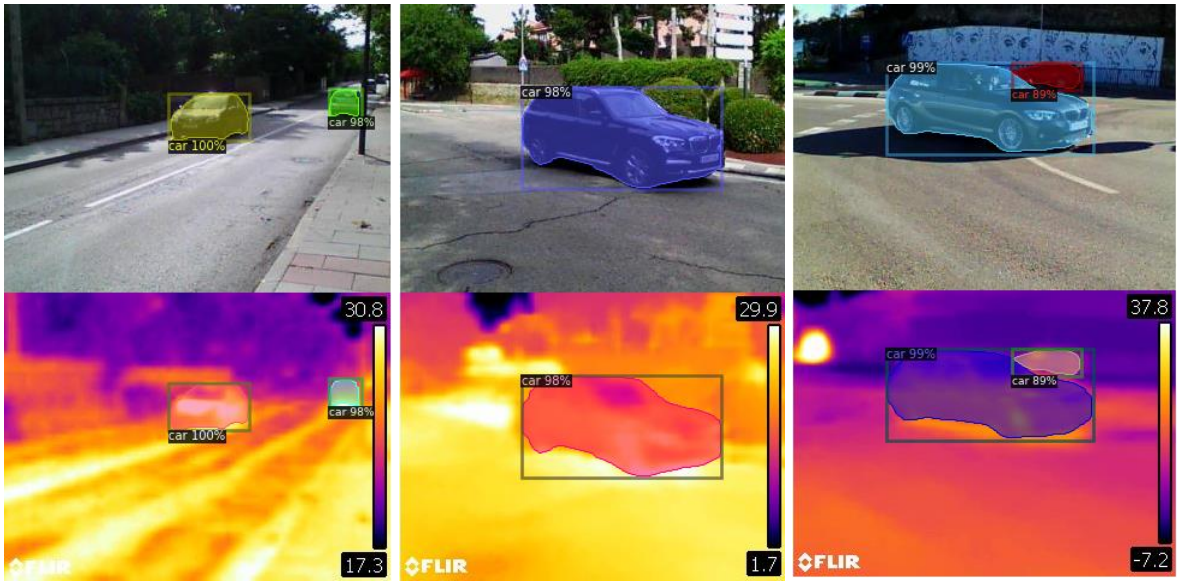

Figura 29: Ejemplos del etiquetado

En la primera fila de la figura anterior hemos marcado, sobre las imágenes del espectro visible, los marcos de los objetos reconocidos.

A efectos de comprobación visual, en la segunda fila hemos representado las imágenes térmicas correspondientes. Sobre éstas hemos superpuesto la información del reconocimiento de las imágenes estándar. Esta información, una vez tratada y estructurada, servirá para crear el dataset de imágenes térmicas de entrenamiento.

Los resultados observados en la revisión realizada fueron muy satisfactorios. Los errores de etiquetado (21 casos de un total de 431 imágenes) son en su totalidad falsos negativos y ninguno de ellos es un falso positivo. Es decir, en ningún caso se indicó como detectado un coche que no estuviera en la imagen, sin embargo, si se dio el caso de vehículos no detectados.

	Hay coche	No hay coche
Detecta un coche	409	0
No detecta un coche	21	2

Figura 30: Tabla de resultados

Estos errores de detección fueron causados en su totalidad por una mala calidad de la imagen, un reflejo provocado por tomar la imagen a contraluz o una mala iluminación del coche fotografiado. De hecho, en alguno de los casos, a simple vista era imposible reconocer que en la fotografía hubiera un vehículo, aunque éste si se podía apreciar en la correspondiente imagen térmica. En la siguiente figura podemos apreciar unos ejemplos del tipo de situación en la que el sistema estándar fue incapaz de reconocer los vehículos en el espectro visible.

