UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

"Entrenamiento, optimización y validación de una CNN para la localización de un robot móvil mediante tareas de clasificación y regresión".

TRABAJO DE FIN DE GRADO

Septiembre - 2021

AUTOR: Juan José Cabrera Mora

DIRECTOR/ES: Luis Payá Castelló
Sergio Cebollada López

Contenido

Lista de Figuras

Lista de tablas

1 Introducción

En la actualidad, la robótica es una tecnología muy presente en nuestra vida cotidiana. La podemos encontrar de forma directa en nuestras viviendas en caso de poseer un robot de limpieza o de forma indirecta en aquellos productos que han requerido de la intervención de robots para llevar a cabo su proceso productivo, como es el caso de un automóvil.

La robótica es una tecnología relativamente joven que reúne diferentes campos de la ingeniería para la creación de robots. A pesar de su juventud, esta tecnología es ampliamente empleada en industria, sector servicios, sanidad, aplicaciones militares y espaciales entre otros.

En industria se hace uso de brazos robóticos que forman parte de cadenas de producción (Figura **1-1**), por otro lado existen robots colaborativos que son capaces de trabajar junto a un humano sin poner en peligro la integridad física del mismo (Figura **1-2**). La idea es que estos robots realicen tareas repetitivas, físicamente exigentes y/o peligrosas para las personas.

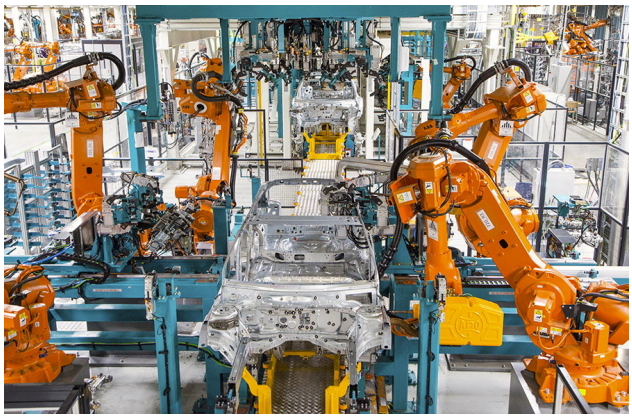


Figura 1-1: Robot industrial.

Figura 1-2: Cobot de Universal Robots.

En el sector servicios se están desarrollando robots que pueden interactuar con personas y realizar trabajos de hostelería como servir las mesas de un restaurante (Figura **1-3**), incluso robots manipuladores móviles empleados en logística para organizar y transportar cajas de forma autónoma (Figura **1-4**).

Figura 1-3: Robot camarero.

Figura 1-4: Robot Handle diseñado por la empresa estadounidense Boston Dynamics.

La robótica autónoma móvil es una rama de la robótica que estudia robots capaces de navegar en un entorno y además realizar la tarea para la cual fueron diseñados sin la intervención de un humano. El desarrollo de robots autónomos móviles se ha incrementado de forma exponencial en los últimos años debido a su gran variedad de aplicaciones para automatizar procesos y/o realizar tareas sin poner en riesgo a personas. Se ha demostrado con diferentes proyectos que se pueden destinar en la exploración de entornos desconocidos y peligrosos, desde las profundidades del oceano (Figura **1-5**) hasta lugares tan lejanos como Marte (Figura **1-6**).

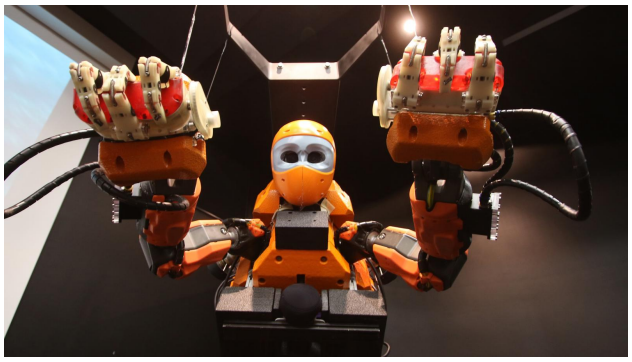


Figura 1-5: OceanOne diseñado por la Universidad de Stanford.

