Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Detección y seguimiento de conectores mediante CNN, para el guiado de operarios en el proceso de ensamblado de satélites de la Agencia Espacial Europea

Autor: Ana Isabel Quirós Gámez
Tutor: Jesús Capitán Fernández
Tutor Externo: Eduardo Ferrera Cabanillas

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo de Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Detección y seguimiento de conectores mediante CNN, para el guiado de operarios en el proceso de ensamblado de satélites de la Agencia Espacial Europea

Autor:
Ana Isabel Quirós Gámez

Tutor:
Jesús Capitán Fernández
Profesor Contratado Doctor

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Tutor Externo:
Eduardo Ferrera Cabanillas
Jefe de la Unidad de Automatización y Robótica

Fundación Andaluza para el Desarrollo Aeroespacial.
Centro Avanzado de Tecnologías Aeroespaciales
(FADA-CATEC)

Sevilla, 2021

Proyecto Fin de Carrera: Detección y seguimiento de conectores mediante CNN, para el guiado de operarios en el proceso de ensamblado de satélites de la Agencia Espacial Europea

Autor: Ana Isabel Quirós Gámez

Tutor: Jesús Capitán Fernández

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha

A mi familia,

A todos a los que les pertenece un pedacito de mí.

Agradecimientos

Siendo breve y concisa, agradecer a todos los que, de alguna forma u otra, han contribuido en formar parte de quien soy. Agradecer siempre a mi familia, a mis padres, a mis hermanos, a mis abuelos, a mi tía, a mis amigos. A las personas que tanto me aportan y que tan ameno hicieron el camino.

A FADA-CATEC por la oportunidad de formarme, a Eduardo, y a todos mis compañeros, por contar conmigo y hacerme sentir como en casa.

A Jesús, por ser incondicional, y a María Jesús, por acompañarme en esta etapa trabajando siempre a una, mano a mano. A todo lo bonito que me llevo del camino y de estos años. A mis compañeros, a la familia que hemos formado. A mis maestros. Al esfuerzo. Al crecimiento personal. Y a mí.

Ana Isabel Quirós Gámez

Sevilla, 2021

Resumen

A lo largo del presente proyecto, se ha iniciado una línea de investigación enfocada al estudio y la puesta a punto de algoritmos de Inteligencia Artificial para la localización e identificación de conectores industriales. La finalidad de esta línea reside en integrar esta tecnología en los procedimientos de asistencia a operarios a la hora de ejecutar la conexión de cableado en estados finales del ensamblaje de los satélites de la Agencia Espacial Europea. Se espera que el uso de esta tecnología acelere la ejecución de las tareas, al disminuir las tasas de error y sus consiguientes retrabajos asociados. Este enfoque permitirá resaltar la importancia del desarrollo de las diferentes metodologías referentes a la localización en cuanto al soporte ofrecido al operario; apoyándose en el uso de las innovadoras técnicas de Deep Learning y las Redes Neuronales Convolucionales (CNN).

Para lograr este objetivo, durante el presente proyecto ha sido necesario la instalación y configuración de un dispositivo de captura de imagen industrial, además de cumplimentar numerosos requisitos de preparación de una estación capaz de procesar en tiempo real mediante el uso de GPU los diferentes procesos llevados a cabo. Dicha estación ha sido utilizada para entrenar el modelo de forma que pueda ser útil en estados posteriores para la asistencia en la conexión del mencionado cableado.

Con este propósito, mediante el entrenamiento de ciertos algoritmos de redes neuronales convolucionales, se ha logrado elaborar un detector con un elevado porcentaje de fiabilidad que satisface dicho objetivo. El proceso cuenta con numerosas labores, entre las que podemos destacar (i) la creación del conjunto de datos enfocado para labores de entrenamiento, (ii) el proceso de etiquetado de cada una de las imagen pertenecientes al mismo, (iii) la configuración de las diferentes capas de red para dotar al modelo pre-entrenado de capacidad para aprender a identificar nuevos objetos, (iv) la transferencia de conocimientos sobre una red pre-entrenada, capaz de detectar determinadas piezas aeronáuticas, (v) el entrenamiento en sí del modelo, (vi) la creación de un set de validación, y (vii) la evaluación sobre éste de los resultados del entrenamiento, además de la consideración de errores comunes durante dicho entrenamiento, y la selección de los pesos con una predicción más fiable.

Durante el proyecto, además, se ha desarrollado una aplicación en C++ que consiste en un visualizador en tiempo real, capaz de coordinar la clase identificada y la fiabilidad de acierto sobre la clase. La integración de esta tecnología supone una mejora en la experiencia de producción para el operario y para el éxito del programa en el que se encuentre encuadrado el satélite en cuestión. Esta línea de investigación cuenta con reconocido potencial, ya que, está previsto sea combinada con otros múltiples desarrollos de realidad aumentada.

Abstract

Within the present project, it has been developed a line of research focused on the study and implementation of Artificial Intelligence algorithms for the location and identification of industrial connectors. The purpose pursued resides in integrating this technology in procedures directed to operator's assistance, when executing the wiring connection in the final stages of the assembly of the satellites of the European Space Agency. Thus, it is expected, due to the development of this technology, that it will result in accelerating the production in the supply chain, by reducing error rates regarding the human factor. This approach evidences the importance of the development of different methodologies for inspection, tracking and location, making use of the innovative techniques of Deep Learning and the Convolutional Neural Networks (CNN).

To achieve this objective, it has been necessary to install and configure an industrial image capture device, in addition to cover numerous requirements for preparing a station capable of processing in real time through the use of GPUs for the different processes carried out. The station has been used to train the model so that it shall be useful in later states for assistance in the mentioned wiring connection.

For this purpose, ought to the convolutional neural network training, it has been possible to develop a detector with a high percentage of reliability that satisfies this objective. The process has consisted of numerous tasks, where it can be highlighted: (i) the creation of the data set focused on training tasks, (ii) the process of labeling each of the images belonging to it, (iii) the configuration of the different network layers to provide the model pre-trained ability to learn to identify new objects, (iv) the transfer of knowledge on a pre-trained network, capable of detecting certain aeronautical parts, (v) the training itself of the model, (vi) the creation of a validation set, and (vii) the evaluation on it of the training results, in addition to the consideration of common errors during said training, and the selection of the weights with a more reliable prediction.

Furthermore, a C ++ application has been developed. It consists of a real-time viewer capable of coordinating the identified class and the reliability of the class detection. The integration of this technology means an improvement in the production experience for the operator and for the success of the environment in which the satellite will be manufactured. This line of research has a recognized potential, since it can be combined with multiple other augmented reality developments.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Acrónimos

CNN	Convolutional Neural Network/ Red Neuronal Convolucional
ESA	European Space Agency/ Agencia Espacial Europea
LRO	Lunar Reconnaissance Orbiter/ Orbitador de Reconocimiento Lunar
UE	Unión Europea
GNSS	Global Navigation Satellite System / Sistema Global de Navegación por Satélite
FCC	Federal Communications Commission/ Comisión Federal de Comunicaciones
HOG	Histograma de Gradientes Orientados
YOLO	You Only Look Once
SVM	Support Vector Machines/ Máquina de Soporte Vectorial
WxH	Weight x Height
CCD	Charge-Coupled Device/ Dispositivo de Carga Acoplada
GPIO	General Purpose Input/Output, Entrada/Salida de Propósito General
HxWxL	Height x Width x Lenght / Alto x Ancho x Largo
E/S	Entrada/Salida
GVCP	GigE Vision Control Protocol
GVSP	GigE Vision Stream Protocol
IP	Internet Protocol
MS COCO	Microsoft Common Objects in Context
CPU	Central Processing Unit/ Unidad de Procesamiento Central
GPU	Graphics Processing Unit/ Unidad de Procesamiento Gráfico
mAP	Mean Average Precision
IoU	Intersection over Union

1 INTRODUCCIÓN

En 2020, la población mundial se estimaba en 6.070 millones de personas. A día de hoy, se espera que crezca hasta los 9.700 millones para 2050 [1]. La evolución y transformación social asociada a este incremento poblacional supone un desafío sin precedentes para el sector tecnológico, que se debe adaptar con rapidez para prestar servicio y soporte a la masa poblacional. En los últimos años, los avances en el campo de la ingeniería robótica han sido significativos y han servido para contribuir a mejorar procesos en cuanto productividad, flexibilidad y también calidad. En este contexto, la aportación de esta disciplina a cambios de paradigma tan relevantes como, por ejemplo, la revolución del modelo industrial hacia la Industria 4.0, ha sido especialmente notable. [2]

Estos momentos de precipitados cambios y profundas transformaciones en los que se encuentra sumergida la sociedad global están también marcados por la exploración espacial. Desde sus inicios en el año 1957 con la puesta en órbita del primer satélite artificial, el Sputnik 1, la investigación y el desarrollo en esta materia se han visto sometidos a unos adelantos que no conocían precedentes. Fruto de esos avances, se espera que muchas de las aspiraciones de la ciencia y la técnica se puedan hacer realidad en los próximos años, abriendo también la puerta a la explotación comercial del espacio exterior. Sin embargo, y pese a este extraordinario progreso, la construcción de los satélites, desde su diseño hasta puesta en marcha, sigue suponiendo un desafío para las capacidades técnicas de hoy y requieren de la implicación de numerosas disciplinas tecnológicas.

Este desafío se presenta de especial relevancia, al contextualizarse en la era aeroespacial que estamos viviendo:

- Por una parte, el auge del concepto de conectividad y los sistemas de posicionamiento están suponiendo un factor estratégico en el desarrollo de mercados en relación con la navegación por satélite; aumentando significativamente las inversiones en constelaciones como GALILEO. GALILEO es una iniciativa de radionavegación impulsada por la Agencia Espacial Europea en conjunto con la UE para conformar el Sistema Global de Navegación por Satélite (GNSS) Europeo, compatible con los existentes en la actualidad; el GPS de procedencia americana, el sistema BEIDOU desarrollado por China, o el GLONASS (de orígenes rusos), entre otros. El servicio desarrollado se encuentra desplegando una red de satélites que contará con un margen de error menor a un metro de exactitud, aportando a los usuarios un servicio de conocimiento sobre posicionamiento en tiempo real [3].
- Por otra parte, en el año 2017, Space X presentó ante la Comisión Federal de Comunicaciones (FCC), un proyecto consistente en una constelación de 12.000 pequeños satélites más repuestos espaciales, con perspectivas de lanzamiento para mediados de la década de 2020 [4]. El fin perseguido abarca la producción en masa de satélites de órbita terrestre baja comunicada con sus correspondientes transceptores terrestres; abordando la implementación de comunicaciones en la banda V y el consecuente acceso a Internet a bajo coste a escala mundial [5]. Esta iniciativa fue registrada bajo el nombre de 'Starlink' para la identificación de la constelación de la red de banda ancha satelital. Desde el momento, se han lanzado 1.737 satélites, en el intervalo de tiempo comprendido entre el 22 de febrero de 2018 y el 20 de junio de 2021, de los cuales 1635 se encuentran activos [6]

La industria relativa a este campo, por tanto, se encuentra inmersa en un crecimiento exponencial de la producción, surgiendo así la necesidad sobre la productividad de la cadena de suministros en el sector, incrementando el número de dispositivos fabricados frente a un mismo intervalo de tiempo.

En este entorno, la Inteligencia Artificial emerge como disciplina bandera entre otros motores de cambio, siendo una de las que experimenta un mayor desarrollo debido a las numerosas ventajas que ofrece respecto a la necesidad en el aumento del rendimiento relativo en el proceso de fabricación de satélites. Más concretamente, los últimos avances en el contexto de la Inteligencia Artificial han demostrado su eficacia en el ámbito del procesado de imágenes mediante la implementación computacional de algoritmos que imitan el funcionamiento biológico de las redes neuronales del cerebro humano, lo que resulta singularmente ventajoso en tareas de detección y extracción de características.

1.1 Motivación

En el marco previamente descrito, la Agencia Espacial Europea, ESA por sus siglas en inglés, se halla inmersa en la introducción, dentro de sus propios procesos, de sistemas y procedimientos que permitan garantizar sus elevados estándares de calidad a la vez que elevar su producción. Alguno de los procesos señalados para la implementación de esta metodología, se encuentran estrechamente ligados a los distintos estados de la fabricación de los satélites y, especialmente, su ensamblaje e integración final. No obstante, y dado que esta última fase representa dificultades de cara a la automatización del proceso, se espera conseguir para ella una producción totalmente asistida al operario que reduzca el número de errores, minimizando las posibilidades de que tenga lugar un fallo humano. La complejidad del proceso puede ser observada en la Figura 1-1, motivo por el cual, la ESA ha manifestado su interés en potenciar el uso de sistemas de realidad aumentada, capaces de prestar un servicio eficaz al operario durante el ensamblaje final del cableado del satélite.

Figura 1-1. Proceso de ensamblaje del Orbitador de Reconocimiento Lunar (LRO). Fuente [40]

1.2 Objetivos

El principal objetivo de este Trabajo de Fin de Grado es proporcionar un primer modelo para ese sistema de realidad aumentada que permita asistir la conexión del cableado. La función principal que debe cumplir el prototipo es servir como punto de partida para desarrollos más profundos y complejos.

Por ese mismo motivo, este TFG toma la forma de un demostrador de tecnología que permite la identificación en tiempo real de diversos conectores comerciales sobre una mesa de trabajo ideada a semejanza del proceso real de fabricación y conexionado de satélites. En el mismo se implementan algoritmos de Redes Neuronales Convolucionales, a fin de dotar al modelo de flexibilidad y adaptabilidad para el uso futuro de conectores más complejos propios de la industria espacial. Por su parte, el uso de la mesa está justificado dada la naturaleza del propio proceso relativo al manejo del cableado durante el ensamblado de satélites, que suele caracterizarse por el empleo de superficies planas como espacios de trabajo, tal y como se puede ver en la Figura 1-2.

Figura 1-2. Proceso de fabricación del Satélite Galileo. Fuente [7]

1.3 Estructuración del proyecto

Generalmente, y con el fin de abordar una primera aproximación que cumpla con las necesidades de la Agencia Espacial Europea, se detalla el proceso de desarrollo del diseño del Sistema de Detección de Conectores para la prestación de asistencia al operario, gracias al uso de sistemas de realidad aumentada durante el ensamblaje final del cableado del satélite.

A tal efecto, el documento se estructurará atendiendo a diferentes secciones. La primera de ella consiste en la introducción, objetivos alcanzables, motivación y justificación sobre las necesidades reales de la Industria entorno al desarrollo del producto, y detalle sobre la estructuración del documento. La segunda sección, será enfocada hacia la revisión del estado del arte de las herramientas de IA y los métodos de detección existentes en el mercado, realizándose un estudio teórico sobre las diferentes opciones disponibles y conceptos relativos al campo de Deep Learning y Redes Neuronales.

Tras detallar el preámbulo teórico, se especifica en una tercera fase, la metodología, arquitectura y tecnología empleada, tanto para el desarrollo de hardware como software de la herramienta de identificación y seguimiento.

A lo largo de cuarta sección relativa a la implementación de los conceptos teóricos y la metodología, se describirán las diferentes soluciones llevadas a cabo para cohesionar y unificar las diferentes tecnologías. Durante este epígrafe, será objeto de desarrollo, tanto la aplicación en C++ del software relativo a la plataforma de escáner BB-500GE, como la descripción detallada de la casuística relacionada con el entrenamiento de la red neuronal profunda, incluyendo los procesos de etiquetado del set de datos, transferencia de conocimientos, y una breve aproximación de los siguientes puntos abordados en el capítulo relativo del banco de pruebas sobre el modelo y uso posterior.

Sobre la quita sección, se enumerarán las diferentes métricas utilizadas para el cálculo de la fiabilidad del modelo. Se estudiarán, además, los costes computacionales consumidos durante el entrenamiento, y serán objeto de análisis los resultados de los pesos calculados a través de la evaluación de los parámetros mencionados. Además de ser evaluado, el modelo será mejorado mediante diversas técnicas. Se validará por tanto el sistema de detección diseñado, y los tiempos empleados en la detección.

El sexto apartado, abarcará lo relativo al desarrollo de una interfaz gráfica para la asistencia del operario durante el procedimiento de ensamblado del satélite. Finalmente, serán sugeridas líneas de mejoras del demostrador, y conclusiones sobre resultados alcanzados, además de la viabilidad del producto en el marco de la Industria 4.0 y la repercusión sobre el desarrollo de algoritmos inteligentes en la sociedad.

2 ESTADO DEL ARTE

A lo largo de este capítulo, serán objeto de estudio aquellos fundamentos teóricos relativos a las Redes Neuronales Profundas (Deep Learning), con el fin de familiarizar al lector con conceptos específicos relativos a este campo. Se ofrece así una revisión sobre el paradigma actual entorno al que se desarrolla esta tecnología, y las herramientas utilizadas a la hora de cimentar las bases sobre las que se construye esta solución.

En primer lugar, será llevada a cabo una revisión bibliográfica sobre los métodos de detección en Inteligencia Artificial, diferenciando los métodos de detección basados en modelo y los basados en aprendizaje. Para el desarrollo del detector para el guiado de operarios en el proceso de ensamblado, serán de aplicación estos últimos, de forma que se justificará el uso de los algoritmos basados en Deep Learning para esta labor, exponiendo sus ventajas frente a otros métodos de clasificación y seguimiento. Por tanto, será el eje central la investigación conceptos relacionados con el Aprendizaje Profundo, centrando los consiguientes epígrafes en los principales aspectos teóricos relativos a las Redes Neuronales Convolucionales.

Además de ello, se incluirán especificaciones sobre la implementación de estas arquitecturas, las CNN pre-entrenadas de las que se harán uso, y conceptos relativos al entrenamiento dentro del framework seleccionado. Serán objeto de estudio a su vez la evaluación sobre los modelos obtenidos durante el entrenamiento, anotando ciertos conceptos sobre errores relativos y las métricas que se llevarán a cabo para determinar la fiabilidad.

2.1 Revisión bibliográfica sobre diferentes métodos de detección de Inteligencia Artificial.

2.1.1 Métodos de detección basados en modelo.

Generalmente, los métodos de detección basados en modelo contemplan un enfoque intuitivo y directo. Los algoritmos que siguen esta línea de investigación se focalizan en características visibles del objeto a identificar, analizando la información del color o la forma de los elementos, comparando ciertos umbrales heurísticos para la localización del elemento. A pesar de la eficiencia demostrada a lo largo de los años, este método ha sido mejorado mediante técnicas de detección basadas en aprendizaje, las cuales están dotadas de capacidad de detección con independencia de tonalidades, además de contar con un aumento de viabilidad para su implementación debido a la incorporación de aprendizaje para la extracción de características no detectables dentro del campo de visión humano.

2.1.2 Métodos de detección basados en aprendizaje.

Los métodos de detección basados en aprendizaje serán los seleccionados para el desarrollo del detector. Los algoritmos que implementan este método son capaces de deducir una función gracias a un conjunto de datos sobre los que son entrenados.

Por un lado, se contemplan diferentes posibilidades que compongan soluciones robustas ante cambios de iluminación y luminosidad debido a la volatilidad de entornos. Es destacado HOG, histograma de gradientes orientados, junto al clasificador para el entrenamiento SVM, conocido como máquina de soporte vectorial. Este algoritmo utilizará para la detección la orientación del gradiente en áreas locales del fotograma.

También podría ser de aplicación la eficiencia del detector en visión computada Integral Channel Features (ICF), también conocido como ChnFtrs, que hace uso de imágenes integrales para caracterizar datos mediante sumas locales, histogramas de diferentes canales y el cual utiliza imágenes integrales para extraer datos como sumas locales o histogramas de distintos canales.

A pesar de ofrecer ambos unas estadísticas favorables en la detección e identificación de objetos, se decidió el uso de las Redes Neuronales Convolucionales (CNNs) debido a la amplia cabida en el mercado y a los

numerosos avances que ha ido experimentando a lo largo de su relativamente corta vida [8]. Los resultados obtenidos gracias a esta técnica han sido reconocidos y extendidos hasta el marco de clasificación en tiempo real, lo cual supone una coyuntura ideal para los exigentes requisitos de la Agencia Espacial Europea.

En los siguientes epígrafes, se extenderá la revisión bibliográfica proyectada entorno a esta línea de investigación.