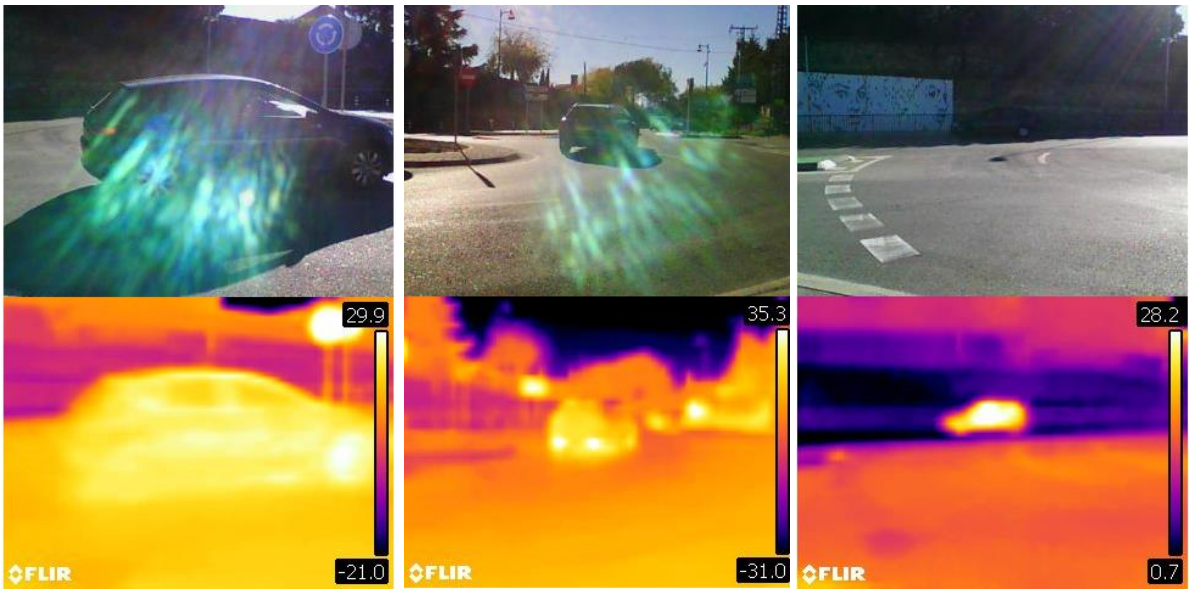

Figura 31: Algunos ejemplos de imágenes en las que no reconoce el coche

4.2. RESULTADOS DEL ENTRENAMIENTO DEL SISTEMA

Cuando realizamos el entrenamiento con el nuevo *dataset* de la red, Detectron2 nos proporciona distintas métricas sobre cómo ha sido el aprendizaje del sistema. Estas métricas nos dan información como la precisión de la red, los falsos negativos o la función de perdida entre otras cosas.

Para representar estas métricas en gráficos utilizamos la herramienta de Tensorboard, la cual nos representa en graficas todos los datos proporcionados por el entrenamiento.

Estas gráficas suelen tener como eje x el número de iteraciones del entrenamiento. Gracias a esto podemos observar cómo ha ido el entrenamiento según las iteraciones que se han hecho y ver si hay que aumentarlas o disminuirlas. En el caso en el que veamos que el sistema no ha aprendido suficiente tendremos que aumentar el número de iteraciones y en el caso en el que se observe que se produce un sobre-entrenamiento u *overfit* habrá que disminuirlas.

La primera grafica importante es la relativa a la precisión de la red, en la cual se puede observar como a medida que se realizan iteraciones va mejorando la precisión. La red comienza teniendo una precisión muy baja y sube rápidamente hasta un 93-94%, después, a medida que se realizan más iteraciones, crece poco a poco hasta alcanzar una precisión de 98%.

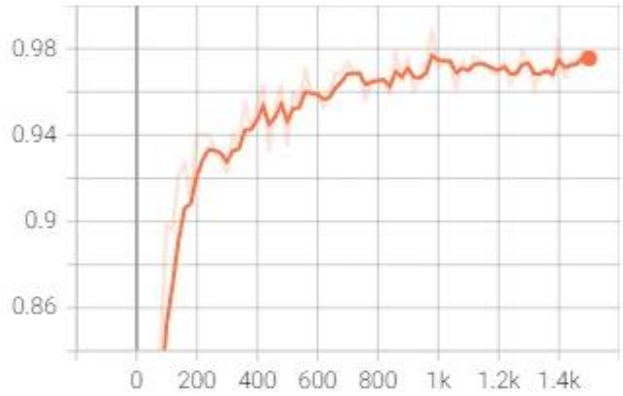

Figura 32: Captura de la gráfica de precisión de la red

La siguiente gráfica interesante es la que representa los falsos negativos. Podemos observar cómo empiezan en 0 y aumentan muy rápidamente para después descender con el paso de las iteraciones. La explicación de esto es que al principio al no detectar nada tenemos 0 falsos negativos, cuando empezamos a detectar se disparan los falsos negativos y según se van realizando más y más iteraciones estos van bajando.