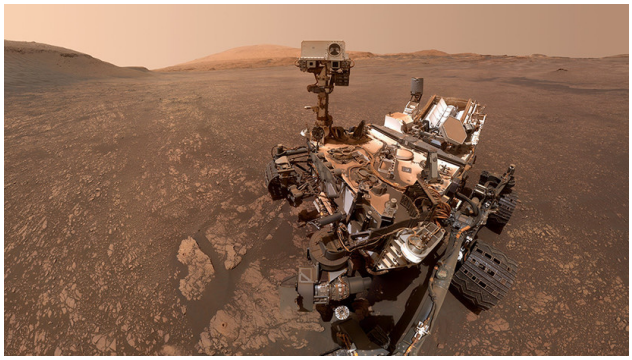


Figura 1-6: Curiosity diseñado por la NASA.

Un robot móvil ha de ser capaz de navegar en el entorno de trabajo. Para ello, debe ser capaz de construir un mapa del mismo y además, estimar su pose (posición y orientación) de forma suficientemente precisa. Durante la realización de estas tareas es necesario tener en cuenta que, en numerosas ocasiones, el robot debe ser capaz de operar en entornos complejos y cambiantes, que pueden cambiar la apariencia del entorno de trabajo. A la hora

de disenar un sistema para construcción de mapas y localización, podemos optar bien por resolver el problema de forma métrica o bien de forma topológica. Un mapa métrico contiene información de la posición de ciertas características del entorno respecto de un sistema de referencia, con una determinada incertidumbre asociada. En este caso, el robot sera capaz de estimar su posición con precisión geométrica respecto dicho sistema de referencia. Por su parte, los mapas topologicos son una representación del mundo que contiene únicamente varias localizaciones y las relaciones de conectividad entre ellas. Este tipo de mapas son un mecanismo eficiente para que el robot se localice con la precisión suficiente y planifique sus trayectorias.

Para poder construir un modelo del entorno y estimar su posición, el robot debe ir equipado con sensores que permitan conocer el estado de entorno respecto al robot. Entre los diferentes sensores empleados en robótica móvil podemos destacar:

- **GPS**: Permite conocer el posicionamiento del robot dentro del planteta Tierra, su principal desventaja reside en el elevado margen de error, de 10 metros. En entornos cercanos a edificios y/o árboles los errores de medida se incrementan y requieren de la combinación con otras técnologías como la odometría para obtener un sistema de localización robusto tal y como explica Onho *et al.*, [34]. Además, el GPS no se puede utilizar en interiores.

- **Láser**: Es un sensor que permite medir la distancia a los elementos del entorno, y junto con los algoritmos de procesamiento adecuados permite tanto construir modelos del entorno como localizar al robot. Uno de los sistemas más importantes que emplean la tecnología láser es el LiDAR, que permite medir distancias a todos los elementos que se encuentran dentro de un determinado ángulo frente al robot. Como inconveniente, las condiciones meteorológicas adversas pueden provocar que los rayos se dispersen y que las medidas tomadas no sean buenas. Este sistema se ha empleado en numerosas aplicaciones en robótica móvil y algunos autores como Wolcott *et al.*, [59] recurren a métodos probabilisticos para conseguir una herramienta robusta frente a condiciones adversas como fuertes nevadas.

- **Sensores de proximidad**: Permiten la detección de obstáculos. Existen varios tipos de sensores en función del principio físico que utilizan. En el ámbito de los robots móviles autónomos podemos destacar:

    - **Infrarrojos**: Detección de objetos mediante luz, suelen emplear un fototransistor o un fotodiodo. Martín *et al.*, [30] desarrollaron un sistema sensorial para medir los desplazamientos diferenciales de fase en una señal infrarroja modulada sinusoidalmente transmitida desde el robot. Se obtuvieron distancias diferenciales de estos desplazamientos de fase, y la posición del robot se estimó mediante trilateración hiperbólica.