2.2 Deep Leaning

2.2.1 Redes Neuronales Convolucionales

Las Redes Neuronales, como su nombre indica, componen estructuras computacionales que imitan el funcionamiento del sistema neuronal biológico. La constitución de la red consiste en un determinado número de neuronas, conformando numerosas conexiones entre ellas, organizadas en capas. Cada neurona recibirá un parámetro de entrada y entregará una determinada salida, que se procesará gracias a ciertos parámetros internos llamados pesos sinápticos. A tal salida, le será aplicada una Función de Activación como se puede observar en la Figura 2-1. En este modelado simple de neurona, la salida Y se rige por la ecuación:

$$Y = f(\sum_{i=1}^{n} w_i \, x_i)$$

Donde:

Y: conforma la salida del sistema

x: representa las entradas del mismo

w: consiste en los pesos sinápticos asociados a cada entrada

f: compone la función de activación

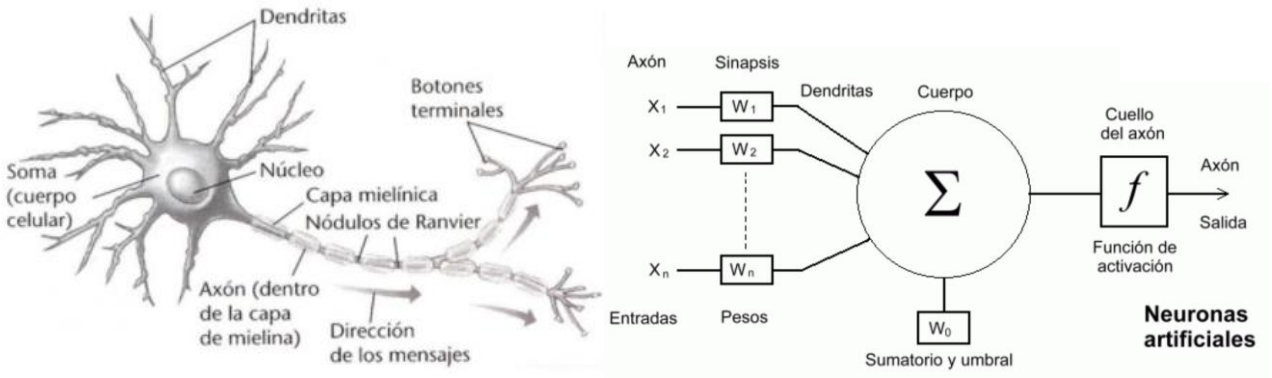


Figura 2-1. Comparativa sobre la estructura de una neurona biológica y una neurona artificial. Fuente [9]

El funcionamiento de la red neuronal básica consistirá en el ajuste de pesos durante el entrenamiento mediante aprendizaje supervisado, utilizando un set datos etiquetados sujetos a cierto criterio establecido anteriormente para esta labor, entregando pares de entrada-salida sobre una de las posibles entradas a la red y la salida que debería de ser entregada para la misma. Para cada una de las iteraciones, como normal general se seguirán los siguientes pasos descritos, esquematizando un modelo simple de Red Neuronal en la Figura 2-2.

Inicialmente, será procesada la entrada x, y será obtenida la salida y. Se comparará mediante la Función de Pérdidas o Costes el resultado correcto con el ofrecido a la salida de la red, de forma que será calculado el error cometido. Mediante el algoritmo de Back Propagation serán ajustados los pesos de la red, con relación al cálculo de gradiente del error respecto a los pesos que dicha técnica calcula con el fin de minimizar fallos en la siguiente iteración. Serán actualizados los pesos de la red según la configuración del parámetro de Tasa de Aprendizaje o Learning Rate, además de la relación entre el valor actual de los pesos con respecto al error cometido.

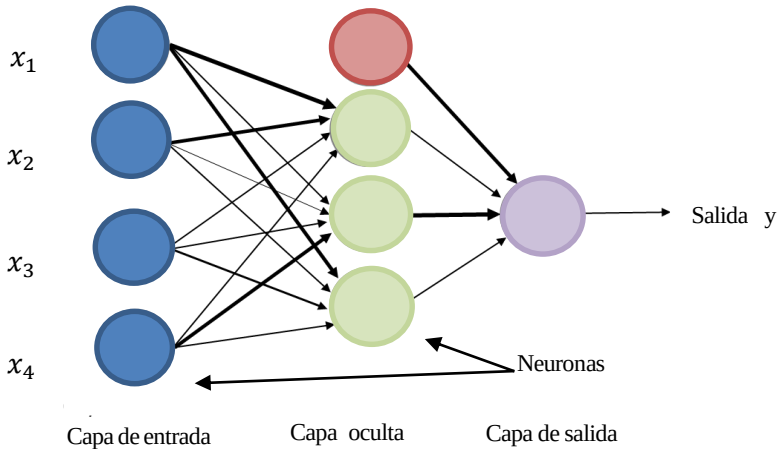


Figura 2-2. Esquema de una Red Neuronal.

Con respecto a las Redes Neuronales Convolucionales sobre las cuales se cimientan las técnicas de detección de objetos planteadas para la resolución del proceso de identificación y seguimiento, éstas consistirán en estructuras más complejas basadas en el comportamiento y la arquitectura descritas con anterioridad.

La extracción de características de las entradas, y el procesamiento de las mismas para identificar y clasificar los diferentes elementos, se realizarán imitando el comportamiento de las neuronas pertenecientes a la corteza visual primaria del cerebro humano. Una CNN está conformada por una arquitectura interna compuesta por la sucesión de diferentes capas como las Convolucionales y Capas de Pooling, Funciones de Activación o Redes de Neuronas Clásicas, entre otras, dependiendo del objetivo perseguido con el uso de la red.

2.2.2 Estructura de la Red Neuronal Convolucional

Una red neuronal convolucional es una red conformada por múltiples capas convolucionales y de reducción alternas que desembocan en la capa de conexión, similar a una red perceptrón multicapa.

Serán desarrolladas a grandes rasgos tres tipos de capas básicos mencionados a continuación.

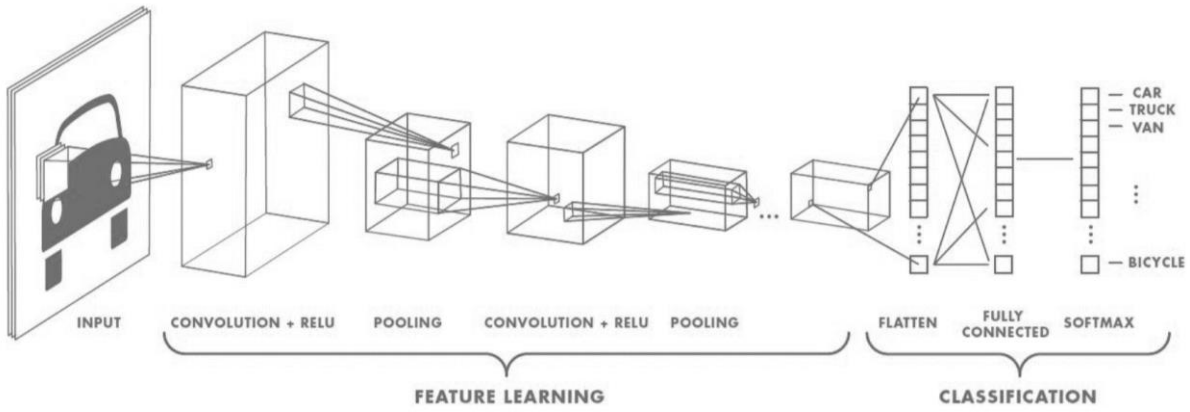


Figura 2-3. Arquitectura de una Red Neuronal Convolucional. Fuente: [11]

Será introducido además el concepto de Kernel para un mejor entendimiento global de la arquitectura, el cual consiste en un conjunto de algoritmos para el análisis de patrones y extracción de relaciones entre éstos y el set de datos derivados para el entrenamiento [10].

Capa Convolucional:

La capa convolucional, se caracteriza, como bien su nombre indica, por efectuar cálculos basados en operaciones de productos y sumas entre las neuronas de partida y los Kernel, lo cual genera un mapa de características relativas a cada una de las ubicaciones probables del filtro en la imagen de entrada. Cada una de las neuronas pertenecientes a esta capa cuenta por tanto con un filtro de convolución, que se irá ajustando a medida que avanza el entrenamiento, para detectar así patrones en la imagen y reflejarlos en la matriz de salida. La primera capa será responsable de la obtención de patrones a bajo nivel (aristas, colores, etc)

El filtro aplicado a este tipo de operaciones es de menor tamaño que la entrada, lo cual reduce notablemente el número de cálculos, de conexiones, y de parámetros de salida respectivamente.

Será esquematizado el proceso anteriormente descrito en la Figura 2-3. La ejemplificación se realiza a través de la matriz de la imagen de entrada, el filtro Sobel convolucional de dimensiones 3x3, y la matriz resultante, tal y como se puede observar en la Figura 2-4. El píxel de destino indicado se obtendrá mediante la multiplicación indicada en la figura. La matriz resultante de salida será de menor tamaño, y será capaz de filtrar ciertas características sobre la imagen de entrada. Una vez obtenido el mapa de características, es aplicada una función de activación a la salida.

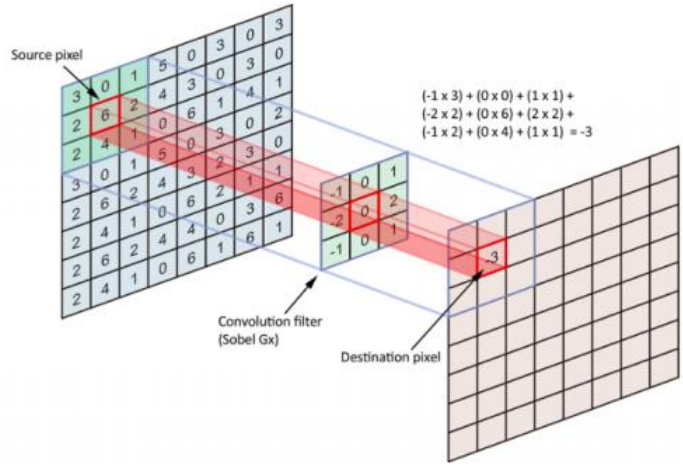


Figura 2-4. Operación de convolución. Fuente: [11]

Capa de reducción o pooling:

Tras la capa convolucional, suele situarse habitualmente la capa de reducción o pooling, cuya funcionalidad consiste en reducir las dimensiones espaciales del mapa de características de la matriz de salida de la capa convolucional, ahorrando cálculo computacional. Generalmente, destacan las operaciones de max-pooling y el average-pooling. Siguiendo el procedimiento ilustrado en la Figura 2-5, será dividida la imagen en un conjunto de submatrices de menor tamaño, y sobre ellas será seleccionado el valor máximo y medio respectivamente.

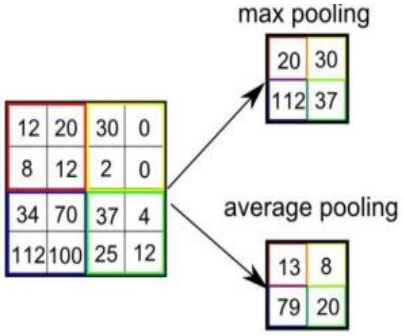


Figura 2-5. Operación de pooling. Fuente [11]

Capa clasificadora

Al final de la red, será situada la capa de clasificación, que será la encargada de analizar las salidas de la red, y de comparar con la clase que más se asemeje a dicha forma. Es por tanto por lo que el número de neuronas en esta última capa será igual al número de clases instanciadas para la detección.

2.2.3 Algoritmos de implementación para el aprendizaje

Gracias a la implementación de CNNs, no será necesaria la caracterización de imágenes gracias al aprendizaje automático. La mayoría de los enfoques propuestos entorno a esta línea consiste en la clasificación e identificación pendiente del estudio de regiones, que podrán contener (o no) las determinadas clases.

2.2.3.1 R-CNN, FAST R-CNN y FASTER R-CNN

R-CNN, FAST R-CNN y FASTER R-CNN [12] son metodologías de búsqueda selectiva mediante regiones candidatas, de forma que se combinan dichos recuadros de forma recursiva y se agrupan los de mayor similitud en espacios potenciales para la localización de la clase. Éstas compondrán las regiones finales, y sobre ellas se aplicará una red que convierta las características en vectores, que supondrán la entrada de un clasificador tipo SVM.

2.2.3.2 YOLO

Será objeto de estudio y aplicación durante el proyecto por la rápida actuación a la hora de clasificar la metodología empleada por YOLO, You Only Look Once [13], que basa su funcionamiento en ejecutar una predicción en tan solo un paso de red, diferenciándose así de sus competidores. En líneas generales, la imagen de entrada queda dividida en celdas de 13x13 en el caso del ejemplo propuesto. Cada una de estas predice 5 cuadros delimitadores, sobre los cuales es generada una confianza o probabilidad de que realmente el objeto esté contenido en ese cuadro.

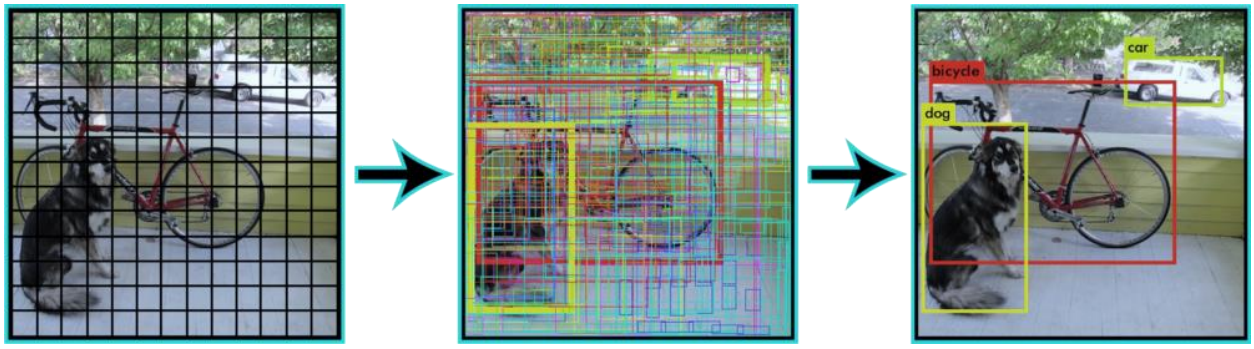


Figura 2-6. Método de detección YOLO. Fuente: [13]

Esta herramienta ponderará dicha confianza de tal forma que se establezca una relación entre el grosor de las líneas y el mayor porcentaje de fiabilidad basado en las clases sobre las que se entrena el modelo, tal y como se muestra de forma simplificada en la segunda imagen de la Figura 2-6. Para cada uno de los delimitadores, la celda ejecutará determinados cálculos de porcentaje de acierto sobre las clases definidas utilizando una distribución de probabilidad. Obtenida la confianza para el recuadro y la clase, será generada una puntuación final que indica qué probabilidad de acierto existe en dicho cuadro para localizar un tipo específico de objeto, lo que se ve reflejado en la segunda imagen de la secuencia anterior.

Inicialmente el número de cuadros generados, unos 845, obtendrán puntuaciones relativamente bajas. Se establece un umbral, por tanto, para mantener aquellos con puntuación final mayor o igual al límite establecido, ajustado según la precisión del detector deseada. Aplicando este filtro, la restricción impuesta

desembocaría en la última imagen de la secuencia anterior, en la que sólo se conservarán tres predicciones con las conclusiones obtenidas, todo ello realizado de una sola pasada ejecutando el proceso de la red neuronal una vez.

YoloV3

YOLOv3 fue lanzado por primera vez en el mes de abril de 2018. Para esa fecha, ya contaba con 53 distintas capas de convolución y era capaz de detectar objetos en un amplio rango de tamaños gracia a la detección en tres escalas. Los algoritmos R-CNN [14], FastR-CNN [15] y Mask R-CNN [16] podían llevar a cabo los mismos objetivos que YOLOv3, pero este último representaba una mejora al estar entrenado para llevar a cabo en el mismo momento la clasificación y la regresión del bounding box [13].

En la actualidad, Yolov3 cuenta con 106 capas y una velocidad de detección compatible con la computación en tiempo real, por lo que será de uso esta red para la obtención de un primer modelo y su posterior evaluación. Sin embargo, la CNN final estará conformada por la mejora sobre el mismo YoloV4 debido a los menores tiempos de detección y a su elevada mejora en cuanto a exactitud.

YoloV4

YoloV4 es lanzado en abril del año 2020, conformando una importante mejora con respecto a YoloV3. La arquitectura del detector, esquematizada en la Figura 2-7, consiste en un detector de dos etapas, donde el backbone es sustituido por CSPDarknet53 y son incluidas ciertas modificaciones en Neck que mejoran el parámetro de evaluación mAP un 10% y el procesamiento de fotogramas por segundo en un 12%.

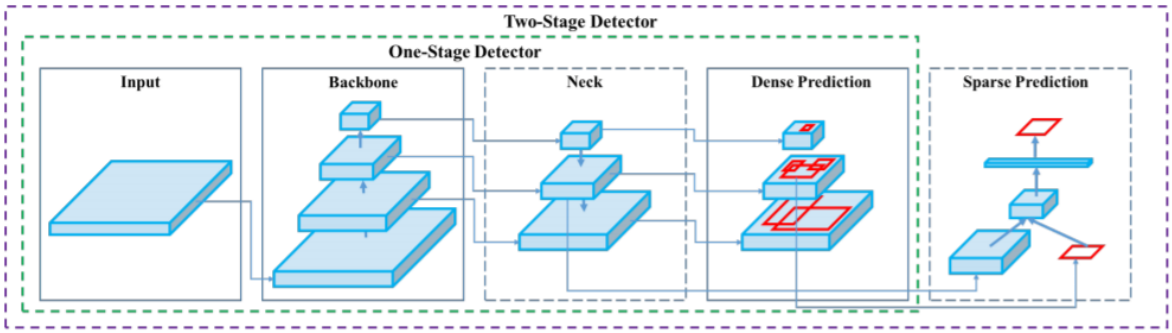


Figura 2-7. Arquitectura del detector YoloV4. Fuente: [17]

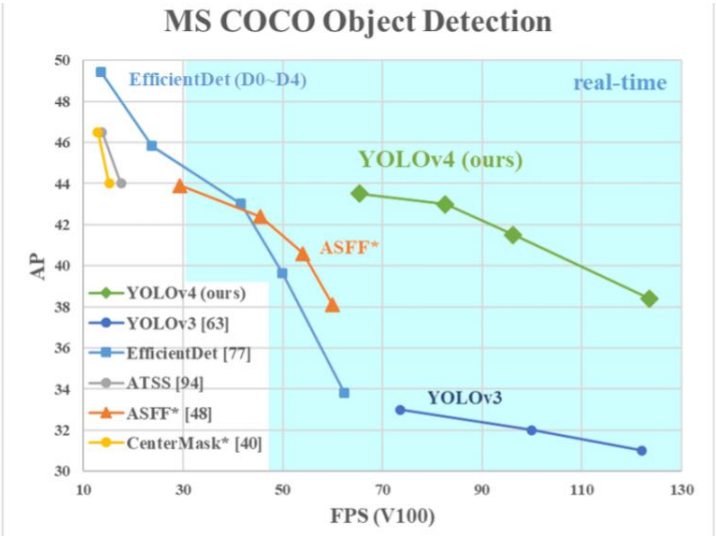


Figura 2-8. Comparación de la velocidad y precisión de diferentes detectores de objetos. Fuente: [17]

(a) (b)

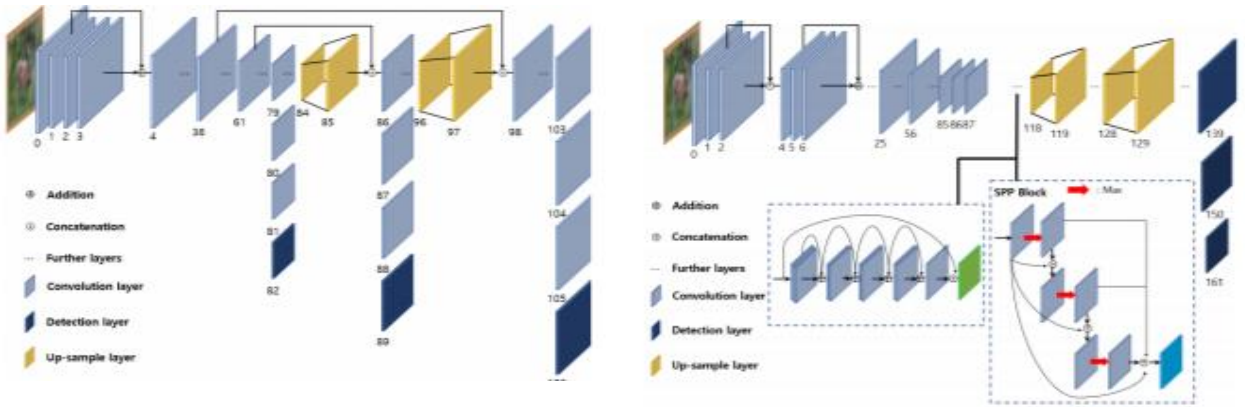


Figura 2-9. Comparativa de la arquitectura de red (a) YoloV3 (b)Yolov4. Fuente: [44]

3 MÉTODO PROPUESTO

Dado a conocer el preámbulo teórico sobre las herramientas disponibles, y las que serán de aplicación para construir la base de una primera aproximación para la solución del problema presentado, se procede con la descripción de los materiales y la metodología que conciernen al prototipo inicial, englobando en ese aspecto a la arquitectura tanto software como hardware, así como la del sistema integrado del demostrador de tecnología, que permitirá el funcionamiento del mismo. Además, serán objeto de estudio las herramientas utilizadas a lo largo del proceso, que conformarán el entorno de programación tanto como para el entrenamiento como para el desarrollo software.

Teniendo en consideración la finalidad que persigue la propuesta, y en consecuencia a la naturaleza del propio proceso de ensamblaje, la arquitectura consistirá en un dispositivo escáner fijo que permitirá la visualización mediante una HMI (Human Machine Interface) ubicada en la estación, que ofrecerá soporte relativo a la identificación y localización de clases en tiempo real, indicando el tipo de conector detectado y la tasa de fiabilidad sobre el mismo, cumpliendo con los estándares de calidad y las exigencias en materia de tiempo,

Finalmente, sobre la justificación en torno a estas decisiones, anotar que éstas se han tomado en base a una primera aproximación y arranque dentro de un proyecto de elevado coste perteneciente al marco europeo que persigue la finalidad de asistencia al operario. Queda, por tanto, encajado, en este prototipo inicial, la contemplativa sobre la revisión bibliográfica anteriormente expuesta, y quedan justificadas las decisiones sobre las necesidades reales de la industria.