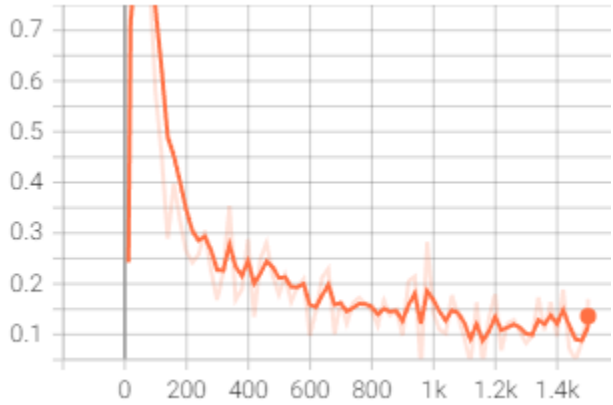

Figura 33: Captura de la gráfica de falsos negativos

Un objetivo principal del entrenamiento de una red es, como ya se mencionó en el estado del arte, reducir al mínimo posible la función de perdida. En la siguiente figura se puede apreciar como nuestra red va reduciendo esta función hasta llegar a valores por debajo de 0.05.

El comportamiento de esta red es justo el contrario que la de la precisión, empieza con una función de perdida muy alta, desciende rápidamente en las primeras 200 iteraciones hasta 0.2 y después va descendiendo poco a poco hasta alcanzar el valor mínimo de 0.05.

Figura 34: Captura de la gráfica de la función de pérdida

Por último, tenemos la gráfica que nos muestra el *learning rate*, en la que podemos observar como el sistema tiene un aprendizaje lineal durante las primeras 1000 iteraciones, que son las de *warm up*, y después el ritmo de aprendizaje se reduce mucho. Esta reducción del ritmo de aprendizaje se hace para evitar que se produzca *overfit* en nuestro sistema.

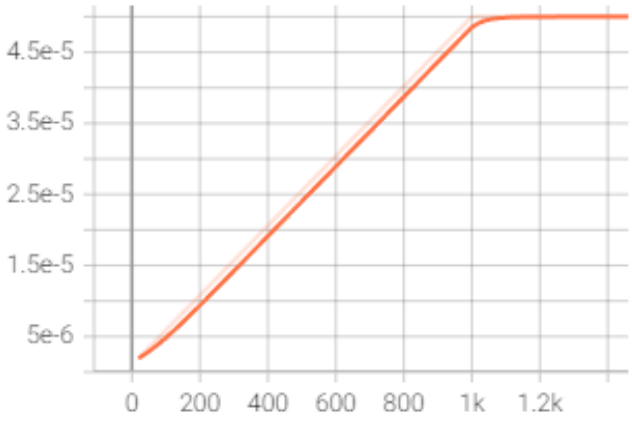

Figura 35: Captura de la gráfica del *learning rate* de la red

4.3. RESULTADOS DEL TEST

Aunque las métricas anteriores sobre el entrenamiento son importantes para saber cómo de bien funciona la red, la mejor forma de comprobarlo es poniendo el sistema a prueba con un conjunto de imágenes del espectro térmico y ver la fiabilidad a la hora de reconocer los coches.

Las imágenes que se le van a pasar al sistema son las correspondientes al dataset reservado para test, además de aquellas que se descartaron para el entrenamiento al no estar etiquetadas.

En todas las imágenes aparecen coches excepto en tres de ellas, estas se usan para ver si el sistema es capaz de devolver una imagen sin ninguna predicción al no encontrar lo que busca.

En la primera revisión de las imágenes del espectro térmico, con las cajas superpuestas correspondientes a los objetos detectados, parecía que algunas de estas cajas se habían predicho en lugares al azar de las imágenes. A simple vista era imposible determinar si realmente el objeto reconocido era o no un coche.

Para comprobar esto mejor, se decidió imprimir por pantalla, además de la imagen térmica con las cajas predichas, la imagen del espectro visible correspondiente con las mismas cajas.

Gracias a esto se pudo comprobar que las cajas que parecían pintadas al azar sí que eran coches, los cuales, a simple vista, en el espectro térmico eran muy difíciles de identificar.