  - **Ultrasonidos**: El sensor emite una onda y recibe el eco producido al impactar con un objeto. Moreno *et al.*, [32] integraron sensores ultrasónicos con un filtro no lineal basado en un algoritmo genético como método de optimización logrando así la implementación de esta técnica en sistemas de navegación más complejos.

- **Sensores de visión**: El empleo de cámaras como sensores en robótica móvil ha permitido la resolución de numerosas tareas, gracias a la gran cantidad de información que pueden capturar del entorno. Las cámaras posibilitan conocer ampliamente el entorno del robot pudiendo identificar objetos y determinar distancias. Hay distintos tipos de cámaras entre los cuales podemos destacar:
  - **Cámara estándar**: Se trata de la cámaras que podríamos tener en nuestros dispositivos móviles. Las imágenes obtenidas con este tipo de cámaras no permiten determinar profundidades o distancias dado que disponen de una sola cámara. Pero si que pueden ser empleadas para otras tareas como el reconocimiento facial, clasificación de objetos y detección de los mismos. Es común la combinación de una o más cámaras estándar dispuestas en múltiples direcciones para cubrir todo el entorno del robot. Un ejemplo de esta técnica es la que emplea el Autopilot de Tesla, combinando 8 cámaras de diferentes alcances con 12 sensores ultrasónicos y un radar. En cuanto a la tarea de mapping y localización, este tipo de cámaras sólo se podrían usar para crear mapas topológicos, y no métricos.
  - **Cámara estereoscópica**: Lleva a cabo la composición de imágenes 3D mediante el empleo de dos o más cámaras separadas una cierta distancia, esto permite extraer la información tridimensional de un objeto visto desde diferentes ángulos de visión. En 1997 Murray *et al.*, [33] demostraron que la visión estereoscópica era una alternativa fiable frente a otro tipo de sensores como el láser o el sonar .
  - **Cámara omnidireccional**: Captan imágenes con un ángulo de visión de 360º lo que las hacen idóneas para tareas de navegación. Permiten obtener una descripción completa del entorno con una sola imagen a diferencia de las cámaras estereoscópicas que requieren de más de una imagen con diferente ángulo para su correcto funcionamiento. Otra ventaja de este tipo de cámaras es que las imágenes resultantes contienen la misma información independientemente de la orientación relativa del robot en el plano del suelo, con lo que permiten construir mapas con los que el robot podrá localizarse independientemente de su orientación, e incluso estimar esta orientación relativa.

El empleo de sensores de visión para tareas de navegación es una decisión muy acertada debido a la gran cantidad de información que podemos extraer de las imágenes. La única desventaja podría residir en que estas imágenes precisan de ser procesadas por un computador, desarrollando los algoritmos adecuados podemos extraer la información realmente importante para la tarea deseada. Todo este procedimiento conlleva un elevado coste computacional

que hace unos años era inasumible. En la acualidad, este problema se ha solventado por el gran desarrollo de la electrónica y la computación.

En particular, en este trabajo final de grado nos vamos a centrar en las cámaras omnidireccionales. Podemos destacar dos configuraciones, un sistema multicámara como el que podemos encontrar en la (Figura **1-7**) y el modelo compuesto por una cámara y un espejo hiperbólico (Figura **1-8**). Esta última configuración es la elegida para la realización del presente trabajo debido a su simplicidad.

Figura 1-7: Insta360 Titan.

Figura 1-8: Sistema catadióptrico omnidireccional.

La utilización de cámaras omnidireccionales puede ser interesante respecto a otro tipo de sensores, como los lidar, por la gran cantidad de información que ofrecen y su coste relativamente bajo. Y a diferencia de otro tipo de sensores, son capaces de distinguir diferentes texturas y contornos, lo cual puede ser interesante según la aplicación.