3.1 Arquitectura del prototipo

El demostrador de tecnología consiste en un dispositivo escáner industrial capacitado para la toma de imágenes en un tiempo cercano a 15 fotogramas por segundo. Gracias al protocolo Ethernet, se retransmiten los fotogramas captados y se derivan a una estación capaz de procesar, gracias a un alto rendimiento de GPU, la implementación de la Red Neuronal Convolucional que se encargará de la detección del conector en escena, entrenada previamente para este requerimiento. El sistema integrado final consistirá en una interfaz hombre-máquina (HMI) construida sobre lenguaje C++, que se aplicará en tareas relacionadas con la mejora de la experiencia del usuario en procesos industriales manuales. Para ello, se cuenta con un visualizador capaz de identificar conectores industriales e indicar la fiabilidad de acierto sobre la clase detectada en tiempo real.

Para implementar la arquitectura hardware que conforma el sistema integrado, se introduce el esquemático a continuación detallado en la Figura 3-1, además del sistema real, recogido en la Figura 3-2.

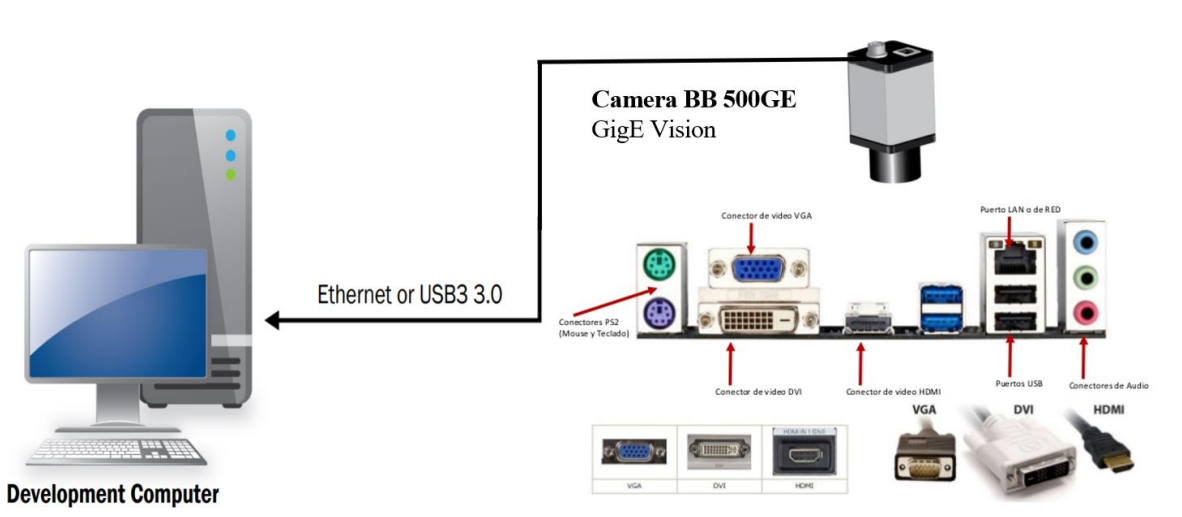


Figura 3-1 Esquemático de la Arquitectura Hardware del Detector.

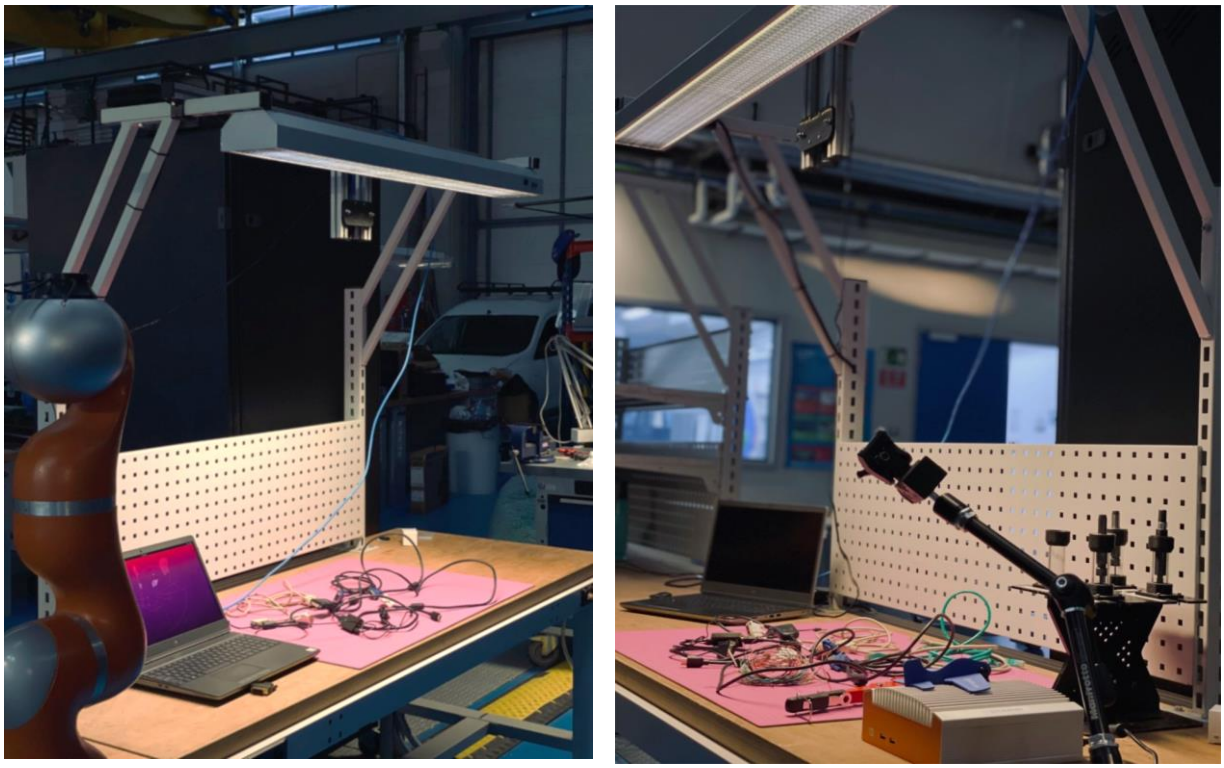


Figura 3-2 Sistema real.

Se explicarán los procesos seguidos en profundidad hasta alcanzar el objetivo final. El demostrador de tecnología, por tanto, quedará compuesto por los elementos que se desarrollarán en los consiguientes epígrafes.

3.2 Dispositivo BB-500 GE

El sistema es dotado de una cámara de escaneo progresivo a color mosaico Bayer con una resolución de 5 megapíxeles e interfaz GigE Vision. La cámara utiliza el CCD Sony ICX625 y funciona a 15 fotogramas por segundo en modo continuo en resolución completa. El consumo del dispositivo es de 5.80 Watt, operando en temperaturas desde -5°C hasta +45°C.

La salida de video puede contener una profundidad de color 8, 10 y 12 bits, de resolución WxH 2456 x 2058 píxeles. El producto pertenece a la Linea Serie B, y cuenta con GPIO con entradas y salidas optoaisladas.

3.2.1 Arquitectura hardware

El escáner posee unas dimensiones 55 x 55 x 55 mm y un peso de 210 gramos, lo cual nos permite contar con un dispositivo ligero y fácilmente transportable. Se especifican a continuación, en la Tabla 3-1, las características hardware del escáner que posibilita la toma de muestras del entorno. Se adjunta además, en la Figura 3-3, los planos técnicos de la arquitectura hardware del dispositivo, pertenecientes al catálogo del fabricante.

Tabla 3-1. Especificaciones técnicas del dispositivo

Especificaciones	BB-500GE
Sensores	1XCCD
Nombre de los sensores	ICX 625AQA
Dimensiones del sensor activo	8.5 x 7.1 mm
Formato óptico	2/3 pulgadas
Tamaño de la celda	3.45 x 3.45 µm
Tipo de obturador	Obturador global
Sensor diagonal	11.1 mm
Montura de la lente	Montura C

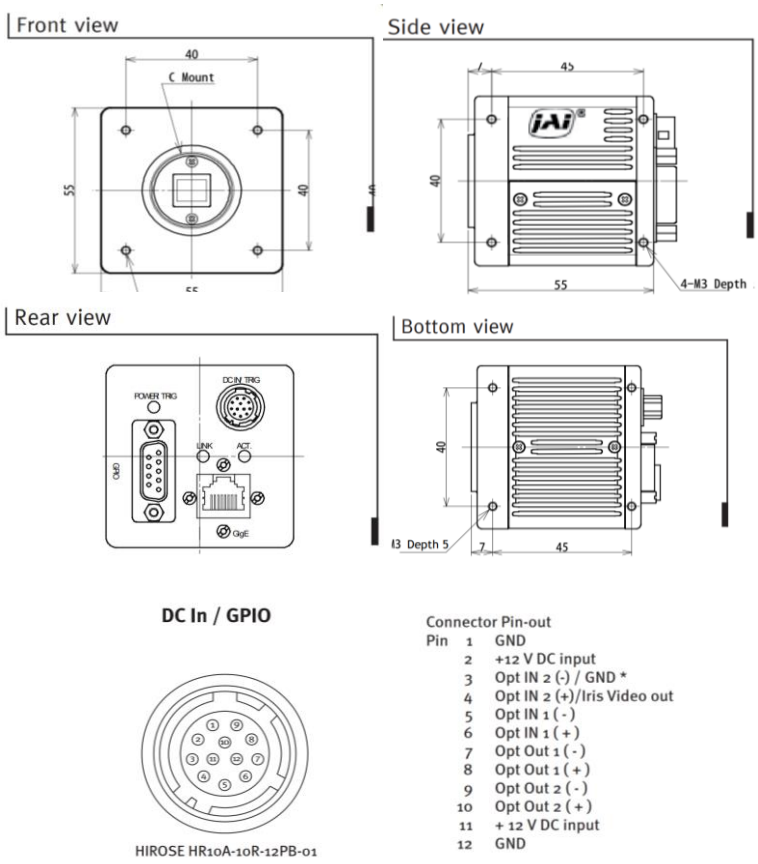


Figura 3-3. Planos técnicos de la arquitectura HW del dispositivo.
Fuente: catálogo del producto que provee el fabricante

Para completar la composición del dispositivo empleado para la toma de muestras del entorno, se analiza el espacio de trabajo disponible para la demostración y se dota al escáner de un objetivo f=25mm/F1.4 (Figura 3-4) que permite captar de forma óptima el escenario analizado.

Figura 3-4. Perspectivas del objetivo para la captación un escenario óptimo.

Debido al uso de la tecnología GigE Vision, la transmisión de datos será a través de un conector Ethernet con una tarjeta de red con capacidad de transmisión de al menos un Gigabit.

3.2.2 Arquitectura software

3.2.2.1 GigE Vision

La transferencia tecnológica persigue estandarizar y unificar protocolos actuales con la finalidad de universalizar la innovación en el ámbito de la ingeniería. Se hace uso por ello de un dispositivo que tiene integrado GigE Vision, un estándar de interfaz introducido en 2006 para cámaras industriales de alto rendimiento, que proporciona un marco para transmitir video de alta velocidad y datos de control relacionados a través de redes Ethernet. Se consigue facilitar a las organizaciones de terceros el desarrollo de software y hardware compatibles en el ámbito de desarrollo de cámaras industriales.

Cabe a destacar que GigE Vision no es un protocolo abierto, por lo que es necesaria una licencia especial para el desarrollo de controladores que usan esta tecnología

3.2.2.2 Ebus SDK 6.2

Se empleará el kit de desarrollo software (SDK) eBUS ™ de Pleora, diseñado para su uso en cámaras y sistemas de video digital de alto rendimiento [18].

La versión mejorada eBUS SDK 6.2 será la seleccionada persiguiendo la reducción de consumo de CPU. Esto supondrá un mayor rendimiento de procesamiento para aplicaciones implementadas a bordo en dispositivos integrados para funciones de escaneado, ideales en labores de inspección. Estas mejoras garantizan la disponibilidad de recursos capaces de llevar a cabo una activación precisa en las aplicaciones de esta naturaleza, además de avalar que podrán ser llevadas a la práctica en el sistema embebido labores de procesamiento y análisis de datos 3D.

Requisitos

EBUS SDK introduce soporte tanto para sistemas operativos Windows como Linux. En este caso, la estación estará dotada de Ubuntu 20.04 LTS de 64 bits. La compatibilidad da soporte para plataformas x86 en la versión 6.2. Para ello, será necesaria la instalación de libavcodec58, una librería FFmpeg, que proporciona un marco genérico para la codificación, decodificación, multiplexación, demultiplexación, transmisión, filtración, reproducción de audio, vídeo, y diferentes filtros de flujos de bits. Abarca a su vez, elevado contenido prolongado en la línea del tiempo, pues es capaz de compatibilizar formatos relativamente longevos con vanguardistas.

La arquitectura compartida proporciona varios servicios que van desde E/S de flujo de bits hasta

optimizaciones de procesamiento de señal digital, lo cual la hace adecuada para implementar códecs robustos y rápidos, así como para la experimentación.

También será necesario, para establecer la conexión entre el dispositivo BB-500 GCE y la estación, un adaptador de red Ehernet de al menos un Gigabit, al utilizar dispositivos compatibles con GigE Vision.

Inicialmente no será necesaria la instalación de un entorno de programación, puesto que la configuración se establecerá a través de una API preprogramada, eBUS Player, el cual se detallará a continuación. Una vez configurada la cámara y establecida la conexión con la estación, se desarrollará una aplicación propia en C++, para lo cual sí será requisito el uso del entorno de desarrollo de programación Visual Studio y el compilador gcc.

El paquete de eBUS SDK, eBUS_SDK_JAI_Ubuntu-20.04-x86_64-6.2.4-5552.deb, será descargado desde el Centro de Soporte de Pleora.

Arquitectura modular

Generalmente, el esquemático en que se basará el comportamiento básico de eBUS SDK en la comunicación con un transmisor Pleora conectado a la cámara se refleja en el diagrama que compone la Figura 3-5. La aplicación de usuario que recibe las imágenes buscará en el sistema una copia de la base de datos XML de la memoria local. Los comandos del dispositivo se envían y reciben a través del puerto de comando GVCP. Las imágenes serán recibidas a través del puerto de transmisión GVSP y serán convertidas a matrices de OpenCV.

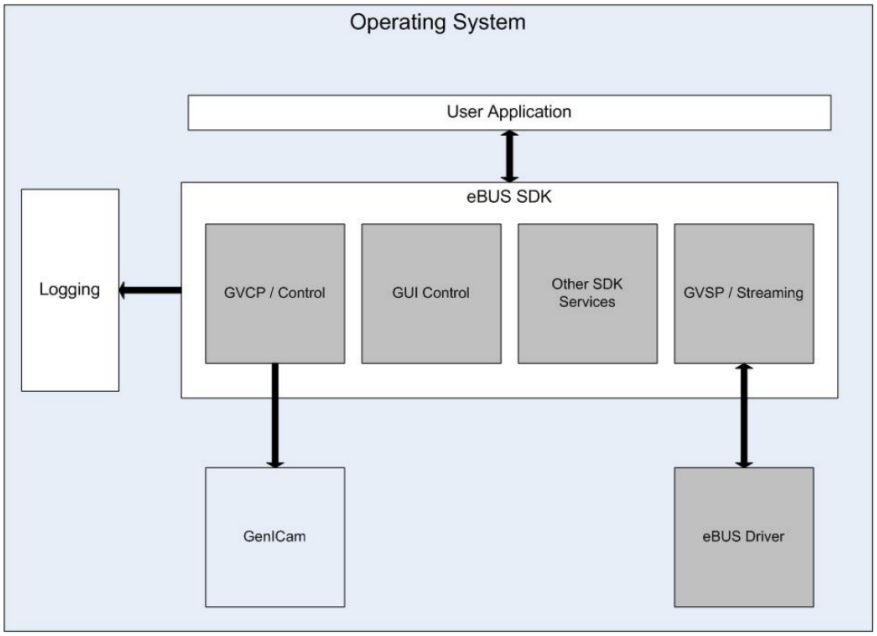


Figura 3-5. Arquitectura modular EBUS SDK. Fuente: [19]

EbusPlayer

Las funciones eBUS Player engloban la funcionalidad Rx del dispositivo, utilizada con la finalidad de detectar y configurar los dispositivos GigE Vision, mostrando y transmitiendo en tiempo real los fotogramas capturados.

Inicialmente con el fin de ajustar los parámetros de configuración y enfoque del dispositivo, además de establecer de forma simple la conexión con la estación, se ha hecho uso de la herramienta eBUS Player, que facilita la experiencia del usuario a la hora de probar y evaluar las características de la aplicación, asentando las bases de desarrollo software que será llevado a cabo con posterioridad.

El reproductor de eBUS Player recibe video y permite a los usuarios ver la transmisión de datos y ajustar la configuración del dispositivo para determinar la configuración óptima para el sistema de visión.

El procedimiento de conexión entre la estación y la cámara se iniciará ejecutando el script de eBUSPlayer, que establecerá las variables de entorno adecuadas, y lanzará el ejecutable.

Para que el dispositivo sea visible, será necesaria la configuración de la IP dentro del mismo rango de red, tal y como se ilustra en la Figura 3-6.

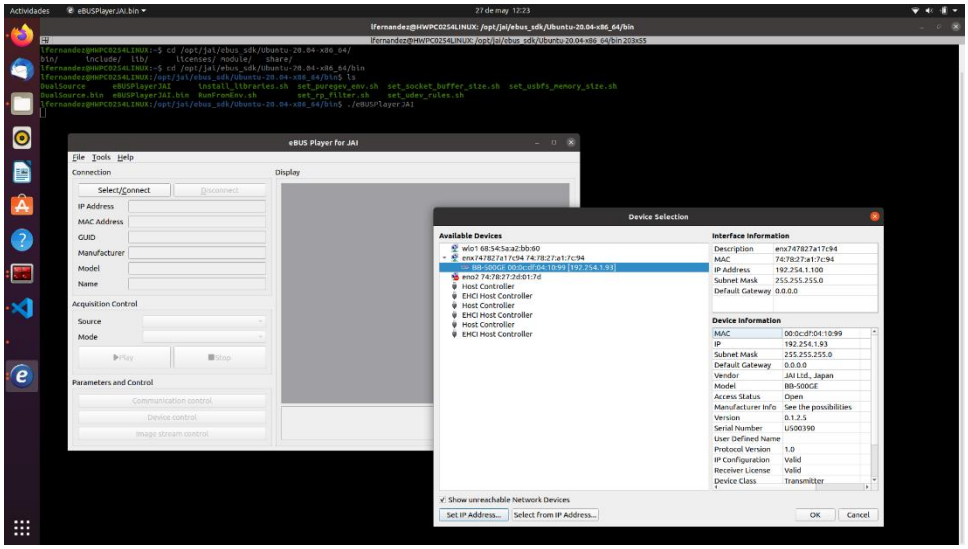


Figura 3-6. Interfaz de eBUSPlayer. Búsqueda del dispositivo dentro de un rango red visible.

Resultados de la configuración del dispositivo

Durante el procedimiento, se han ido recolectando diferentes informaciones referentes a la configuración y al establecimiento de conexión de la cámara. Se facilita a continuación documentación gráfica detallada de ello mediante la Figura 3-7, además de los resultados obtenidos en la Figura 3-8.

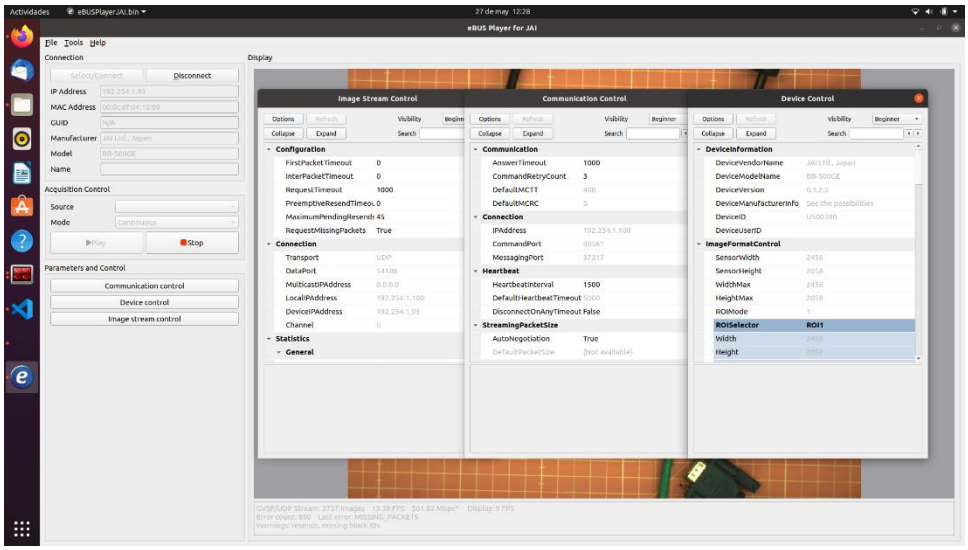


Figura 3-7. Interfaz de eBUSPlayer. Parámetros y control del dispositivo BB-500GE.