Figura 36: 6 ejemplos de la detección en imágenes del espectro térmico

Los resultados observados son muy satisfactorios, ya que es capaz de reconocer los coches incluso cuando el ojo humano es incapaz. Pero no todo es perfecto y también tiene imágenes en las que falla.

La mayor parte de los fallos se producen en las imágenes del dataset que fueron tomadas en verano. Esto se debe a que cuando las imágenes fueron tomadas la zona con más temperatura de la imagen era el asfalto, el cual tomaba tonos naranjas claros y amarillos, mientras que los coches aparecen en colores que tiran más al morado y azul, mientras que en el resto de las imágenes ocurre lo contrario. El problema viene porque estas imágenes corresponden a algo menos de un quinto de las imágenes totales del dataset, lo cual ha sido insuficiente para que el sistema aprendiera a detectar con alta fiabilidad los coches, en este tipo de imágenes.

Esto se puede apreciar en la siguiente figura, el sistema es capaz de detectar el vehículo que esta justo en frente, pero no detecta el que está detrás, aunque a simple vista se puede distinguir la figura en la imagen del espectro térmico.

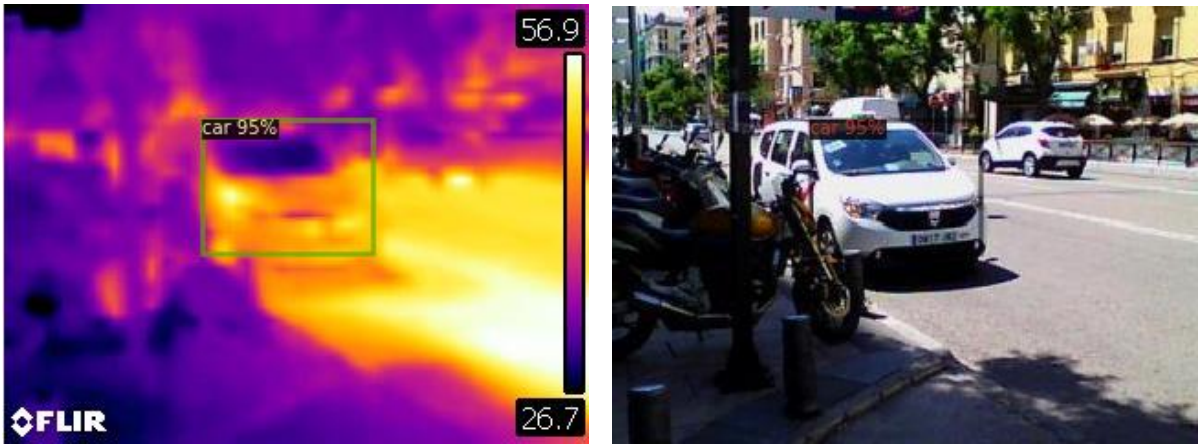

Figura 37: Ejemplo de la detección en imágenes de verano

El otro error destacable que se produce es que en un par de ocasiones el sistema detecta un único coche como dos distintos. En las dos ocasiones en que esto ocurre, los coches fotografiados están aparcados y han empezado a enfriarse, pero sus ruedas delanteras siguen calientes y son detectadas como parte de otro coche.

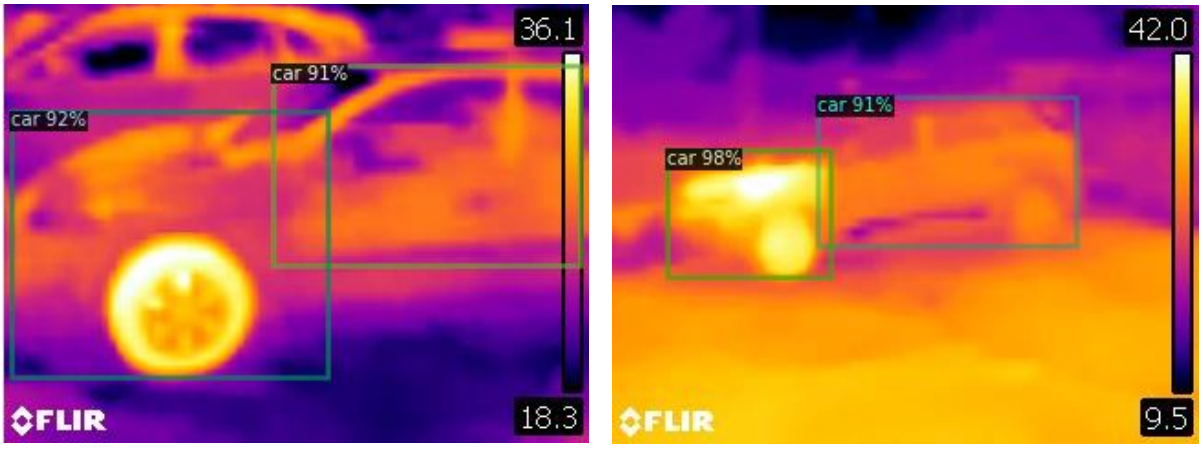

Figura 38: Ejemplos de fallo en la detección

Aunque el sistema falla en algunas ocasiones, por lo general los resultados son muy satisfactorios con algo más de un 90% de acierto. Además, como he mencionado antes, este porcentaje podría subir mucho si se ampliase un poco el *dataset* de entrenamiento con más imágenes tomadas en verano.

A veces se dan casos en que un coche no es reconocido en el espectro visual pero la misma imagen, en el espectro térmico sí que permite el reconocimiento. Por otro lado, la situación inversa también se da.

De hecho, este trabajo demuestra que un sistema capaz de reconocer en paralelo los objetos en los espectros visual y de infrarrojos incrementa considerablemente la fiabilidad en relación con los sistemas existentes de reconocimiento de imágenes.

5. CONCLUSIONES Y LÍNEAS FUTURAS

5.1. CONCLUSIONES

A partir de los resultados obtenidos, es posible sacar varias conclusiones sobre el trabajo aquí efectuado, siendo la primera el hecho de que se han logrado los objetivos principales del trabajo:

•Se ha creado un nuevo sistema de reconocimiento de imágenes infrarrojas que proporciona un alto grado de fiabilidad.

•Se ha desarrollado un sistema automatizado para la generación de conjuntos de datos anotados de objetos en imágenes térmicas.

Durante el desarrollo del proyecto se ha limitado el alcance del reconocimiento al caso concreto de vehículos, así, una vez resueltos los problemas y dificultades encontrados, es fácil e inmediato ampliar el ámbito del trabajo al reconocimiento de cualquier otro objeto (siempre que tenga una huella térmica distintiva).

Todos los subobjetivos definidos al principio del proyecto han sido alcanzados:

1.- Se preparó el entorno informático (hardware y software) adecuado para la ejecución del trabajo.

2.- Se generó el amplio conjunto de imágenes, en formatos visual y térmico, con la temática seleccionada. Este conjunto de imágenes se organizó y estructuró para su uso.

3.- Se creó un proceso automatizado para la generación automática de datasets anotados de imágenes térmicas.

4.- Se ejecutó el proceso de entrenamiento de un nuevo sistema de reconocimiento para su uso con imágenes en el espectro infrarrojo.

5.- Se probó el nuevo sistema de reconocimiento de imágenes térmicas, consiguiendo un alto grado de fiabilidad.

Otra conclusión que se puede obtener del trabajo es que se ha validado la posibilidad de obtener la información de objetos en imágenes mediante el espectro infrarrojo, lo cual es muy interesante en aquellos casos en que la información visual está limitada por circunstancias externas, como la falta de luz, el exceso de contraste en las imágenes o las obstrucciones a la visibilidad producidas por nieblas o humos.

Esto se puede apreciar en las imágenes de ejemplo que aparecían en la figura 1 de la introducción, donde un sistema de detección de objetos en el espectro visible era incapaz de detectar el coche que aparecía debido a la mala iluminación del objeto. Pero si se usa esa misma imagen, en su versión del espectro térmico, por el nuevo sistema, éste es capaz de detectar el vehículo sin problemas, como se ve en la siguiente figura.

Figura 39: Captura de la detección de un coche en ambos espectros

Otra importante conclusión del trabajo es que el sistema desarrollado es flexible: si preparamos nuevos conjuntos de imágenes de otras temáticas (tanto mono-objeto como para diferentes tipos de objetos) podremos utilizar los procedimientos aquí desarrollados para crear nuevos sistemas de reconocimiento de imágenes térmicas.