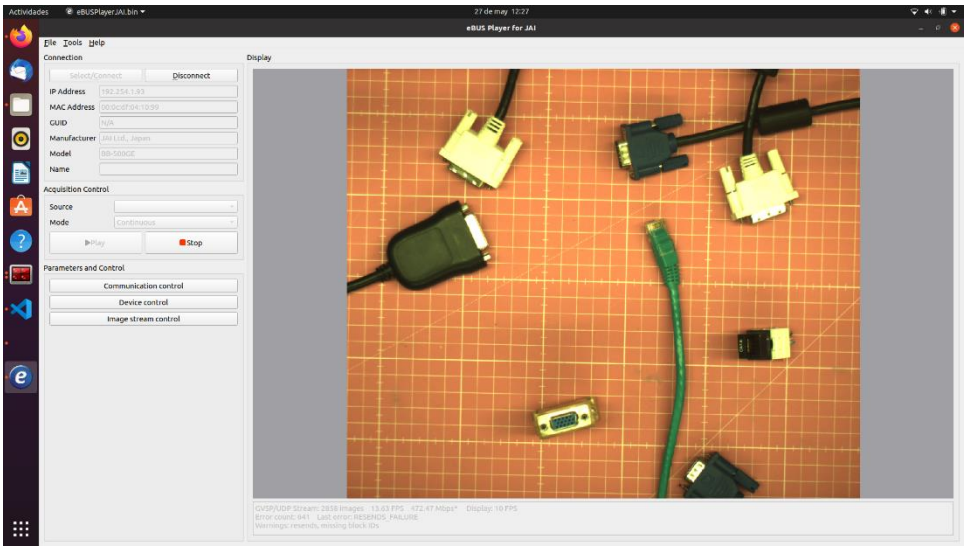


Figura 3-8. Interfaz de eBUSPlayer Resultados de configuración y visualización de fotogramas.

3.3 Set de datos.

El desarrollo de esta tecnología se encuentra orientado al proceso de ensamblaje de satélites espaciales. Además, la línea de investigación cuenta con un potencial proyectable a aplicaciones de realidad aumentada y de inspección que faciliten a los operarios de fábrica su labor y reduzcan sus errores. A efecto de lanzar una primera aproximación, el conjunto de conectores a identificar será de uso comercial, y no los específicos utilizados durante el procedimiento real debido al elevado coste de los mismos. El presupuesto empleado para la demostración no es excesivamente elevado en relación coste-beneficio, por lo que, a tal efecto, el rendimiento mostrado satisface los objetivos a cumplir por la empresa.

Para componer el demostrador de tecnología, serán instanciados 6 tipos de conectores con la finalidad de analizar y evaluar la fiabilidad de la red neuronal entrenada para esta casuística. Éstos serán denominados clases, que contendrán las categorías VGA, HDMI, y RJ45 machos y hembras respectivamente (véase Figura 3-9).

Serán elaborados dos sets de datos, uno orientado al entrenamiento de la red y desarrollo de CNN, consistente en un 90% de la totalidad de los fotogramas capturados. El otro, será utilizado en la evaluación y validación del modelo, consistiendo en un número de muestras menor, cercano al 10% del conjunto total recolectado

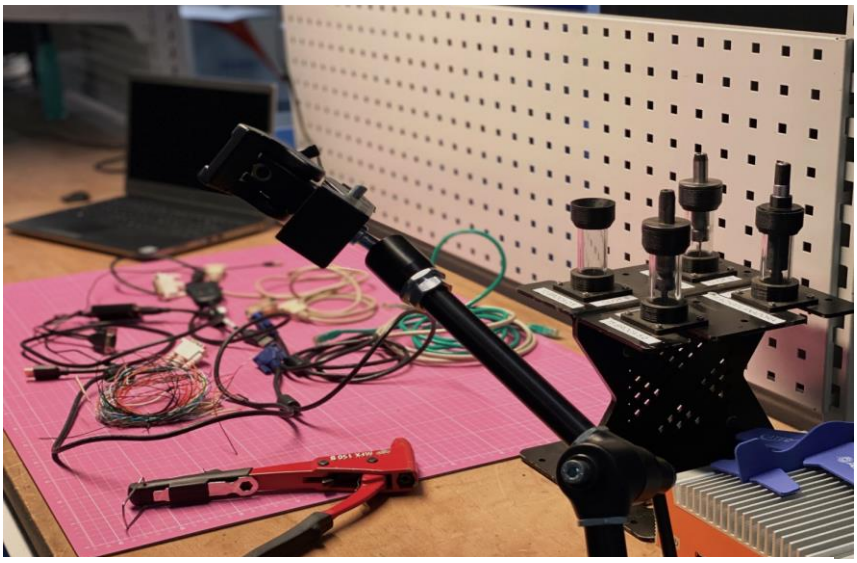


Figura 3-9. Conectores en el espacio de trabajo real.

3.4 Estación para el entrenamiento

Para poder cumplir con los exigentes requisitos computacionales que conlleva la realización del proyecto en tiempo real, se hará uso de la unidad de procesamiento gráfico o GPU, para así reducir tiempos de computación, y liberar a la unidad central de procesamiento CPU.

El equipo utilizado a lo largo del desarrollo del proceso consiste en la computadora Precision 7550 del fabricante DELL. La estación de trabajo portátil de 39,6 cm (15"), soportará alta carga computacional a fin de trabajar en tiempo real, por lo que se encuentra equipada con los últimos procesadores.

Se recogen diferentes especificaciones técnicas en la Tabla 3-2.

Tabla 3-2. Especificaciones técnicas de la estación para el entrenamiento

Componentes	Valor
Fabricante	Dell Inc.
Modelo de sistema	Precision 7550
BIOS	1.7.0
CPU	Intel® Core™ i7-10850H530-1600 kHz @ 2.7 GHz (12 CPUs)
GPU	NVIDIA Quadro RTX 3000
RAM	16384MB RAM
SSD	512GB
OS	Ubuntu 20.04 LTS 64bits

Gracias a esto, es posible trabajar con arquitecturas complejas y tiempos de entrenamientos de la red de una media de 25 horas, facilitando así, estos tiempos computacionales, la comparativa entre diferentes modelos de entrenamiento.

3.5 Entorno de programación

3.5.1 Entorno de programación para entrenamiento

El método de aprendizaje para el desarrollo de las Redes de Deep Learning consistirá concretamente en el uso de la librería de código abierto Darknet, un marco de trabajo de red neuronal escrito en C y CUDA, debido a su fácil instalación, rapidez, y admisión de cálculos con la CPU y la GPU. Concretamente, será utilizado el fork de AlexeyAB. El código fuente se encuentra disponible en GitHub [20], y combina nuevas funciones como WRC, CSP, CmBN, SAT, activación Mish, CmBN, regularización DropBlock o pérdida CIoU, ofreciendo resultados de vanguardia: 43.5% AP (65.7 % AP50) para el conjunto de datos MS COCO a una velocidad en tiempo real de ~ 65 FPS en Tesla V100 [20].

3.5.1.1 CUDA

Compute Unified Device Architecture (Arquitectura Unificada de Dispositivos de Cómputo), también conocido con el acrónimo CUDA, consiste en una plataforma de computación en paralelo que cuenta con modelo de programación desarrollada por NVIDIA®, la cual fue propulsada para codificar algoritmos de hilos simultáneos en la GPU gracias a los múltiples núcleos con los que ésta cuenta.

Esta arquitectura en C/C ++, permite la selección, según la naturaleza de la tarea, de la unidad de procesamiento encargada de ejecutarla, paralelizando de esta forma el proceso, y aumentando exponencialmente el rendimiento y eficiencia de los recursos disponibles. A tal efecto, las implementaciones en serie de las aplicaciones se ejecutarán en la Unidad Central de Procesamiento, y las partes paralelas serán llevadas a cabo por la Unidad Gráfica de Procesamiento.

Al tratarse ambas, CPU y GPU, como dispositivos independientes con espacios propios de memoria, se computa de forma simultánea sin contención de recursos de memoria los diferentes subprocesos. Todo ello es posible debido a los recursos distribuidos, que incluyen un archivo de registro y memoria compartida. La memoria compartida en el chip hará posible que los datos que se encuentran siendo ejecutados en los núcleos de la GPU comuniquen estos datos sin enviarlos a través del bus de memoria del sistema. El flujo de procesamiento llevado a cabo por la Unidad Gráfica de la estación generalmente seguirá el consiguiente patrón.

Primeramente, se copiarán los datos pertenecientes a memoria principal a la memoria de la GPU. Tras esto, la CPU lanzará el arranque del proceso a la GPU, que lo ejecutará paralelamente en cada uno de los núcleos. Finalmente, los resultados obtenidos serán copiados desde la Unidad Gráfica de Procesamiento a la memoria principal. Se detallará en la Figura 3-10, un esquemático del procedimiento descrito.

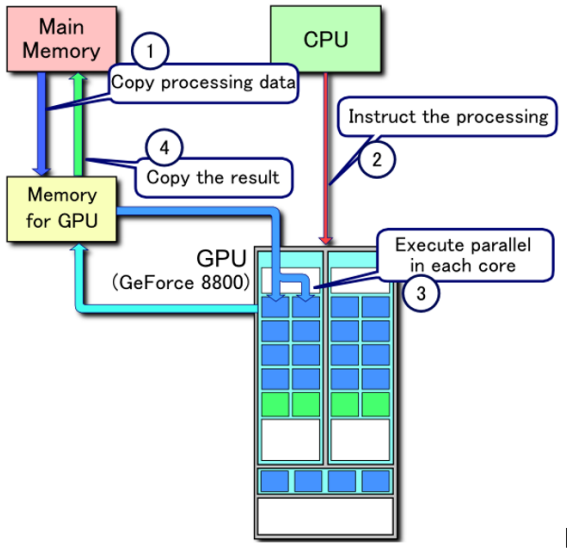


Figura 3-10. Flujo de procesamiento CUDA. Fuente: WIKIPEDIA. Ilustración perteneciente a Tosaka. (2008)

Considerada por finalizada la introducción sobre el flujo de procesamiento de esta arquitectura, se especificará de forma genérica la instalación y verificación del correcto funcionamiento de las herramientas de desarrollo CUDA.

Para la instalación, serán requisitos contar con una Unidad de Procesamiento Gráfica compatible con CUDA. Además, será necesario la instalación de NVIDIA CUDA Toolkit y compilador gcc. Para el desarrollo del demostrador, se procederá con la instalación de la versión 11.0 compatible con el sistema operativo Ubuntu 20.04 LTS, con las especificaciones mostradas en la Figura 3-11.

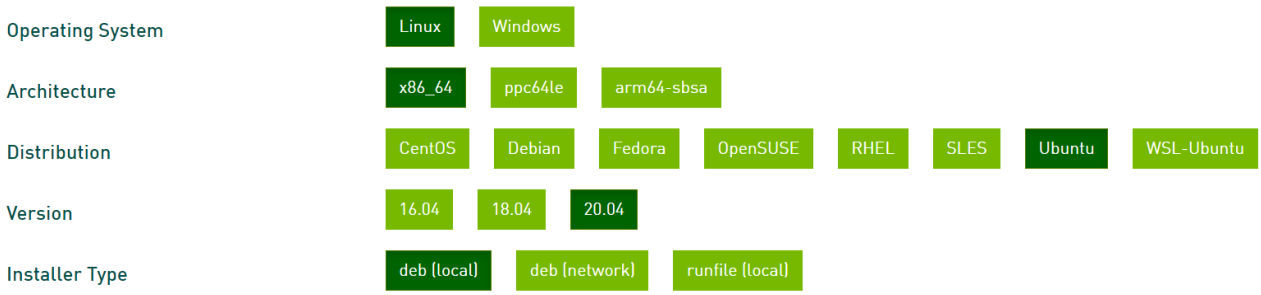


Figura 3-11. Especificaciones sobre la versión CUDA. Fuente: [21]

Será de carácter obligatorio llevar a cabo las acciones de post-instalación especificadas en la página del desarrollador NVIDIA para poder hacer uso del Kit de Herramientas y el Driver.

La primera de ellas consistirá en la configuración del entorno. A tal efecto, la variable PATH se ha de incluir en el directorio, agregando la siguiente ruta a la variable PATH:

```
$ export PATH = / usr / local / cuda-11.0 / bin $ {PATH: +: $ {PATH}}
```