Lógicamente habría que ajustar los parámetros de entrenamiento y ejecutar una serie de pruebas de los resultados, pero todo esto sería una tarea mucho más sencilla que la puesta en marcha desde cero del proyecto.

Esta flexibilidad abre todo un mundo de posibilidades para desarrollar aplicaciones prácticas y soluciones empresariales en aquellos campos en los que se necesita aumentar al máximo la fiabilidad de la detección de cualquier tipo de objeto en cualquier condición de iluminación o visibilidad.

5.2. LINEAS FUTURAS

Este proyecto puede servir como la base para desarrollos más avanzados que incrementen las funcionalidades y capacidades de sistemas existentes.

En general en todos aquellos casos que sea conveniente o necesario incrementar la calidad de la detección de objetos en entornos en los que la iluminación o visibilidad de las imágenes en el espectro visible sea deficiente (por la noche o con mala visibilidad por causas meteorológicas o por humo pueden ser los casos más habituales).

Una primera línea de actuación que permitirá una mayor utilidad práctica en cualquier nuevo campo es la adaptación del sistema para pasar de la detección sobre imágenes estáticas a la detección de videos en espectro infrarrojo.

Otra de las potenciales líneas de evolución del proyecto es la simplificación de sus interfaces de uso, haciendo que esté disponible a través del uso de dispositivos móviles.

Algunos de los campos de trabajo donde estas características de reconocimiento en el espectro infrarrojo pueden ser más beneficiosas son las áreas militares y de seguridad.

Al hablar de proyectos futuros en el campo de la seguridad, se hace en un sentido amplio pues las utilidades pueden ser en segmentos tan variados como la seguridad del tráfico, los sistemas de conducción autónomos de vehículos, la videovigilancia "inteligente", informes para compañías de seguros, investigación policial, etc.

A continuación, se exponen algunos ejemplos de posibles desarrollos que pueden beneficiarse de la capacidad de reconocimiento en espectro infrarrojo para suplir las capacidades de los sistemas de detección "tradicionales".

6. BIBLIOGRAFÍA

[1] **ROUHIAINEN, LASSE**, Inteligencia artificial 101 cosas que debes saber hoy sobre nuestro futuro.

https://static0planetadelibroscom.cdnstatics.com/libros_contenido_extra/40/39308_Inteligencia_artificial.pdf

[2] **B12 ESPAÑA,** Qué es Big Data y cómo funciona.

https://agenciab12.com/noticia/que-es-big-data-como-funciona

[3] **APD,** Big data: ¿qué es y para qué sirve?

https://www.apd.es/big-data-que-es-y-para-que-sirve

[4] **FORBES,** Machine Learning Vs. Artificial Intelligence: How Are They Different?

https://www.forbes.com/sites/forbestechcouncil/2018/07/11/machine-learning-vs-artificial-intelligence-how-are-they-different/?sh=64375a3f3521

[5] **XIAN-DA ZHANG,** Machine Learning.

https://link.springer.com/chapter/10.1007/978-981-15-2770-8_6

[6] **DOT CSV**. 2018. ¿Qué es una Red Neuronal? Parte 1: La Neurona.

https://www.youtube.com/watch?v=MRIv2IwFTPg

[7] **DOT CSV**. 2018. ¿Qué es una Red Neuronal? Parte 2: La Red.

https://www.youtube.com/watch?v=uwbHOpp9xkc

[8] **DOT CSV**. 2018. ¿Qué es el Descenso del Gradiente? Algoritmo de Inteligencia Artificial. https://www.youtube.com/watch?v=A6FiCDoz8_4

[9] **HERNANDEZ GOMEZ, LUIS ALFONSO**. Apuntes asignatura 4 curso gitst.