Además, cuando se utiliza el método de instalación runfile, la variable LD_LIBRARY_PATH debe contener /usr/local/cuda-11.0/lib64 en un sistema de 64 bits, de forma que se han de cambiar las variables de entorno para sistemas operativos de 64 bits:

```
$ export LD_LIBRARY_PATH=/usr/local/cuda-11.0/lib64
${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

El segundo procedimiento necesario consistirá en el Setup de POWER9. Debido a las nuevas características incluidas para el controlador de NVIDIA POWER9 CUDA. Existen algunos requisitos de configuración adicionales para que el controlador funcione correctamente. Estos pasos adicionales no serán gestionados durante la instalación de paquetes CUDA, de forma que si no se ejecutan no funcionará la instalación del controlador.

El demonio de persistencia de NVIDIA debe iniciarse automáticamente para las instalaciones de POWER9. En caso de no estar ejecutándose, ha de ser activado.

Será adicionalmente necesario, deshabilitar la regla udev instalada de forma predeterminada en algunas distribuciones de Linux, que hace que la memoria "hot-plugabble" se conecte automáticamente cuando se prueba físicamente. Este comportamiento evitaría que el software NVIDIA ponga en línea la memoria del dispositivo NVIDIA con una configuración no predeterminada. Se adjunta la comprobación de la correcta instalación en la Figura 3-12.

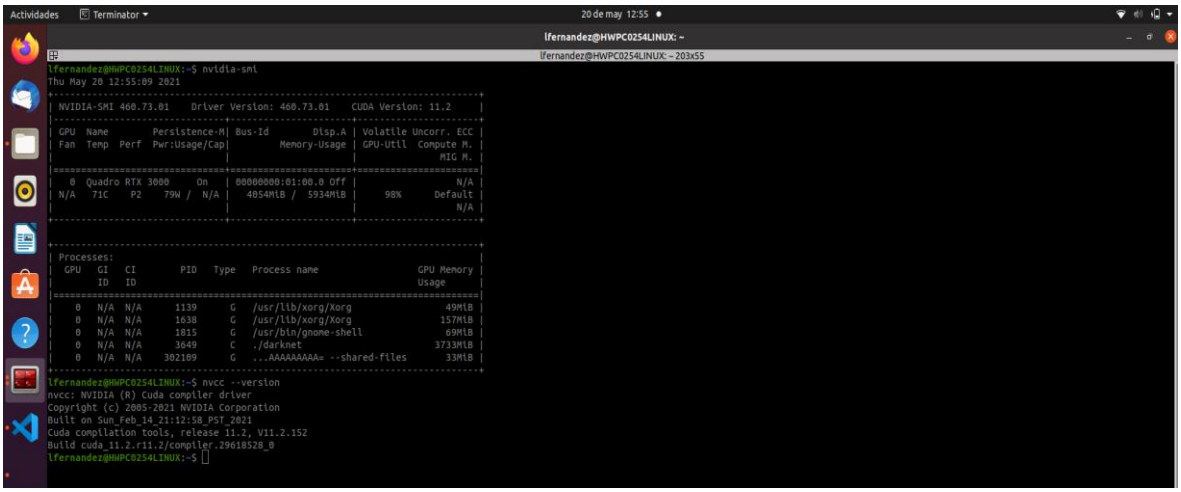


Figura 3-12. Comprobación de la correcta instalación del set NVIDIA y especificaciones.

3.5.1.2 CUDNN

La biblioteca NVIDIA® CUDA® Deep Neural Network Library ™ (cuDNN), consiste en una librería para redes neuronales profundas. Gracias a ello, es posible la aceleración de la GPU de alto rendimiento, ya que proporciona implementaciones altamente ajustadas para rutinas estándar relacionadas con técnicas de Deep Learning, así como convolución hacia adelante y hacia atrás, agrupación, normalización y capas de activación, conformando parte del SDK de aprendizaje profundo de NVIDIA® enfocadas en el entrenamiento de las redes. [22]

La licencia de la versión de la que se ha hecho uso, cuDNN 8.2.1, se encuentra disponible de forma gratuita para los miembros de NVIDIA Developer Program™. Con motivo de evitar futuros problemas de compatibilidad, y finalizar con éxito el tedioso proceso de instalación de CUDA y cuDNN, se procederá a la verificación del correcto funcionamiento de la instalación en Ubuntu, compilando el test mnistCUDNN ubicado en el archivo Debian.

3.5.1.3 OPENCV

Será utilizada, además, la librería de código abierto OpenCV. Dicha librería tiene interfaces para C++, Python y Java, tal que soporta distintos sistemas operativos y agiliza operaciones de procesamiento de imágenes.

Para Darknet, será específicamente un requisito instanciar OpenCV habilitando el módulo de construcción con soporte CUDA, de forma que provea de todas las prestaciones de aceleración de procesado en arquitecturas de GPU ofrecidas NVIDIA. Con ello, se mejorará el rendimiento (eficiencia de Kernels en arquitecturas modernas, flujo de datos optimizado como ejecución asíncrona, copia superpuesta, etc.) y se conseguirá una mayor integridad general de arquitectura. Para diferentes procesos, se puede observar la comparación entre algoritmos OpenCV en CPU y CUDA en la Figura 3-13.

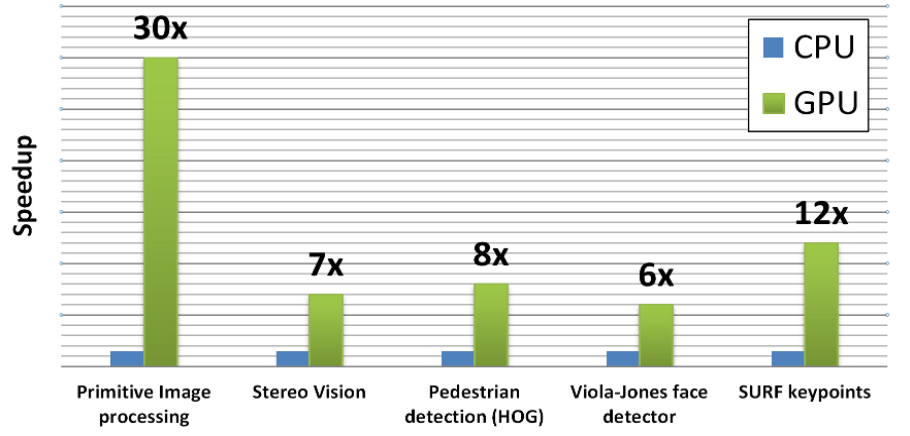


Figura 3-13. Comparación entre algoritmos OpenCV en CPU y CUDA. Fuente: [23]

3.5.2 Entorno de programación para desarrollo software

3.5.2.1 CMAKE

Generalmente, se persigue la integración de las librerías y arquitecturas descritas con anterioridad para el desarrollo de la interfaz usuario-maquina con la que culminarían los objetivos a cumplir del proceso. Para ello, se hará uso de la herramienta de construcción software para desarrollar el proyecto multiplataforma, ofreciendo soporte para el uso concurrente de las mismas.

La versión 3.20 instalada de este framework de código libre está dotada de las últimas modificaciones, de forma que cuenta con las más actualizadas herramientas diseñadas para construir, evaluar y empaquetar software. Es por ello seleccionado para la compilación del proyecto en C++, debido a la posibilidad que ofrece sobre el uso independiente al tipo de plataformas que se gestiona, soportando diversos compiladores y permitiendo la construcción y la búsqueda de rutas de inclusión necesaria de librerías, evitando que éstas sean complejas. Esto supondrá a su vez la flexibilidad y la fácil migración del proyecto a otro computador, de forma que la herramienta será utilizada tanto para la API del SDK de la cámara, como para los futuros desarrollos software que se llevarán a cabo durante el proceso de creación de la HMI.

Además de ser requerido para el correcto funcionamiento de Darknet, CMake será como se ha indicado con anterioridad utilizado para la cohesionan de la totalidad del desarrollo software de la herramienta de visualización y detección de conectores en tiempo real. El archivo de configuración que describe la estructura a seguir el proyecto básico consiste en CMakeLists.txt, donde se recogerán y especificarán los directorios, dependencias de librerías y ejecutables nativos que se generarán.

4 IMPLEMENTACIÓN

En este capítulo serán descritas las diferentes soluciones llevadas a cabo para cohesionar y unificar las diferentes tecnologías que componen el detector de conectores para el guiado de operarios en el proceso de ensamblado de satélites de la Agencia Espacial Europea. Como revisión general del capítulo, se comienza desarrollando la aplicación en C++ que lanzará la configuración de la cámara industrial, y permitirá la toma de imagen y su posterior conversión a matrices de OpenCV. Gracias a ello, se formulará una función capaz de recolectar y almacenar los diferentes sets de datos y validación utilizados con posterioridad para crear los patrones de aprendizaje del modelo.

Descrito lo referente a la programación de la plataforma de escáner, se procederá a la descripción detallada de la casuística relacionada con el entrenamiento de la red neuronal profunda, con el fin de capacitarla para la función específica de detección de conectores requerida, incluyendo los procesos de etiquetado del set de datos, transferencia de conocimientos, y una breve aproximación de los siguientes puntos abordados en el epígrafe sobre el banco de pruebas sobre el modelo y uso posterior.

4.1 Aplicación de usuario C++ EBUS SDK

Con la finalidad de customizar la captura de datos del entorno para el posterior tratamiento de las diversas imágenes tomadas, se desarrolla un software simple específicamente diseñado para detectar los dispositivos disponibles, conectarlos y capturar una secuencia de imágenes en color.

En líneas generales, se desarrolla un código que procesará el stream de datos, el cual contiene las diferentes funciones de conexión y configuración del dispositivo BB-500GE, así como la toma de flujo de fotogramas.

StreamImages.cpp será desarrollado para albergar y configurar lo referente a conexiones y establecimiento de parámetros del dispositivo BB-500GE del fabricante JAI. Para esta labor, se instanciará la clase jai_500_ge y se definirán las funciones:

- *ConnectToDevice*
- *OpenStream*
- *ConfigureStream*
- *CreateStreamBuffers*
- *EnableAcquireImages*
- *AcquireImages*
- *FreeStreamBuffers*
- *raw_to_bgr*

Con el fin de simplificar la posterior implementación de la configuración y conexión de la cámara para la HMI, será desarrollado otro código con la única finalidad de tomar una secuencia de imágenes en color y transformarlas a formato *cv::Mat*. Se llamará a *StreamImages* y a la clase *jai_500_ge*, para establecer la conexión y configuración, y se establecerá una orden de comprobación para que la configuración del dispositivo sea lanzada tan solo una vez. Se retornará *img_clone* para evitar que sólo se retorne la cabecera de la imagen por la optimización de código, evitando de esta forma posteriores conflictos.

Para cohesionar el proceso, se recoge en *CmakeLists.txt* las rutas de librerías para el Ebus SDK Pleora para Linux 64-bits. En cuanto a las opciones de compilador, será establecido C++11. Se llama a la función *find_package* y se incluyen los directorios de EBUS Y OpenCV. Tras ello, se construye una librería en C++ y se enlazan las actuales. Finalmente, se construyen los ejecutables en C++.

Lograda la configuración y control del dispositivo, la comunicación unidifusión y multidifusión, la adquisición de imágenes y datos y la visualización customizada en formato OpenCV, se utilizará el software desarrollado en los posteriores procesos como herramienta en la toma del set de datos. Además, conformará una parte crucial en la integración de la plataforma final capaz de detectar conectores en tiempo real.

4.2 Tratamiento y preparación del conjunto de datos para el entrenamiento y validación.

Generalmente, se amplía la toma de fotogramas de una captura, a un flujo de imágenes. Así, se generará la toma de datos del entorno para el entrenamiento de la red y la posterior validación de la misma, mediante el desarrollo de una función cuya finalidad reside en organizar el conjunto de clases definidas y recolectar imágenes pertenecientes a cada una de ellas.

La carpeta será creada desde el código fuente, especificando el nombre de esta, además del de cada imagen tomada según la clase específica. A tal efecto, quedarían numeradas temporalmente según captura, lo cual facilita la identificación a primera vista de los datos contenidos. Por ende, se contaría con la facilidad y el sencillo manejo de los resultados obtenidos, ya sea en caso de querer aumentar la variabilidad para dotar al algoritmo de mayor transversalidad diferenciando características en diferentes escenarios, o de eliminar resultados no deseados en ciertos tramos de la toma de imágenes.

Debido a los posibles errores revisados en el estado del arte, se estima un número aproximado de 1.200 fotos por clase contemplada para el entrenamiento de la red. Deep Learning ha demostrado su buen funcionamiento con grandes números de variables, motivo por el cual será de aplicación un elevado contenido de datos para la creación del modelo.

Las 6 clases instanciadas y definidas serán:

MALE_VGA_CONNECTOR, FEMALE_VGA_CONNECTOR, MALE_DVI_CONNECTOR, FEMALE_DVI_CONNECTOR, MALE_RJ45_CONNECTOR, FEMALE_RJ45_CONNECTOR.

Del registro de datos elaborado de cada una de las clases, el 10% de su totalidad serán derivados para la elaboración de un set de validación, cuya finalidad y funcionalidad residen en testificar, evaluar y comprobar la fiabilidad de clasificación con la que cuenta la red obtenida durante el entrenamiento para cada una de las clases definidas. El 90% de los datos restantes generados, serán derivados y definidos como el conjunto de entrenamiento, conformando aquellas imágenes junto a su correspondiente mapa de probabilidad etiquetado que la red utiliza a la hora de ajustar los pesos.

Será de crucial importancia la automatización del proceso tanto de etiquetado como de recolecta de datos, aunque se trate de una tarea supervisada, puesto que fundamenta uno de los propósitos perseguidos con la elaboración del proyecto. Debido a la extensión del registro de variables, será crucial la correcta organización del contenido y la meticulosidad a la hora del tratamiento de sets de datos para el entrenamiento de la CNN.

4.3 Darknet

4.3.1 Requisitos

Como se ha especificado de forma anterior, es necesario el uso de la unidad de procesamiento gráfica para proceder de forma exitosa con las diversas funcionalidades que se buscan cumplimentar durante el proceso de desarrollo del proyecto.

Además, será requisito obligatorio, tal y como se ha indicado en la metodología, la instalación de CUDA, que permite usar las GPUs para cálculo paralelo, la librería CuDNN, para poder implementar las librerías que utiliza darknet, OpenCV, GPU con CC>=3, y CMake.

4.3.2 Compilación en Ubuntu 20.04LTS 64-bits

El principal objetivo de este trabajo es diseñar un sistema rápido, con un detector de objetos en sistemas de producción y optimización para cálculos en paralelo con elevada velocidad. Para la labor de entrenamiento, desde GitHub se clonará el repositorio de Darknet perteneciente al autor AlexeyAB. Sobre el directorio donde se encuentra, se ejecuta el comando make y se ajustan los siguientes parámetros en CMakeLists.txt, con el fin de instanciar las dependencias opcionales instaladas como CUDA, CuDNN, y ZED. También creará un archivo de librerías de objetos compartidos para usar Darknet para el desarrollo de código.

GPU = 1 para construir con CUDA para acelerar el proceso usando GPU (Localización de CUDA en

/usr /local /cuda)

CUDNN = 1 para compilar con cuDNN (Localización cuDNN en /usr /local /cudnn)

CUDNN_HALF = 1 para construir Tensor Cores (Titan V/Tesla V100/DGX-2 y posteriores) Acelera la Detección 3x, y el Entrenamiento 2x.

OPENCV = 1 permite detectar archivos de video y transmisiones de video de cámaras de red o cámaras web

DEBUG = 1 para construir la versión de depuración de Yolo

OPENMP = 1 para construir con soporte OpenMP para acelerar Yolo usando CPU de múltiples núcleos

LIBSO = 1 para construir una biblioteca darknet.so y un archivo binario ejecutable uselib que usa esta biblioteca. Esto será utilizado para la integración futura de la HMI. [20]

4.3.3 Creación de la CNN customizada

Una vez comprobado el correcto funcionamiento e instalación de Darknet en la estación, se procede con la creación de una red customizada capaz de cumplir con las expectativas relativas a la detección de conectores. Para ello, se hace uso de los pesos pre-entrenados mencionados con anterioridad.

La configuración de Darknet para la preparación del entrenamiento, consiste en los pasos que se describirán a continuación.

Primeramente, se creará un archivo de configuración *yolo-obj.cfg* con la misma estructuración de contenido que el archivo *yolov4-custom.cfg* con ciertos cambios. Será modificado el campo de la línea batch y subdivisiones al valor 64. Tras ello, se ha de establecer conexión con el parámetro de max_batches, que será modificado multiplicando el número de clases x 2000, de forma que el resultado será no menos del número de imágenes de entrenamiento, y no menos de 6000. Steps también será susceptible a cambios, convirtiéndose en un 80% y 90% del valor de max_batches. El tamaño de la red será establecido como width=416, height=416.

También será objeto de modificación el número de clases en cada una de las tres capas de YOLO, siendo 6 (igual al número de objetos a identificar durante el entrenamiento de la red). Consecuentemente, se realizará una variación el número de filtros en la sección indicada como convolucional antes de cada una de las capas yolo, teniendo en cuenta que solo ha de ser modificado el último convolucional según la regla filtros = (clases + 5) x3, tomando un valor por tanto de filters=33.

Otros parámetros de entrenamientos configurables serán el Learning rate o tasa de aprendizaje, que indica el cambio que ha de experimentar el modelo en respuesta al error estimado cuando se actualizan los pesos del modelo. Dicho parámetro será ajustado con un valor de 0.001. Por otro lado, los anchors, o tamaños de altura y anchura iniciales, alguno de los cuales (los relativamente similares al tamaño de objeto), cambiarán las dimensiones al tamaño del objeto, utilizando resultados del mapa de características de la red neuronal final. Sus valores serán establecidos como 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 459, 40.

Por otro lado, se procede con la creación de un archivo nombrado como *obj.names* en el directorio build /darknet /x64/data/, que contiene los nombres de las clases, cada una en una nueva línea. Dichas clases serán igual al número de objetos a identificar.

El set de datos que contiene las imágenes de los conectores para el entrenamiento será ubicado en el directorio build/darknet/x64/data/obj/. Será creado un archivo *train.txt* que contiene todas las rutas relativas de ubicación de dicho dataset.

Igualmente, el set para validación será ubicado en build/darknet/x64/data/obj/ y tendrá asociado un archivo *valid.txt* que indica la ruta de ubicación de las imágenes.

Cada uno de los objetos deben de estar correctamente etiquetados siguiendo las directrices que se especificarán en la siguiente sección.

Toda esta información será agrupada en un nuevo archivo llamado *obj.data*, situado en el directorio build /darknet /x64/data/, que contiene las siguientes definiciones y asignaciones: clases=6, train=ruta local hasta donde se ubica el archivo train.txt, valid= ruta local hasta donde se ubica el archivo de validación *valid.txt*,

names= ruta local hasta donde se ubica el archivo *obj.names.txt* y backup=ruta donde se guardarán los pesos generados durante el entrenamiento.

4.3.4 Labelling o etiquetado del set de datos y validación

La automatización del etiquetado es uno de los objetivos perseguidos durante la ejecución del proyecto, conformando un punto clave en el mercado laboral, debido a lo tedioso y lento del proceso. Además de mejorar el proceso de elaboración de bases de conocimiento de la red, la implantación de un sistema de tracking automático facilita la labor de mejorar el modelo o de realizar cambios en determinadas etapas del proceso sin interferir notoriamente en el tiempo y trabajo ya realizado.

El etiquetado tanto del set de validación como del set de datos para el entrenamiento consistirá en la creación de un archivo con extensión *.txt* para cada una de las imágenes, en el mismo directorio y con el mismo nombre de éstas. El contenido del archivo *.txt*, contendrá el objeto (u objetos) presentes en su imagen correspondiente, así como las coordenadas, en un formato reconocido por Darknet (véase Figura 4-1). Cada objeto encontrado en la imagen será definido en una línea, y contendrá la siguiente estructura:

<clase> < x_center > < y_center > < width > < height >

Donde:

<clase> - número de objeto entero de 0 a (clases-1)

<x_center> <y_center> <width> <height> - son valores flotantes relativos al ancho y alto de la imagen, pudiendo variar en un intervalo de (0 1.0]

coordenada <x> = <absolute_x> / <image_width> o la altura <height> = <absolute_height> / <image_height>

Consideración: <x_center> <y_center> - serán el centro del rectángulo que se trazará alrededor del objeto etiquetado.

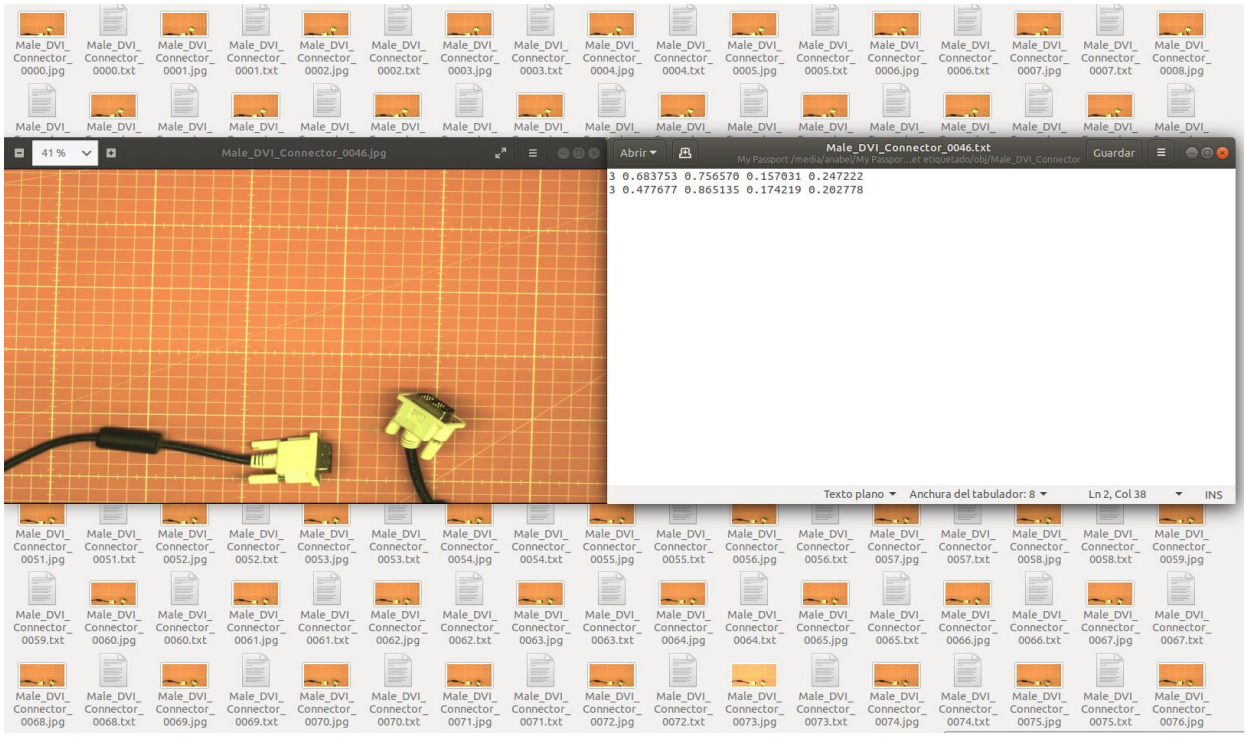


Figura 4-1. Etiquetado del set de datos.

Para este procedimiento, se ha hecho uso de la herramienta YOLO-mark, una interfaz gráfica de usuario de código abierto compilable en Linux [24]. Se configuran los archivos y las rutas de imagen correspondientes, estableciendo un paralelismo en la forma de proceder con la creación de los archivos *obj.data*, *obj.names*, *train.txt*, etc.

Preparados los archivos de configuración y rutas relativas, se lanza el ejecutable perteneciente a esta herramienta. Para cada imagen, se identifica el tipo de conector del que se trata, trazando una bounding box sobre éste. Instanciados los cuadros delimitadores sobre la imagen, se inicia el seguimiento automático sobre la clase en los consiguientes fotogramas del proceso de etiquetado. Todo ello es supervisado con el fin de crear unas bases de datos fiables y obtener el mejor de los resultados durante el entrenamiento, de forma que mejora notoriamente la experiencia del usuario.

4.3.5 Transfer Learning sobre modelos pre-entrenados

Debido a que los futuros pasos han de ser dirigidos hacia abarcar un mayor número de conectores, incluyendo los pertenecientes a los satélites espaciales; se hará uso de técnicas de transferencia de conocimientos desde una red neuronal pre-entrenada a fin de comprobar la viabilidad futura de las técnicas de Transfer Learning. Por ello, además de procesar el reconocimiento de conectores, el modelo obtenido estará capacitado para identificar las 91 categorías incluidas en MS COCO (Microsoft Common Objects in Context), una base de datos de segmentación, detección de puntos clave y subtítulos de objetos, consistente en 328K imágenes, con más de 200K etiquetadas. Este set, ilustrado mediante la Figura 4-3, sienta las bases de modelo pre-entrenado sobre el cual se comenzará a construir.

Previamente a la propuesta indicada, se moduló sobre ésta red otro sistema inteligente capaz de reconocer determinadas piezas aeronáuticas que podrían ser de utilidad para la asistencia del operario en los escenarios propuestos. Por consiguiente, esta red será reutilizada, reentrenando el modelo con el fin de aumentar los campos de detección. Cabe a destacar, que la finalidad perseguida y los valores de precisión se encuentran focalizados en los conectores recogidos en la Figura 4-2, VGA, RJ45 y DVI tanto hembras como machos, aunque debido al mencionado Transfer Learning, el modelo sea capaz de reconocer otras clases como los objetos aeroespaciales mencionados o las categorías incluidas en el set de MS COCO.

Figura 4-2. Especificaciones sobre los conectores a identificar.

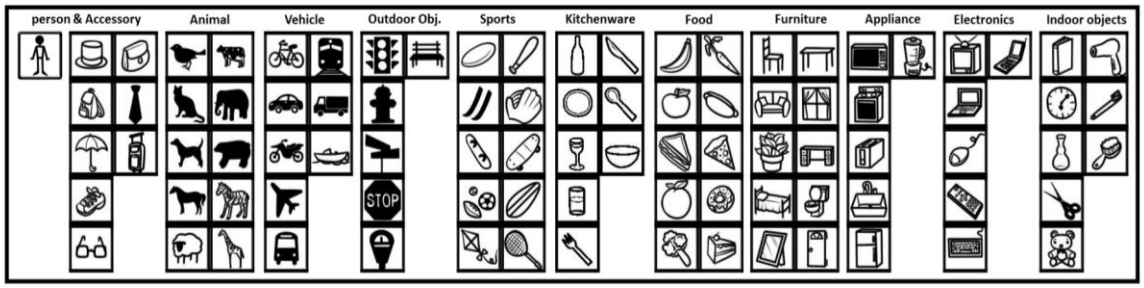


Figura 4-3. Iconos de las 91 categorías del set de datos de MS COCO agrupadas en 11 super-categorías. Fuente Microsoft COCO: Common Objects in Context

Para el entrenamiento en sí, se creará, tal y como se ha especificado con anterioridad, un archivo de texto, *train.txt*, en el directorio build\darknet\x64\data\. Este documento contendrá los nombres de las imágenes, cada uno en una nueva línea, con su ruta relativa.

Por otro lado, los pesos pre-entrenados para las capas convolucionales se situarán en el directorio build\darknet\x64.

Una vez cumplidos los requisitos anteriores, se procede con el entrenamiento en sí: cada 100 iteraciones, se guardará el archivo *yolo-obj_last.weights*; y cada 1000 iteraciones, se guardará un archivo *yolo-obj_xxxx.weights*. Con el fin de poder comparar posteriormente los modelos obtenidos, será activado el cálculo de mAP & Loss-chart durante el entrenamiento por cada 4 Epochs. Se adjunta en la Figura 4-4, una imagen recogida durante el entrenamiento de la red.

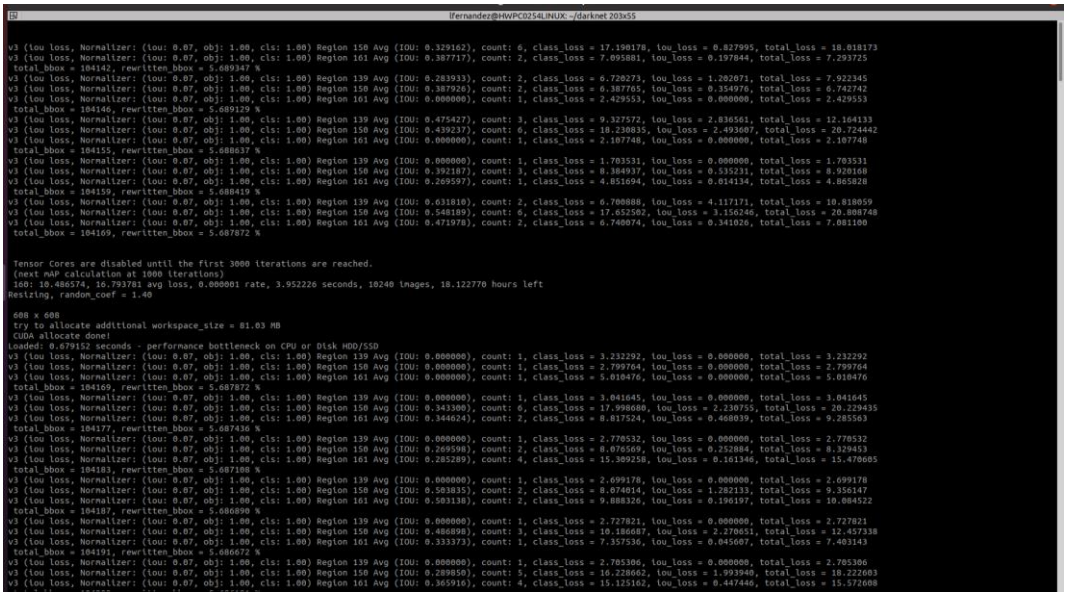


Figura 4-4. Imagen captada durante el entrenamiento.

4.3.6 Banco de pruebas sobre el modelo y uso posterior

El entrenamiento de la red funciona con convoluciones de pixeles aleatorios. Durante el proceso, se tiene acceso a diferentes parámetros que se van calculando durante el proceso de iteración

Como se ha comentado en anteriores epígrafes, la estación de entrenamiento toma un tiempo relativamente largo para esta labor; alcanzando una media de entre 25 y 30 horas para este caso de entrenamiento. Con el fin de evaluar a lo largo del proceso los resultados que se van generando, y a fin de evitar posibles errores de overfitting o underfitting, se van extrayendo ciertos pesos en un determinado número de iteraciones para valorar la necesidad de continuar con el entrenamiento o en caso contrario de pararlo.

Serán de crucial importancia los factores de pérdida y la precisión media de acierto sobre el set de validación. Más detalles sobre estas métricas se darán a lo largo del desarrollo de la Sección 5 para la confección y desarrollo de una red de elevada fiabilidad.

Una vez conseguido aumentar el porcentaje, se seleccionará el modelo pertinente que ofrezca mejores resultados, el cual se volcará sobre interfaz humano-máquina. Se establecerá como parámetro los pesos con mayor tasa de acierto sobre el set de validación resultante del proceso de evaluación sobre los diferentes entrenamientos realizados, y las técnicas de mejoras aplicadas. Los epígrafes 5 y 6 contienen información detallada sobre el procedimiento seguido para postular al mejor candidato e integrar la red en la aplicación de usuario.

5 RESULTADOS

En este capítulo serán objeto de análisis, los pesos resultantes del proceso descritos en la sección anterior. Además de ser evaluados, éstos serán mejorados mediante diversas técnicas. Se validará por tanto el sistema de detección diseñado.

Para establecer la estructura y orden de los datos a lo largo de esta sección, se presentan los recursos computacionales consumidos a lo largo del entrenamiento. La evaluación será llevada a cabo a través de Darknet. Una vez revisados estos parámetros, se dará paso al análisis de resultados obtenidos con la red convolucional YoloV3, utilizando diversas técnicas para evaluar precisiones y pérdidas del modelo, obteniendo la fiabilidad sobre la clase detectada.

Sobre los resultados, será de aplicación un conjunto de acciones para afinar la fiabilidad sobre los primeros acercamientos al modelo. Se desarrollará dicha mejora mediante la técnica de aumento de datos y el uso de una red pre-entrenada con mayor número de capas convolucionales, YoloV4, obteniendo una aproximación final con un 93% de fiabilidad sobre el set de validación. Serán objeto de análisis también los tiempos de detección sobre el modelo seleccionado para validar la implementación del modelo en una aplicación en tiempo real, detallada a lo largo de la Sección 6.

5.1 Recursos computacionales consumidos durante el entrenamiento

Sobre los recursos computacionales requeridos durante el entrenamiento, se destaca el alto rendimiento de la GPU durante el proceso, con una ocupación que ronda el 90-100% de la capacidad total.

Se muestran gráficamente en la Figura 5-1 los detalles sobre la computación. Desde el eje central hacia la izquierda, se muestra el proceso del entrenamiento en tiempo real. Desde el eje central hacia la derecha, en la parte superior se ejecuta NVidia TOP, un monitor se tareas tipo *(h)top* para las GPU NVIDIA [25]. Se imprime de tal manera la información referente a las múltiples unidades de procesamientos gráficas con las que cuenta la estación de forma intuitiva.

En el marco inferior derecho, se muestra el resultado de la ejecución del comando *htop*, un sistema de monitorización, administración y visor de procesos interactivo. En la parte superior, o cabecera, se muestra el porcentaje de uso de CPU, la memoria RAM y la memoria de intercambio SWAP. También se muestra el número de tareas activas durante el entrenamiento de la red, en este caso 177, el promedio de carga del sistema y el tiempo de actividad, que son unas 22 horas desde el comienzo del entrenamiento.

En el cuerpo de esta representación, encontramos todos los procesos en ejecución activa del sistema.

- PID representa la ID del proceso llevado a cabo.
- USER el usuario de la estación.
- PRI la prioridad del Kernel.
- NI el valor y prioridad del proceso.
- VIRT el tamaño del proceso.
- RES representa el tamaño de la cantidad física real que consume la tarea.
- SHR representa la cantidad de memoria compartida.
- R indica que el proceso de entrenamiento se encuentra activo.
- CPU % representa el porcentaje del tiempo de CPU utilizado.
- MEM% el porcentaje de memoria en uso.
- TIME el número de ciclos de reloj invertidos.
- Command indica la ruta del comando.

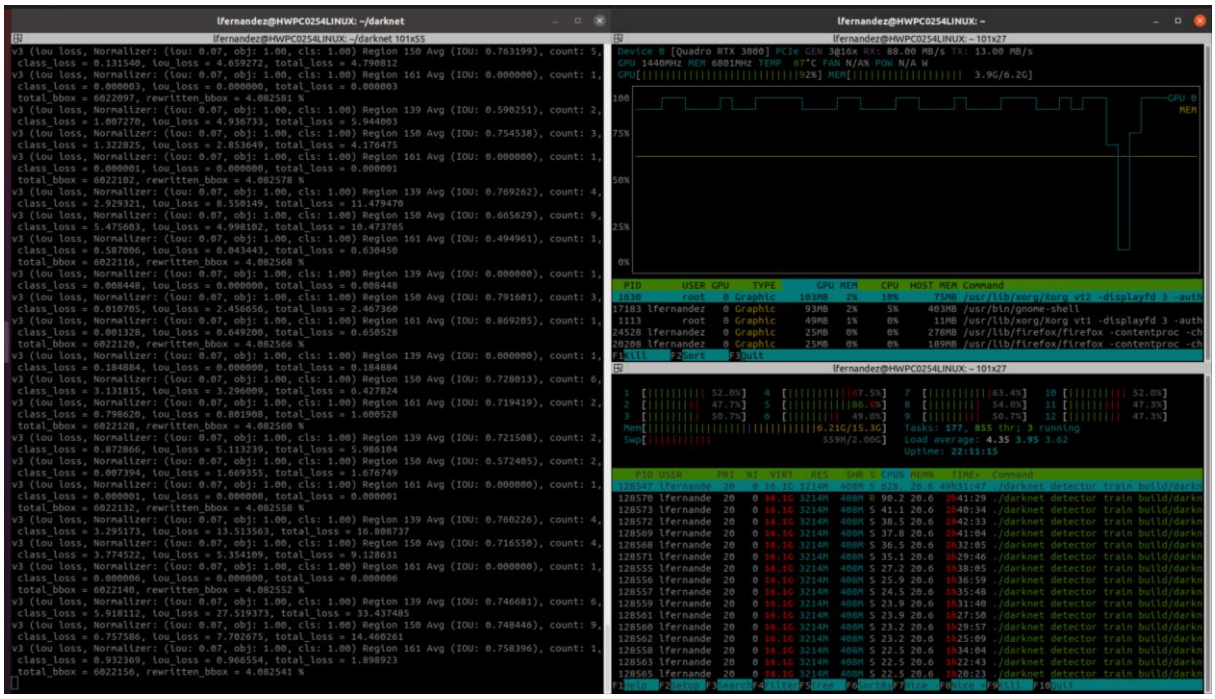
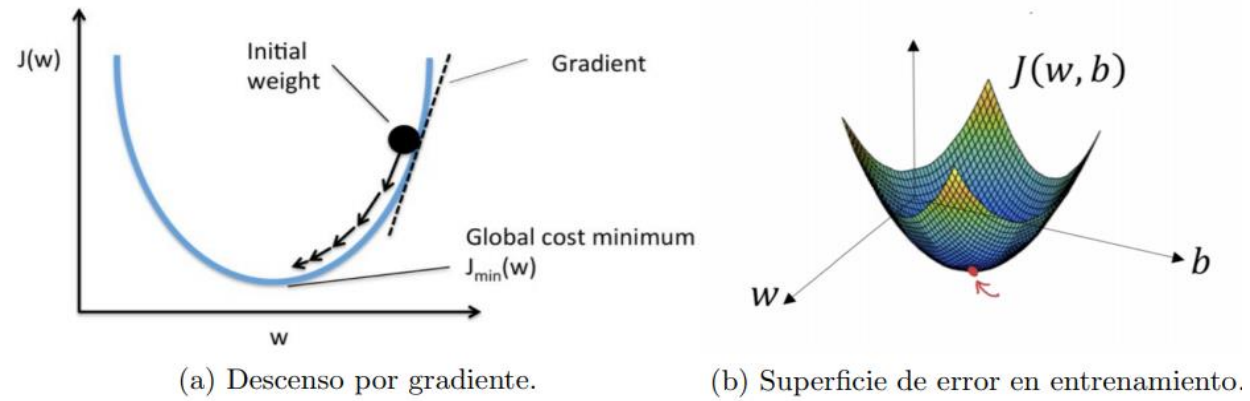


Figura 5-1. Recursos computacionales consumidos durante el entrenamiento. Evaluación del modelo y obtención de métricas durante el entrenamiento

5.1.1 Consideraciones sobre el modelo durante el entrenamiento

El entrenamiento consiste en el proceso mediante el cual se desarrolla la capacidad de aprendizaje de la red, a través del cual los pesos del modelo pre-entrenado serán siendo modificados mediante la Función de Pérdida o error, midiendo la diferencia existente entre la clase correspondiente a la salida y la clase real. Mediante el cálculo del gradiente de error, serán reducidos los errores de manera gradual gracias al descenso por gradiente. El mínimo global sobre la gráfica de error de la Figura 5-2 será el objetivo perseguido a lo largo de las horas de entrenamiento.



(a) Descenso por gradiente. (b) Superficie de error en entrenamiento.

Figura 5-2. Gráficas del descenso por gradiente y superficie de error en entrenamiento. Fuente: [26]

Es importante tener en consideración el número de datos utilizados en para el entrenamiento: las técnicas de Deep Learning se caracterizan por la gran cantidad de información que la red puede utilizar para su aprendizaje, por lo que errores de "underfitting" por un número limitado de datos no son frecuentes cuando se hace uso de esta tecnología. Por el contrario, sí que es necesario tener en consideración los casos de sobreajuste del modelo,

también conocido como "overfitting", que sucede cuando en el entrenamiento se obtienen resultados poco generalizables a otros casos. Esto es debido a que la red aprende a asociar las entradas y salidas que provienen del set de datos para el entrenamiento, fallando el aprendizaje de la tarea, pues sólo tendrá capacidad de detección el modelo sobre entradas ya procesadas. Se grafica en la Figura 5-3, los errores más comunes del entrenamiento.

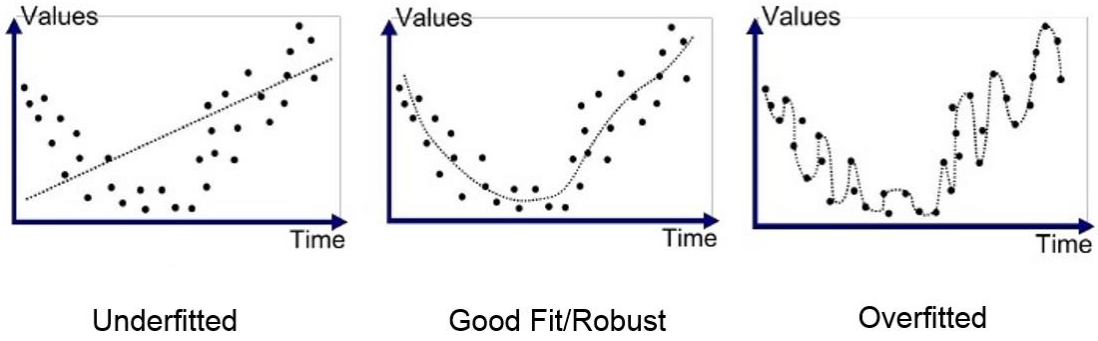


Figura 5-3. Gráficas representativas sobre los errores más comunes durante el entrenamiento. Fuente: [27]

5.1.2 Parámetros influyentes durante el entrenamiento

Se definirán de forma breve los parámetros que guardan una relación directa con respecto al proceso de entrenamiento.

- Tamaño del Batch. Consiste en el número de muestras que serán procesadas por la red antes de actualizar los pesos internos. Es importante lograr un equilibrio sobre el valor dado, debido a que un batch elevado podría derivar en overfitting, y uno demasiado pequeño podría no obtener el rendimiento esperado por la red.
- Epochs o número de épocas. Consiste en las veces que la red procesa el set de datos del entrenamiento. Si el número de datos procesados cuentan con suficiente variabilidad y son representativos, a mayor número de épocas se logra la minimización de la función costes.

$$1\ Epoch = \frac{images\ in\ train\ txt}{batch\ iterations}$$

5.1.3 Parámetros para la evaluación de los resultados obtenidos durante el entrenamiento

Durante el entrenamiento, serán llevados a cabo ciertos cálculos para verificar posteriormente la fiabilidad del modelo CNN obtenido. Destacan, entre otros, a la hora de tomar decisiones sobre la continuidad del entrenamiento, los indicadores mAP, IoU, Precision y Recall.