[10] **APRENDE MACHINE LEARNING.** ¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador.

https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

[11] **PROGRAMADOR CLIC.** Notas de la red neuronal convolucional regional (R-CNN)

https://programmerclick.com/article/66931062622/

[12] **ICHI.PRO.** Detectron2: el tutorial básico de principio a fin.

https://ichi.pro/es/detectron2-el-tutorial-basico-de-principio-a-fin-99819572924643

[13] **PYTHON AWESOME.** Detectron2 for Document Layout Analysis.

https://pythonawesome.com/detectron2-for-document-layout-analysis/

[14] **DETECTRON2.** Getting Started with Detectron2

https://detectron2.readthedocs.io/en/latest/tutorials/getting_started.html

[15] **FACEBOOK.** DETECTRON2.IO.

https://detectron2.readthedocs.io/

[16] **PROGRAMADOR CLIC.** Introducción a Anaconda: Navigator, Spyder y Jupyter Notebook.

https://programmerclick.com/article/52111626118/

[17] **JUPYTER NOTEBOOK**. Documentos web para análisis de datos, código en vivo y mucho más

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/jupyter-notebook/

[18] **BARRIOS, JUAN.** Instalación de la Suite Anaconda Navigator

https://www.juanbarrios.com/wp-content/uploads/2020/11/Instalaci%C3%B3n-Anaconda-Navigator-.pdf

[19] **ADICTOS AL TRABAJO.** Google Colab: Python y Machine Learning en la nube.

https://www.adictosaltrabajo.com/2019/06/04/google-colab-python-y-machine-learning-en-la-nube/

[20] **GOOGLE.** Colaboratory.

https://research.google.com/colaboratory/intl/es/faq.html

[21] **FLIR**. Flir c3 datasheet.

[22] **FLIR.** FLIR Tools/Tools+ - Tutorials.

https://flir.custhelp.com/app/answers/detail/a_id/1568/~/flir-tools%2Ftools%2B---tutorials

[23] **GONZALEZ HERNANDEZ, MIGUEL.** Implementación de un sistema automático de medición de temperaturas basado en inteligencia artificial.

[24] **JDHAO'S BLOG.** Warmup in Maskrcnn-benchmark and how does it work?

https://jdhao.github.io/2020/08/14/warmup_maskrcnn_how_does_it_work/

[25] **TECHNICAL FRIDAYS.** Evaluation metrics for object detection and segmentation: Map.

https://kharshit.github.io/blog/2019/09/20/evaluation-metrics-for-object-detection-and-segmentation

[26] **STACK OVERFLOW.** What does mask r-cnn's AP, AP50, AP70 mean?

https://stackoverflow.com/questions/48457239/what-does-mask-r-cnns-ap-ap50-ap70-mean

[27] **LIN, TSUNG-YI, MICHAEL MAIRE, SERGE BELONGIE, LUBOMIR BOURDEV, ROSS GIRSHICK, JAMES HAYS, PIETRO PERONA, DEVA RAMANAN, C. LAWRENCE ZITNICK, AND PIOTR DOLLÁR.** 2014. Microsoft COCO: Common Objects in Context. http://arxiv.org/abs/1405.0312

[28] **APRENDE MACHINE LEARNING.** Sets de Entrenamiento, Test y Validación.

https://www.aprendemachinelearning.com/sets-de-entrenamiento-test-validacion-cruzada/

[29] **KAGGLE.** Object Detection with Detectron2 – PyTorch.

https://www.kaggle.com/maartenvandevelde/object-detection-with-detectron2-pytorch

[30] **TOWARDS DATA SCIENCE.** R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms.

https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

ANEXO A: ASPECTOS ÉTICOS, ECONÓMICOS, SOCIALES Y AMBIENTALES

A.1. INTRODUCCIÓN

Este proyecto permite sentar las bases y crear una prueba de concepto para ampliar, de una manera cómoda y eficiente, el ámbito de detección automática, basada en la Inteligencia Artificial, a los objetos detectables en el espectro infrarrojo.

Por tanto, este proyecto es la base para desarrollos más avanzados que incrementen las funcionalidades y capacidades de sistemas existentes, especialmente en temas relacionados con la seguridad.

Cuando se habla de proyectos futuros del campo de la seguridad, se hace en un sentido amplio pues las utilidades pueden ser en segmentos tan variados como la seguridad del tráfico, los sistemas de conducción autónomos de vehículos, la videovigilancia "inteligente", informes para compañías de seguros, investigación policial, etc.

Por tanto, aunque el alcance del presente proyecto no tenga especiales implicaciones en aspectos sociales, éticos o legales, lo cierto es que futuros desarrollos subsecuentes deberán plantearse estos asuntos, especialmente en todo lo relacionado con la protección de los derechos de privacidad y confidencialidad de las personas.