Para el cálculo de dichas métricas, se introducen los siguientes parámetros:

- TP: True Possitive o Verdaderos Positivos
- FP: False Possitive o Falsos Positivos
- FN: False Negative o Falsos Negativos

Los indicadores Precision, Recall e IoU serán calculados durante el entrenamiento de acuerdo con las fórmulas gráficas de Figura 5-4:

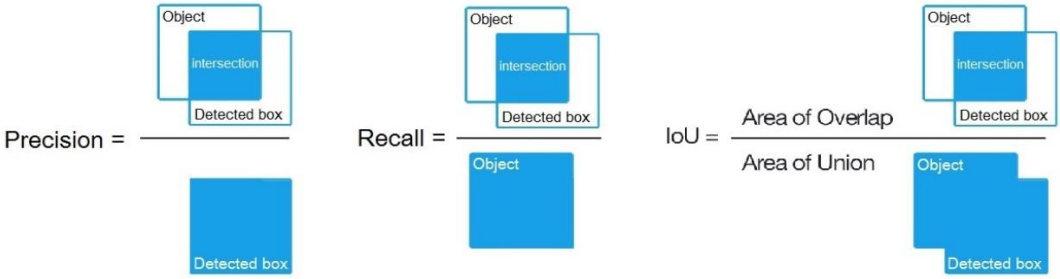


Figura 5-4. Métricas calculadas durante el entrenamiento. Fuente: [20]

La precisión consiste en el cálculo sobre el número correcto de predicciones del modelo.

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

El recall, evaluará por otro lado, la efectividad a la hora de encontrar los positivos

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

La IoU o intersección sobre unión, medirá la intersección promedio sobre unión de objetos (superposición entre cuadro delimitador previsto y real) y detecciones para un cierto umbral. A mayor intersección, mayor valor de la métrica.

En caso de que el valor de la IoU calculado sobre la muestra sea mayor que el umbral establecido, se clasifica la predicción como Verdadero Positivo (TF). En caso contrario, será Falso Positivo (FP). Por tanto, el número de verdaderos o falsos binarios serán dependientes del umbral IoU establecido, que en este caso será del 50%.

Será por tanto mAP, valor medio de Average Precisions para cada clase, donde Average Precisions consiste en el valor medio de 11 puntos en la curva Precision-Recall para cada posible umbral (cada probabilidad de detección) dentro de una misma clase.

$$1\ Epoch = \frac{images\ in\ train\ txt}{batch\ iterations}$$

El parámetro mAP será decisivo para la toma de decisiones sobre la continuidad o no del entrenamiento debido la fiabilidad del mismo sobre el rendimiento de la red, mostrando a tal efecto la precisión media

5.2 Resultados obtenidos con la CNN YoloV3

5.2.1 Resultados y evaluación del entrenamiento

La construcción del complejo sistema de detección se basa en los procedimientos definidos a lo largo de las secciones anteriores, de forma que los resultados de la red obtenida serán fundamentados y basados en los datos pertenecientes al set de entrenamiento. Para una primera aproximación, se utiliza YoloV3 como red neuronal convolucional (CNN) para la detección de objetos.

Relativo a especificaciones, Yolov3 se ejecuta a 320x320 en 22 ms a 28,2 mAP. Alcanza 57,9 mAP @ 50 en 51 ms en una Titan X, en comparación con 57,5 mAP @ 50 en 198 ms de RetinaNet, un rendimiento similar, pero 3,8 veces más rápido. [13]

Por lo general, serán suficientes unas 2.000 iteraciones para cada clase, aunque no menos del número de imágenes de entrenamiento y no menos de 6.000 iteraciones en total. El entrenamiento en este caso es de 12.000 iteraciones. Se hace uso de la técnica de Early Stopping Point con el fin de evitar un sobreajuste, tal y como se recoge en la gráfica cualitativa de la Figura 5-5. Este mecanismo consiste en memorizar el valor de precisión

calculado al final de cada etapa. En caso de que el último valor máximo sea menor que el calculado, significará que el modelo de CNN ha mejorado, por lo que se guardará el nuevo valor de pesos. Se evita de esta forma el sobre-entrenamiento, que da lugar al sobreajuste, caracterizado por capacitar a la red de detectar tan solo dentro de las imágenes incluidas en el set de entrenamiento, impidiendo así la detección de imágenes en el set de validación.

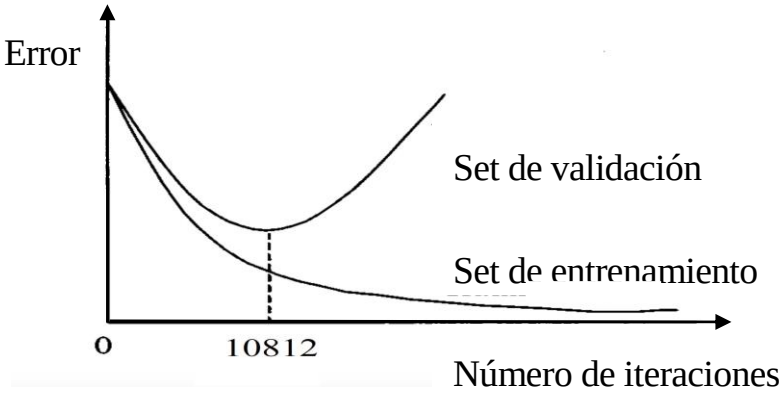


Figura 5-5. Gráfica cualitativa Early Stopping Point en la iteración 10.812.

El mejor resultado ofrecido en esta primera aproximación del modelo perteneciente al primer entrenamiento se obtiene en la iteración 10.812. Se muestra a continuación en la Figura 5-6, la gráfica obtenida durante el primer entrenamiento, enumerando las distintas métricas que se ofrecen como resultados, junto a una breve descripción de cada una.

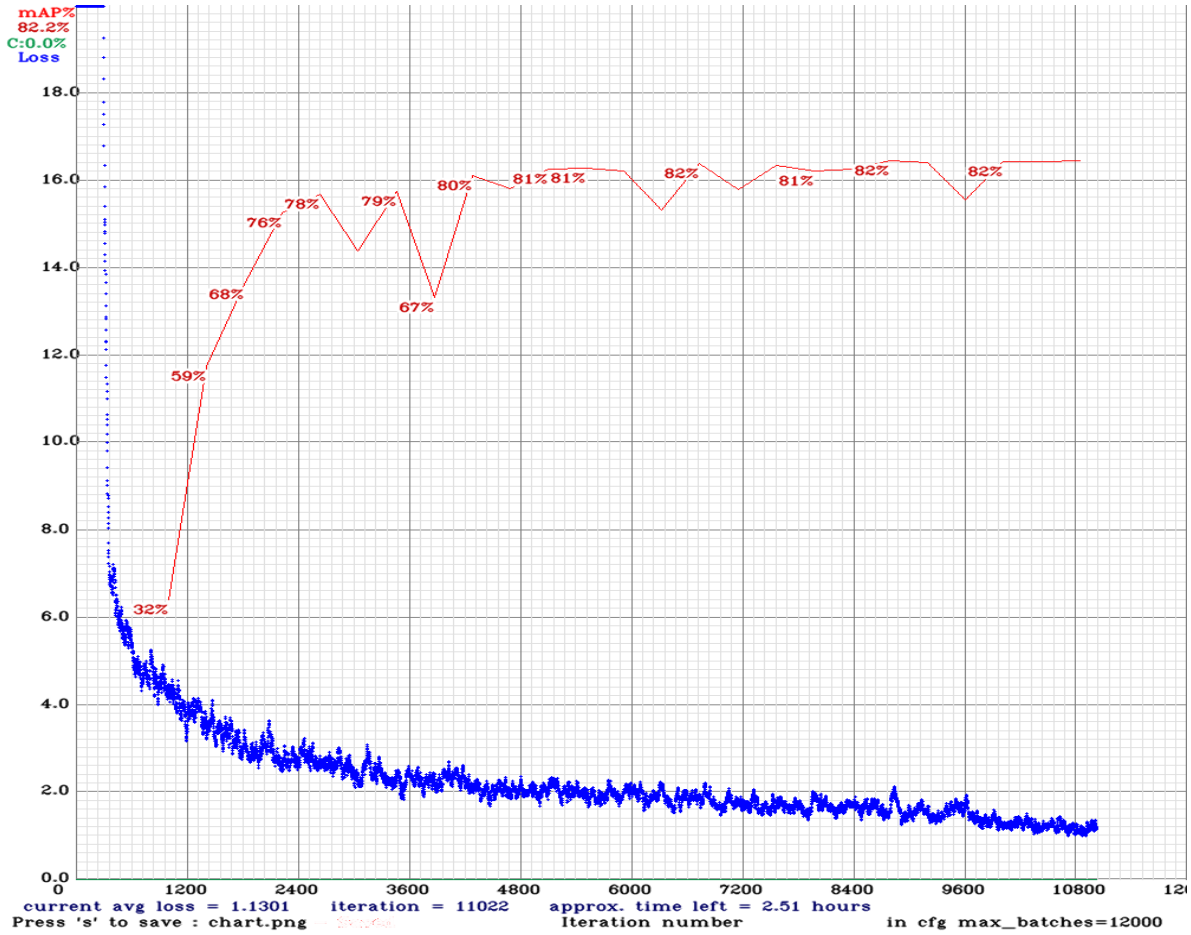


Figura 5-6. Gráfica obtenida durante el primer entrenamiento (CNN base YoloV3).

Generalmente, la pérdida promedio final se suele ubicar dentro de un intervalo que ronda entorno a los límites 0.05 para modelos pequeños y set de entrenamientos sencillos, y 3.0 para modelos grandes con set de entrenamientos más complejos. El valor de la métrica Current Average Loss o error de pérdida media obtenido durante el primer entrenamiento es reducido hasta 1.1301. La disminución de este parámetro es deseable, por lo que será un factor a tener en cuenta a la hora de detener el entrenamiento. También es recomendable realizar un seguimiento de la estabilidad del parámetro, puesto que, en caso de no disminuir su valor durante muchas iteraciones, se debe de dejar de entrenar.

La métrica mAP (mean Average Precision) se ejecutará cada 4 Epochs usando el archivo *valid.txt* especificado en la cuarta línea de *obj.data.txt*, siendo el mismo calculado sobre el set de validación durante el entrenamiento. A tal efecto, se trata de un parámetro que facilita el mapa de probabilidades ofrecido por la red, siendo el más importante a la hora de tomar decisiones a la hora de continuar o no el entrenamiento. A mayor puntuación, el modelo contará con más precisión en sus detecciones, por lo que a medida que aumenta este parámetro, la red ha de seguir siendo entrenada.

Cabe destacar que el set de validación contendrá tan sólo datos sobre conectores empleados, objetos hacia la cual se encuentra centrada esta línea de investigación, por tanto, los resultados de fiabilidad se corresponden con esta clase específicamente.

Finalizado el entrenamiento, se seleccionan los pesos que ofrecen un mejor resultado, relativos al punto de Early Stopping Point, en el directorio Backup de Darknet indicado en el archivo *obj.data.txt*.

5.2.2 Evaluación de pesos obtenidos mediante Early Stopping Point en el primer Entrenamiento

5.2.2.1 Precisión y pérdidas

Se expondrán los resultados obtenidos, en la Tabla 5-1 y Tabla 5-2, una vez seleccionados los pesos pertenecientes a la CNN que proporciona mejores métricas en cuanto a precisión calculada mediante el parámetro mAP sobre el set de validación, y contiene menores pérdidas sobre el set de entrenamiento.

Los pesos seleccionados del sistema de detección serán por tanto validados a la hora de predecir, gracias a la herramienta Test perteneciente a Darknet. Se comprueban y contrastan los resultados que se ofrecerán en un entorno real y cambiante.

Tabla 5-1. Métricas de validación sobre la CNN obtenida con YoloV3.

Especificaciones	Primera aproximación CNN con YoloV3
mAP	82.29%
Pérdidas medias	1.1301
Iteraciones CNN	10800

Tabla 5-2. AP sobre cada una de las clases establecidas con YoloV3.

ID de la Clase	Nombre de la Clase	AP (Precisión media %)
0	Male_VGA_Connector	85.03
1	Male_RJ45_Connector	80.45
2	Female_RJ45_Connector	76.98
3	Male_DVI_Connector	71.09
4	Female_DVI_Connector	89.79
5	Female_VGA_Connector	90.4

5.3 Mejora de resultados

Serán objeto de estudio a lo largo de esta sección, diversas técnicas con la finalidad de aumentar la precisión del modelo. Serán modificados ciertos parámetros y serán aplicadas técnicas de Data Augmentation o Aumento de Datos, a fin de obtener una mayor versatilidad en diferentes escenarios, ampliando el set de datos con las funciones de copia-pega y mosaico. Además de ello, será de nuevo entrenado el mismo set de datos y de validación mediante una CNN que cuenta con mayores prestaciones en cuanto a número de capas y velocidad, YoloV4.

5.3.1 Data Augmentation

La técnica de Aumento de Datos es una de las más utilizadas en este campo, ya que ofrece un aumento notorio de la fiabilidad de la red. Esta labor consiste en procesar cada una de las imágenes, generando X variaciones en cada una de ellas, mediante la aplicación de diversas funciones basadas en principios de volteo, giro, cortes, cambios en la exposición, modificación del color y la saturación, cambio de condiciones de luminosidad, mezcla de imágenes, o fusión de varias imágenes sobre el fotograma original. Pueden apreciarse en la Figura 5-7, algunas de las muestras generadas durante el entrenamiento. Al tratarse, tanto el set de entrenamiento, como el de datos, de imágenes de conectores, funciones como giro cobran sentido en la mejora de datos, puesto que es posible que en un entorno real se encuentre alguno de estos dispositivos girados 180º. La aplicación en otros campos como personas, o animales, suelen carecer de sentido, pues no ofrece ninguna aportación añadir una variación de rotación sobre la muestra debido a que rara vez se encuentran personas o animales al revés.

Mosaico representa un nuevo método de aumento de datos que mezcla 4 imágenes de entrenamiento. Mezcla de recortes, en cambio sólo unirá 2 imágenes de entrada. Por otro lado, MixUp mezclará en un mismo fondo dos fotogramas diferentes. La aplicación de estas técnicas, unidas a las especificadas con anterioridad, permitirá la detección de objetos fuera de su contexto normal, ampliando por tanto el Dataset utilizado para el entrenamiento, y adaptando la detección de la red en diferentes entornos.

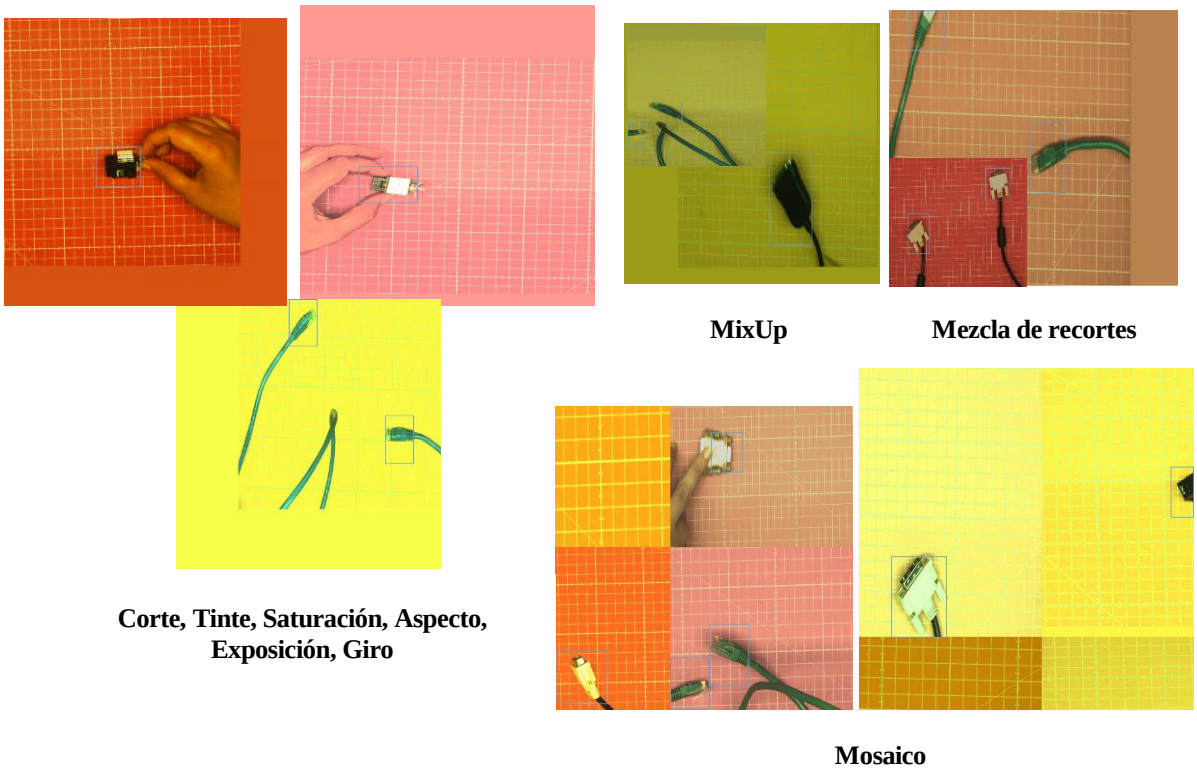


Figura 5-7. Muestras pertenecientes al set de datos aplicadas las técnicas de Data Augmentation.

5.3.2 Resultados del entrenamiento obtenido con YoloV4

La versión YoloV4 de Darket consiste en una red neuronal más rápida y precisa que las CNN en tiempo real de Google TensorFlow, EfficientDet y FaceBook Pytorch/Detectron RetinaNet/MaskRCNN en el conjunto de datos de Microsoft COCO. [17]

Tanto el set de validación como el set de datos serán idénticos al utilizado para el primer entrenamiento con Yolov3, a diferencia de la aplicación de las funciones pertenecientes a las técnicas de Data Augmentation. Se adjunta en la Figura 5-8, los resultados obtenidos durante el segundo entrenamiento con las mejoras pertinentes.

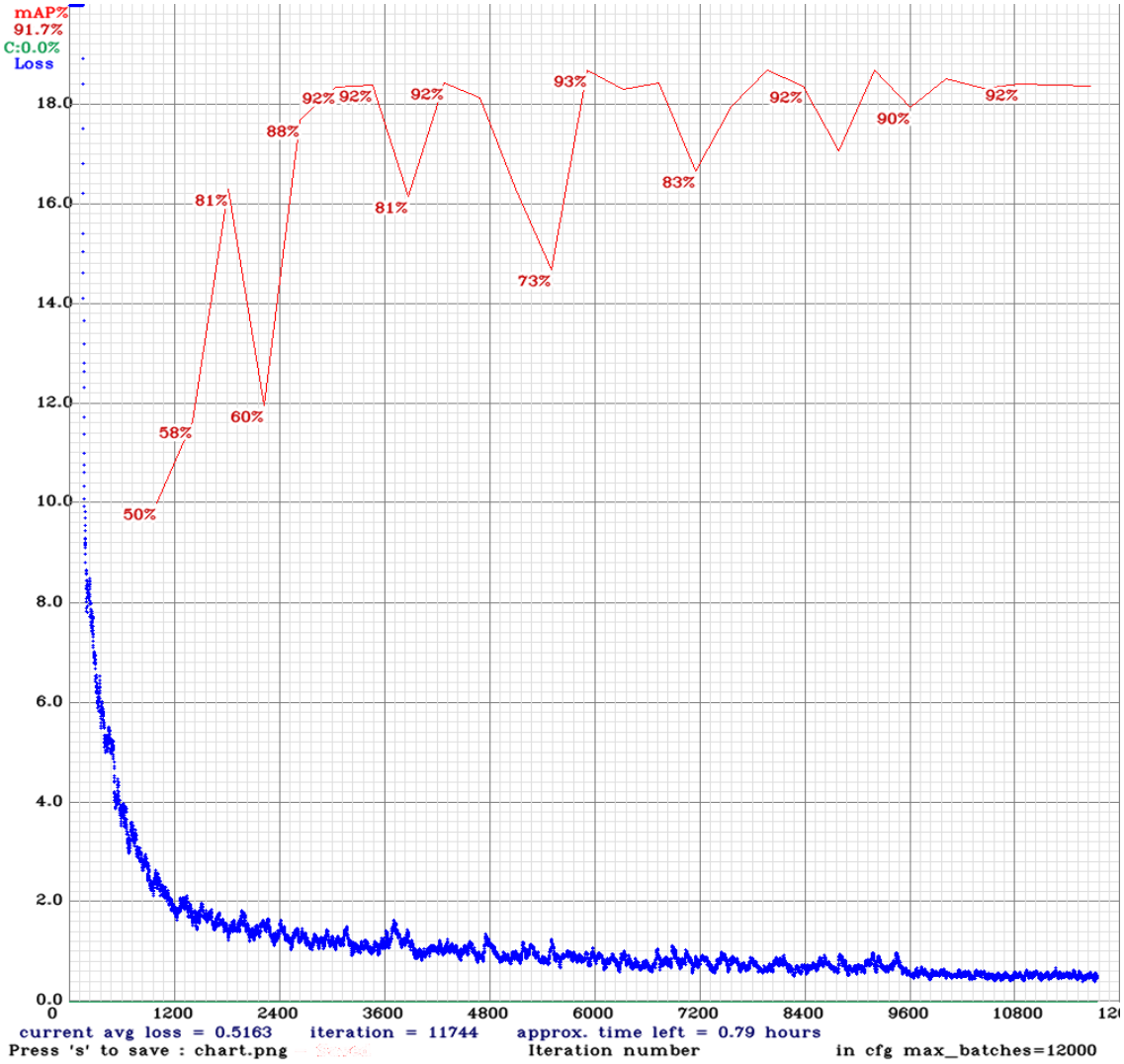


Figura 5-8. Gráfica obtenida durante el segundo entrenamiento (YoloV4 y Data Augmentation).

5.3.2.1 Precisión y pérdidas

Serán analizadas de forma más exhaustiva las métricas sobre el modelo mejorado, debido a que serán éstos los pesos de la red neuronal final utilizada para la integración del proyecto en la HMI, que tendrá cabida en el mercado. Éstas se encuentran recogidas, generalmente en la Tabla 5-3 y en Tabla 5-4, además de la sección dedicada a resultados finales conclusivos.

Tabla 5-3. Métricas de validación sobre la CNN obtenida con YoloV4.

Especificaciones	Segunda Aproximación CNN con YoloV4
mAP	93.54%
Recall	0.92
Precisión	0.78
Pérdidas medias	0.77
Iteraciones CNN	5998

Tabla 5-4. AP sobre cada una de las clases establecidas con YoloV4

ID de la Clase	Nombre de la Clase	AP (Precisión media %)
0	Male_VGA_Connector	89.58
1	Male_RJ45_Connector	96.50
2	Female_RJ45_Connector	93.53
3	Male_DVI_Connector	95.56
4	Female_DVI_Connector	97.49
5	Female_VGA_Connector	88.58

5.4 Resultados finales

Tras el análisis de los diferentes entrenamientos, es importante volver insistir en que, tanto el set de datos de entrenamiento, como el set de datos para la validación, han consistido en secuencias de imágenes reales de conectores.

Por tanto, los resultados obtenidos actúan en concordancia con la fiabilidad para la detección de estas clases. Generalmente, muestran una notoria mejora gracias a la implementación de técnicas de aumento de datos y al entrenamiento de la red neuronal convolucional YoloV4 (véase Figura 5-10), lo que supone en gran medida el porcentaje de acierto del 93% aproximadamente. Frente a datos reales, por tanto, se cumple con los objetivos establecidos para la optimización del proceso de ensamblado e integración, reduciendo el factor error del humano y ofreciendo soporte para mejora de experiencia del usuario.

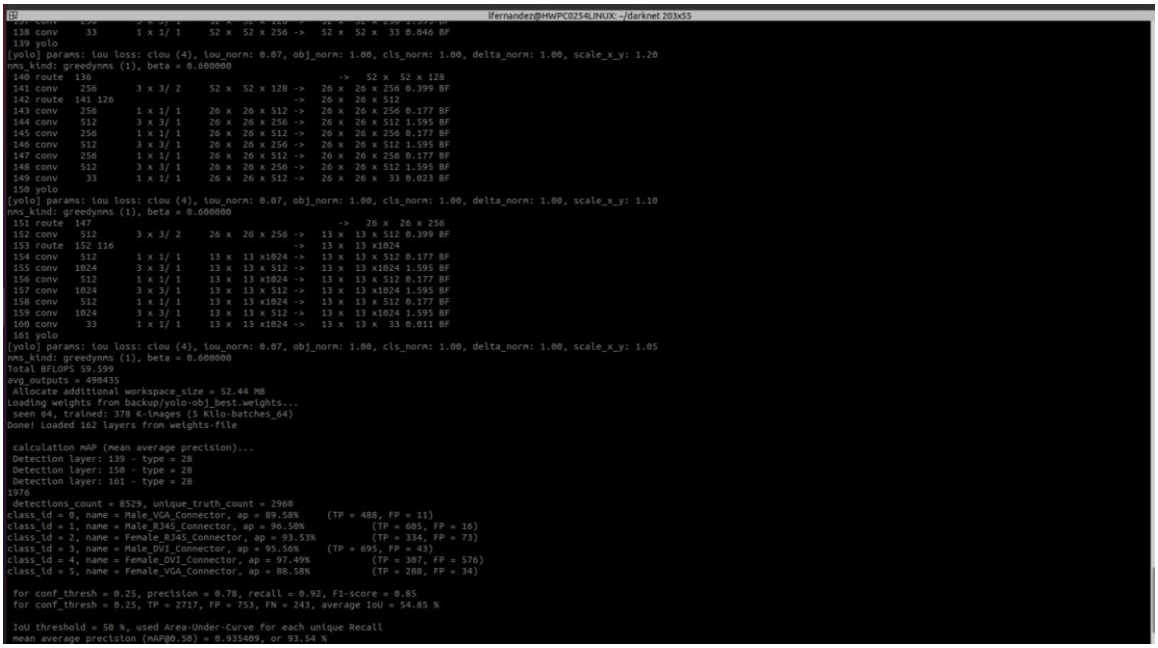


Figura 5-9. Validación de precisión y pérdidas sobre los pesos finales seleccionados.

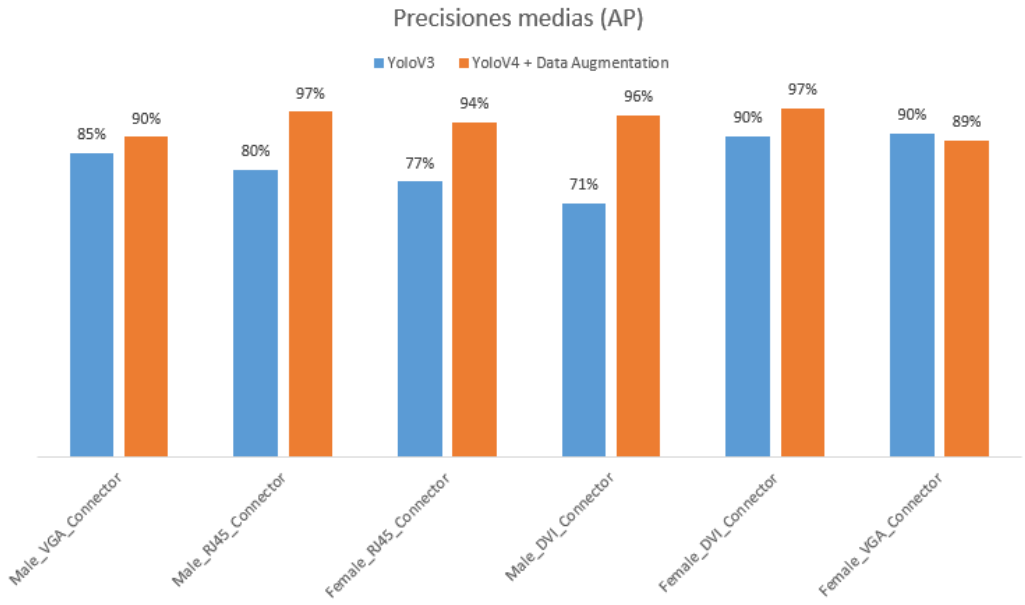


Figura 5-10. Comparativa sobre Precisión Media (AP) obtenida en cada uno de los entrenamientos.

5.5 Tiempos de detección del modelo final

Será utilizada la herramienta de Darknet Detector Test para evaluar los tiempos de detección. A tal efecto se llama a la función y se pasa como parámetro una fotografía ejemplo. Se procesa la red neuronal y la salida del sistema consiste en la probabilidad de acierto de esta, y los tiempos de detección empleados para dicha labor.

La media de predicción por fotograma será de aproximadamente 31 milisegundos, lo cual permitirá trabajar al detector con flujo de imágenes en tiempo real en la interfaz que se desarrollará posteriormente.

Se muestran en la Figura 5-11 algunos de los resultados individuales a modo de ejemplo, obtenidos sobre el modelo CNN mejorado con YoloV4 y Data Augmentation. Cabe destacar que los tiempos obtenidos con la versión anterior, son mínimamente mayores, aunque podrían seguir siendo válidos para aplicaciones en tiempo real.

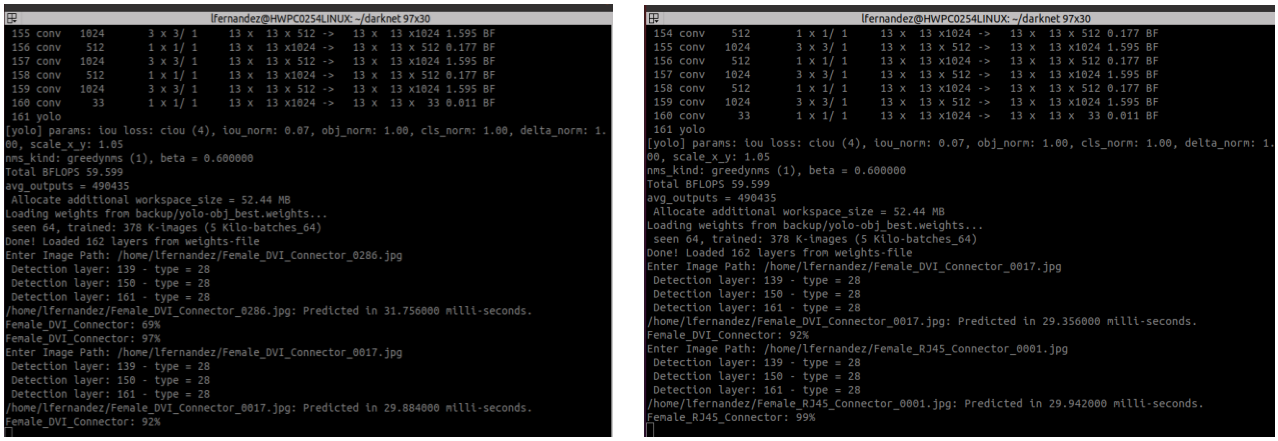


Figura 5-11. Detector Test. Tiempos de detección.

6 INTERFAZ HOMBRE-MÁQUINA

Finalmente, se integrará la funcionalidad de la cámara y se implementará Darknet como librería con la finalidad de elaborar un software propio para cumplir con los objetivos propuestos. La integración de ambas herramientas supondrá una interfaz de programación que dará lugar a una herramienta visual que ejercerá como soporte y guía durante los procesos de fabricación y ensamblado del satélite. Esto transformará la experiencia del usuario y reducirá los errores debido al factor humano, siendo de gran ayuda en un amplio abanico de utilidades aplicadas a las nuevas necesidades de la industria.

En líneas técnicas relativas al software para implementación de la CNN para este primer prototipo, se compilará Yolo como librería compartida en C++, construyendo *yolo_cpp_dll*. Darknet será incluido como 3rd party.

Además, será generado un archivo configuración .ini. Su funcionalidad consiste en proporcionar al código principal las rutas de los pesos y los archivos de configuración previamente evaluados y seleccionados como los que cuentan con una mayor fiabilidad para satisfacer las necesidades entorno al demostrador. Estas rutas hacia los archivos relacionados con el modelo de detección serán conexionadas para el uso de los archivos por la librería de Darknet.

En lo relativo al dispositivo BB-500GE, se hará uso de la programación software ya detallada en la Sección 4, donde se establece la conexión entre dispositivos, la configuración, y toma de secuencia de datos. Todo ello será reutilizado de nuevo para la integración en la HMI.

La conexión de todos los elementos que componen el conjunto supondrá el eje central del desarrollo software. El código que recoge estas funcionalidades será capaz de tomar fotogramas, cargar la red neuronal convolucional, y procesar la identificación de los objetos, imprimiendo por pantalla la clase correspondiente y la probabilidad de acierto sobre dicha clase en tiempo real.