A.2. DESCRIPCIÓN DE IMPACTOS RELEVANTES RELACIONADOS CON EL PROYECTO

Como se indica en el apartado anterior, el presente proyecto plantea el desarrollo y validación de una prueba de concepto para la extensión de la detección automatizada ya existente en el campo del espectro visual a imágenes obtenidas en el espectro infrarrojo. Por tanto, los posibles impactos relevantes del proyecto, es aspectos éticos, sociales, económicos o ambientales habrían de ser analizados para esos desarrollos futuros.

En función del tipo de desarrollo, los impactos a analizar serían diferentes, pero es posible dar una visión no exhaustiva de potenciales impactos en función del tipo de soluciones. Aquí se analizan varios potenciales desarrollos futuros que pueden hacer uso de los conceptos desarrollados en este trabajo:

Soluciones de videovigilancia: Si se plantea un sistema mixto de videovigilancia con imágenes de espectro visible y térmicas, lógicamente los aspectos éticos, sociales y legales de protección de la intimidad y confidencialidad no cambiarían y sería necesario definir los aspectos legales y éticos que facultaran la utilización del sistema de acuerdo con la legislación nacional vigente, así como con la General Data Protection Regulation (GPDR) de la Unión Europea. Es decir, habría que definir los procesos para la obtención de permisos previos por parte de las personas identificadas o, en su caso, para la difusión de avisos o información sobre la captación y tratamiento de imágenes. También sería preciso definir el aspecto de las posibles autorizaciones administrativas para un uso legal del sistema

Sin embargo, si se desarrolla un sistema únicamente basado en el tratamiento de imágenes infrarrojas el tema puede simplificarse considerablemente puesto que no sería factible la identificación de personas concretas, ni la lectura de matrículas de vehículos, etc.

Sistemas de conducción autónomos de vehículos: Este tipo de sistemas, actualmente en desarrollo por parte de los grandes fabricantes de vehículos, necesitan soluciones en el campo de la visión infrarroja para asegurar la fiabilidad del sistema en caso de mala visibilidad.

Los aspectos legales, éticos, sociales, económicos o de sostenibilidad y lucha contra el cambio climático de estos sistemas son cruciales para su desarrollo y puesta en el mercado. El impacto de la conducción autónoma en el futuro de ciertas profesiones como el taxi o el transporte de mercancías puede ser total. El potencial cambio de paradigma de pasar a un entorno en el que predomine el coche compartido puede cambiar la estructura del transporte en las ciudades. Todo el modelo económico del transporte se verá afectado por estos sistemas.

Desarrollo de un sistema de localización de víctimas en derrumbes: La utilización de sistemas de reconocimiento de imágenes térmicas puede permitir la creación de un sistema, instalado sobre un pequeño dron capaz de moverse por espacios estrechos y angostos, entre los restos y los huecos producidos en un derrumbe. El impacto en localización de víctimas para su posterior rescate puede ser extraordinario salvando vidas humanas.

Sistemas de detección de incendios en zonas boscosas: Con el reconocimiento de imágenes térmicas se pueden mejorar los sistemas actuales de vigilancia contra incendios, reconociendo las zonas donde se producen, así como, potencialmente localizando a potenciales pirómanos.

Este tipo de soluciones tendrá un importante efecto de sostenibilidad y conservación del medio ambiente, sin olvidar los factores económicos relacionados con el ahorro de costes que se produce al eliminar los incendios con tiempo, cuando aún son pequeños, o al disuadir a posibles pirómanos.

ANEXO B: PRESUPUESTO ECONÓMICO

COSTE DE MANO DE OBRA (coste directo)	Horas	Precio/hora	Total
	330	15,38 €	**5.075,40 €**

COSTE DE RECURSOS MATERIALES (coste directo)	Precio de compra	Uso en meses	Amortización (en años)	Total
Ordenador personal (software incluido)	2.500,00 €	6	5	250,00 €
Cámara Flir C3	640€	6	5	6,40€

COSTE TOTAL DE RECURSOS MATERIALES	**256,40 €**

GASTOS GENERALES (costes indirectos)	15%	sobre CD	**799,77€**
BENEFICIO INDUSTRIAL	6%	sobre CD+CI	**367,89 €**

SUBTOTAL PRESUPUESTO		**6.499,46 €**
IVA APLICABLE	21%	**1.364,89 €**
TOTAL PRESUPUESTO		**7.864,35 €**