El desarrollo del código fuente estará estructurado de forma que, inicialmente, se incluyen todas las librerías, tanto propias, como necesarias para el funcionamiento de Darknet, como para el SDK del dispositivo BB-500GE.

Dentro de la sección principal, será leídos el archivo de configuración .ini, y se extraerán los *yolo.weights*, *yolo.cfg* y *yolo.clases* indicados en las rutas del archivo.

A partir de este punto, existen dos ramas alternativas, cada una con una funcionalidad diferente.

- Generación del set de datos. Generalmente, esta rama consiste en la implementación del desarrollo comentado para explorar las funcionalidades de la cámara en la Sección 4. De tal forma serán generados tanto el set de datos como el de validación. La funcionalidad de esta rama consiste en el almacenaje de fotogramas numerados linealmente en el tiempo. Se creará una carpeta en el directorio indicado con el nombre establecido para la pieza.
- Herramienta visual para la detección en tiempo real. Esta rama vertebrará el groso de la interfaz, llamando a la librería Darknet para ejecutar la detección en tiempo real de los objetos. Será cargada la CNN capaz de detectar las diferentes clases de conectores seleccionada para tal labor, al pasar a la clase Detector los pesos y archivos de configuración necesarios proporcionados por el archivo .ini. Se instancia la clase relativa al dispositivo BB-500GE, comprobando que su configuración sólo se ejecuta una vez, y comienza la toma de fotogramas que muestren el escenario en tiempo real. Este flujo será procesado por la clase Detections con un umbral de mínimo el 60% de acierto. El cuadro limitador sobre la clase detectada indicará el nombre y el porcentaje de fiabilidad de la pieza en escena.

A efecto visual, será mostrado a través de la estación por pantalla los conectores en escena en ese momento con su correspondiente identificación, clasificado según el nombre de la clase detectada y probabilidad de acierto, todo ello en tiempo real.

Se adjunta en las Figura 6-1 y 6-2 documentación gráfica que muestra dicho funcionamiento.

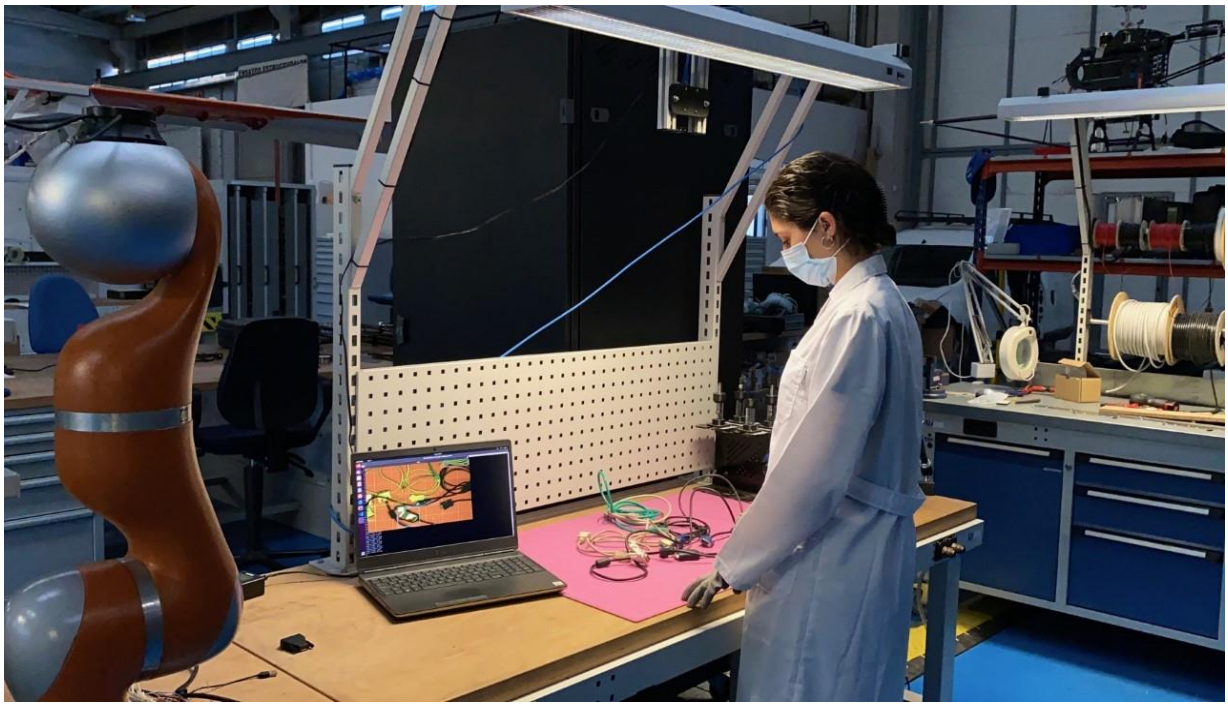


Figura 6-1. HMI. Procesamiento en tiempo real de la CNN e identificación de conectores junto a la probabilidad de acierto de la clase.

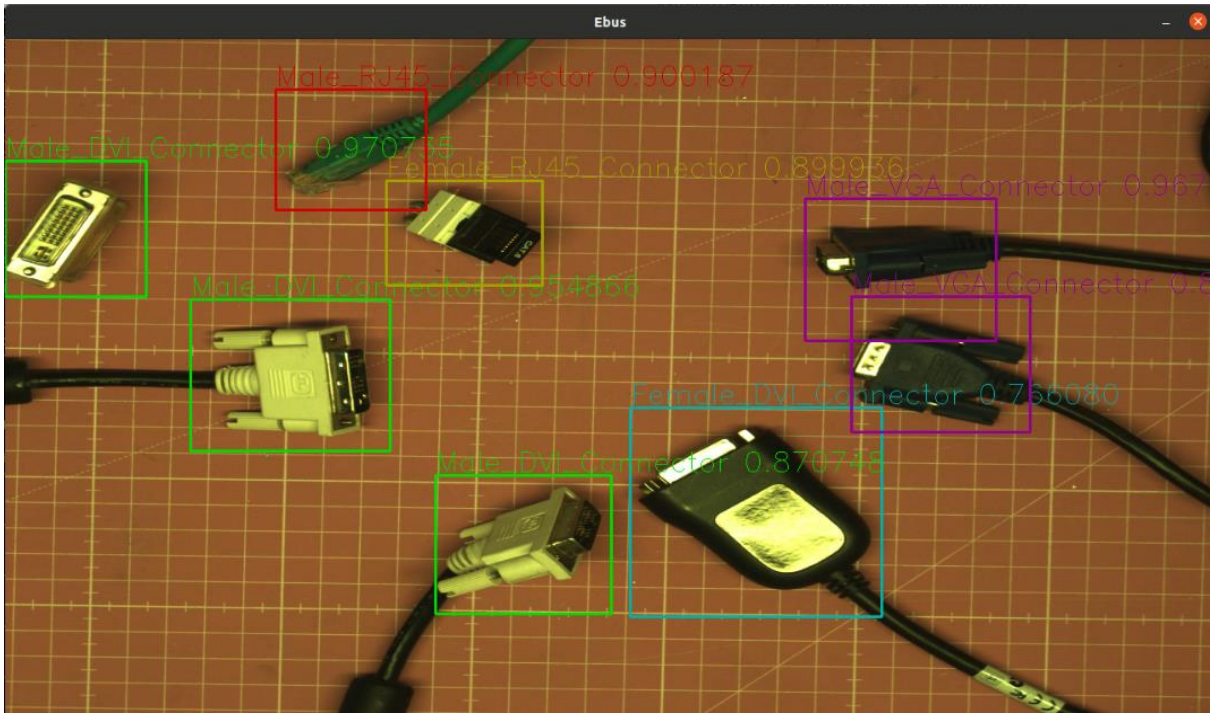


Figura 6-2. Asistencia al operario a través de la HMI creada.

7 LÍNEAS DE FUTURAS MEJORAS

De forma general, la línea de investigación que sigue el Trabajo de Fin de Grado cuenta con una amplia gama de posibles mejoras y proyecciones futuras, hasta la obtención de un sistema robusto que cumpla con las exigencias requeridas y los elevados estándares de conforman las aplicaciones abarcadas dentro del campo espacial.

Cabe destacar que los conocimientos previos en materia de Redes Neuronales por parte de la alumna eran prácticamente inexistentes, por lo que este proyecto ha supuesto en sus inicios una primera toma de contacto con mundo de la Inteligencia Artificial, y más concretamente del Aprendizaje Profundo. Además, a lo largo del proceso, se han adquirido numerosos conocimientos sobre tratamiento de imagen, e implementación, comunicación y desarrollo de las utilidades ofrecidas por las cámaras industriales. Por otro lado, la integración del conjunto en C++ ha supuesto un reto, pues tampoco la alumna había experimentado una formación previa y/o experiencia con este lenguaje de programación. Por tanto, mejoras en la optimización del código serán contempladas en la guía de posibles progresos.

Debido al rápido crecimiento y el amplio abanico de posibilidades a la hora de abarcar este reto, son de aplicación diferentes mejoras relativas tanto a la arquitectura de sistema integrado, como al entrenamiento y elaboración de la red neuronal. Se contaban con ciertas limitaciones a la hora de trabajar con un conjunto tan elevado de datos, y manejar tiempos de computación durante entrenamientos tan largos. Por tanto, al tratarse de una primera aproximación del prototipo, el número de clases para los que se ha entrenado la red era reducido. Una línea futura de mejora sería, por tanto, el aumento del número de conectores capaces de ser identificados por este sistema inteligente, y la similitud de los mismos con respecto a los de aplicación en la Agencia Espacial Europea para el proceso real de manufactura del satélite.

A la utilidad ofrecida por la Interfaz Humano-Máquina, se le podrían incluir a su vez diversas funcionalidades de sistemas de ayuda al operario adaptado a diferentes escenarios, que indiquen qué conector se ha de emplear dependiendo de ciertas circunstancias, realizando previamente un análisis predictivo sobre experiencias reales. La integración del dispositivo en un sistema de realidad aumentada es viable, aunque dependería de la velocidad de procesamiento con la que contase la estación que sustituyese al actual equipo, ya que para la inserción de esta tecnología en equipos móviles sería necesaria la reducción del hardware empleado. El coste ascendería notablemente para que la herramienta tuviese capacidad de procesamiento en tiempo real integrada en sistemas móviles.

Por otro lado, la inclusión de un dispositivo láser que proyecte la localización del conector, o sea capaz de detectar defectos en los mismos y errores procedentes del factor humano existente durante el proyecto de ensamblaje, compondrían una posible línea futura para la investigación en el detector.

8 CONCLUSIONES

Este proyecto fundamenta un demostrador de tecnología cuyo objetivo pretende cimentar unas bases para la propuesta lanzada por la ESA, demostrando la viabilidad del mismo para la asistencia al operario. Sobre los objetivos propuestos orientados a la identificación en tiempo real, es remarcable el cumplimiento de estos sobre la implantación del detector seguidor de conectores para el guiado de operarios en el proceso de ensamblado de satélites de la Agencia Espacial Europea.

A lo largo del TFG, se han llevado a cabo el diseño, implementación, y evaluación de un sistema de detección de conectores en imágenes de profundidad basado en CNNs. Partiendo de la revisión bibliográfica y el estado del arte contemplado en la Sección 2, se construye sobre la base de conocimientos teóricos adquiridos una red neuronal profunda, llevando a cabo su correspondiente entrenamiento sobre un set de datos real tomados por el dispositivo de captura industrial, y el consiguiente etiquetado automático de los fotogramas recolectados del entorno. El porcentaje dictaminado por el mAP obtenido inicialmente contaba con un 82.29% de fiabilidad entorno a la identificación y seguimiento de los conectores empleados durante el primer entrenamiento. Tras la aplicación de las diversas técnicas de Aumento de Datos y el entrenamiento sobre una red mejorada, gracias a la inclusión de reguladores, optimizadores y un aumento de número de capas ocultas, se aumentan las métricas obtenidas durante el segundo entrenamiento, aludiendo a un porcentaje del 93.54% de precisión sobre los pesos obtenidos gracias a las técnicas de Early Stopping Point. Los tiempos de detección, alegan a su vez la capacidad de trabajo de la red en materia de localización en tiempo real.

Se cohesionan en la interfaz hombre-máquina diseñada en C++, las diferentes funcionalidades desarrolladas a lo largo del proceso, mostrando un comportamiento deseado en el guiado en tiempo real del operario. Dichas funcionalidades incluyen tanto la implementación del desarrollo del SDK y la integración del dispositivo BB-500GE, orientado a captura en tiempo real de fotogramas y elaboración del set de datos, como los resultados obtenidos del entrenamiento de Red Neuronal Convolucional, y la librería de Darknet utilizada como Detector. A tal efecto, se alega con el cierre del siguiente documento el cumplimiento sobre las expectativas en relación con el demostrador desarrollado.

A modo de conclusiones alcanzadas en relación con el dispositivo BB-500GE, se reconoce el crucial factor de estandarización de protocolos para la transmisión de datos con la estación. Por otro lado, respecto a la viabilidad del proyecto en otros ámbitos, se recalca el auge de los sistemas de localización y seguimiento de objetos en la disciplina de Aprendizaje Profundo, así como la robótica cognitiva. Estos motores de cambio conforman un factor estratégico en el desarrollo de mercados en relación con la navegación por satélite, aumentando significativamente las inversiones en esta materia.

El presupuesto empleado para el desarrollo del sistema no es excesivamente elevado en relación coste-beneficio, por lo que, a tal efecto, la optimización de operaciones con orquestación impulsada por IA y manejo automatizado de errores podría satisfacer las crecientes necesidades del mercado entorno a la industrialización de pequeñas y medianas empresas. La versatilidad ofrecida por el demostrador de tecnología podría ser a su vez enlazado con las emergentes necesidades de la Industria 4.0, y consolidar un paso adelante hacia el concepto de conectividad propuesto por el incipiente concepto de Industria 5.0 [28]. La amplia cabida de mercado con la que contaría el desarrollo y entrenamiento de un algoritmo de detección inteligente es crítico para ampliar horizontes sobre líneas de investigación entorno a campos como la realidad aumentada, tareas de inspección, segmentación semántica en vehículos autónomos y posible uso para tareas de localización y mapeo simultáneo, así como procesos de montaje, ensamblado, supervisión y tracking de mercancías.

REFERENCIAS

[1] Naciones Unidas, «Desafíos Globales. Población,» 2017. [En línea]. Available: https://www.un.org/es/global-issues/population.

[2] A. I. Quirós Gámez y G. Andreadis, «Prospective analysis of the impact of a pandemic in Industry 4.0,» *MATEC Web of Conferences*, 2020.

[3] The European Space Agency, «¿Qué es Galileo?,» 2011. [En línea]. Available: https://www.esa.int/Space_in_Member_States/Spain/Que_es_Galileo.

[4] NASA, «SpaceX kick-starts global 2020 launch year with Starlink mission,» 2020. [En línea]. Available: https://www.nasaspaceflight.com/2020/01/spacex-kick-start-global-2020-starlink/.

[5] D. Messier, «SpaceX Wants to Launch 12,000 Satellites,» 2017. [En línea]. Available: http://www.parabolicarc.com/2017/03/03/spacex-launch-12000-satellites/.

[6] WIKIPEDIA, «Starlink,» 2021. [En línea]. Available: https://en.wikipedia.org/wiki/Starlink.

[7] Surrey Satellite Technology, «SSTL celebrates Galileo navigation payload order,» 2017. [En línea]. Available: https://www.sstl.co.uk/media-hub/latest-news/2017/sstl-celebrates-galileo-navigation-payload-order.

[8] T. M. T. Md Zahangir Alom, C. Yakopc, P. S. Stefan Westberg, M. S. Nasrin, B. C. V. Esesn, A. A. S. Awwal y V. K. Asari, «The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches,» 2018.

[9] F. S. Caparrini, «Redes Neuronales: una visión superficial,» p. http://www.cs.us.es/~fsancho/?e=72, 2019.

[10] WIKIPEDIA, «Método de Kernel,» 2021.

[11] S. Saha, «A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,» 2018. [En línea]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[12] S. Ren, K. He y R. Girshick, «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,» *IEEE*, 2017.

[13] J. Redmon y A. Farhadi, «YOLOv3: An Incremental Improvement,» de *arXiv*, 2018.

[14] R. Girshick, J. Donahue, T. Darrell y J. Malik, «Rich feature hierarchies for accurate object detection and semantic segmentation,» *arXiv*, 2013.

[15] R. Girshick, «Fast R-CNN,» *arXiv*, 2015.

[16] K. He, G. Gkioxari, P. Dollar y R. Girshick, «Mask R-CNN,» *arXiv*, 2018.

[17] A. Bochkovskiy, C.-Y. Wang y H.-Y. M. Liao, «YOLOv4: Optimal Speed and Accuracy of Object

Detection,» de *arXiv*, 2020.

[18] Pleora Technologies Inc, «eBUS Player User Guide,» 2021. [En línea].

[19] Pleora Technologies Inc, «eBUS SDK Programmer's Guide,» 2021. [En línea].

[20] AlexeyAB, «Darknet,» 2020. [En línea]. Available: https://github.com/AlexeyAB/darknet.

[21] NVIDIA, «CUDA,» 2021. [En línea]. Available: https://developer.nvidia.com/cuda-downloads.

[22] NVIDIA, «cuDNN,» 2021. [En línea]. Available: https://developer.nvidia.com/cudnn.

[23] K. Pulli, A. Baksheev, K. Kornyakov y V. Eruhimov, «Realtime Computer Vision with OpenCV,» 2012. [En línea].

[24] AlexeyAB, «YOLO Mark,» 2019. [En línea]. Available: https://github.com/AlexeyAB/Yolo_mark.

[25] Syllo, «NVTOP,» 2021. [En línea]. Available: https://github.com/Syllo/nvtop.

[26] W. Jaber, «Detection et diagnostic des defaillances des procedes chimiques a l'aide des reseaux neuronaux artificiels v2.0,» *Oral Probatoire Wael JABER*, 2018.

[27] J. Orellana Alvear, «Arboles de decision y Random Forest,» 2018. [En línea]. Available: https://bookdown.org/content/2031/arboles-de-decision-parte-i.html#conceptos-introductorios.

[28] Ö. Önday, «Japan's Society 5.0: Going Beyond Industry 4.0,» 2019.

[29] J. Redmon, «Darknet: Open source neural networks in c.,» 2013-2016. [En línea]. Available: http://pjreddie.com/darknet/.

[30] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. Zitnick y P. Dollár., «Microsoft COCO: Common Objects in Context,» de *arXiv:1405.0312*, 2015.

[31] «CMAKE,» [En línea]. Available: cmake.org.

[32] G. A. Blog, «Exploring Weight Agnostic Neural Networks,» 2019. [En línea]. Available: https://ai.googleblog.com/2019/08/exploring-weight-agnostic-neural.html.

[33] Google, «Exploring Weight Agnostic Neural Networks,» [En línea]. Available: https://ai.googleblog.com/2019/08/exploring-weight-agnostic-neural.html.

[34] N. Dalal y B. Triggs, «Histograms of oriented gradients for human detection,» *IEEE*, 2005.

[35] R. Benenson, M. Mathias, R. Timofte y L. V. Gool, «Histograms of oriented gradients for human detection,» *IEEE*, 2012.

[36] P. Dollár, P. P. Z. Tu y S. Belongie., «Integral channel features,» *BMVC*, 2009.

[37] P. Zheng, S. Meng, C. Chen y K. Xu, «Satellite Assembly Process Optimization Based on Digital,» *2nd International Symposium on Resource Exploration and Environmental Science*, 2018.

[38] NVIDIA, «CUDA TOOLKIT DOCUMENTATION,» [En línea]. Available: https://docs.nvidia.com/cuda/archive/11.0/.

[39] The European Space Agency, «Galileo satellite engineering model platform integration tests completed,» [En línea]. Available: https://www.esa.int/Applications/Navigation/Galileo_satellite_engineering_model_platform_integration_tests_completed.

[40] NASA, «Assembly of the Lunar Reconnaissance Orbiter (LRO),» 2008. [En línea]. Available: https://svs.gsfc.nasa.gov/10326.

[41] S. Ren, K. He, R. Girshick y J. Sun, «Faster R-CNN: Towards Real-Time Object,» *arXiv*, 2016.

[42] M. Wall, «SpaceX's Prototype Internet Satellites Are Up and Running,» 2018. [En línea]. Available: https://www.space.com/39785-spacex-internet-satellites-starlink-constellation.html.

[43] The European Space Agency, «About Materials and Processes,» [En línea]. Available: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/About_Materials_and_Processes.

[44] S. KIM y H. Kim, «Zero-Centered Fixed-Point Quantization with Iterative Retraining for Deep Convolutional Neural Network-Based Object Detectors,» *IEEE*, 2021.