Adicionalmente, las imágenes omnidireccionales contienen información completa del entorno del robot, visto desde el punto de captura, de modo que es posible crear modelos del entorno con un número reducido de escenas, con la consiguiente reducción del coste computacional. Siendo esta característica de especial interés para su aplicación en problemas de navegación.

Durante los últimos años, han emergido varios trabajos que proponen el uso de la Inteligencia Artificial (IA) para resolver tareas de robótica móvil mediante el uso de imágenes. La integración de robótica e Inteligencia Artificial es cada vez más común en numerosos sectores económicos. Un ejemplo de ello podría ser el Autopilot de Tesla, que implementa una herramienta de visión artificial para identificar las diferentes señales de tráfico, peatones, otros vehículos y elementos esenciales en la calzada para conseguir una circulación segura (Figura **1-9**). En el sector primario, el reconocimiento visual de animales se está empleando

para analizar imágenes predictivas sobre el estado de salud y bienestar del ganado (Figura **1-10**), hasta incluso puede ayudar a los agricultores a implementar métodos de crecimiento más eficientes.

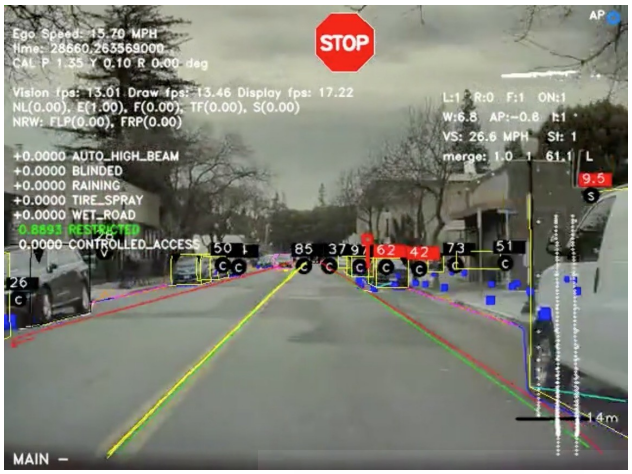


Figura 1-9: Autopilot Tesla.

Figura 1-10: Reconocimiento facial de animales.

El presente trabajo se centra en el uso de las redes neuronales convolucionales (Convolutional Neural Networks), las cuales, son una rama de la Inteligencia Artificial. En concreto, las redes neuronales son una herramienta perteneciente al campo del aprendizaje profundo (Deep Learning).

Las CNNs imitan la estructura del cerebro, el cual está compuesto por una gran cantidad de neuronas conectadas entre sí. Cuando el cerebro guarda cierta información lo hace alterando la forma en la que se asocian las neuronas. En las CNNs se llevaría a cabo un proceso análogo variando los pesos asociados a cada enlace entre neuronas y los umbrales de activación de cada neurona. El esquema básico de una neurona artificial y sus componentes principales se pueden apreciar en la Figura **1-11**, donde $x_1, x_2, ..., x_n$ son las señales de entrada las cuales van asociadas a sus pesos correspondientes $w_1, w_2, ..., w_n$. Por otro lado, w_0 es el bias o sesgo, que es otro factor asociado al almacenamiento de información y toma un valor constante. En otras palabras, la información de la red neuronal se almacena en forma de pesos y sesgo.

La señales de entrada procedentes del exterior $(x_1, x_2, ..., x_n)$ se multiplican por los pesos $(w_1, w_2, ..., w_n)$ antes de llegar al nodo. Una vez que las señales ponderadas se recogen en el nodo, estos valores se suman dando lugar a la suma ponderada. De manera que la señal con mayor peso tiene un mayor efecto. Por ejemplo, si el peso w_1 es 1, y w_2 es 5, entonces la señal x_2 tiene un efecto cinco veces mayor que el de x_1. Cuando w_1 es cero, x_1 no se transmite al nodo. Esto significa que x_1 está desconectado del nodo. Este ejemplo muestra que los pesos de la red neuronal imitan cómo el cerebro altera la asociación de las neuronas. Finalmente, el nodo introduce la suma ponderada en la función de activación y obtiene su salida y.

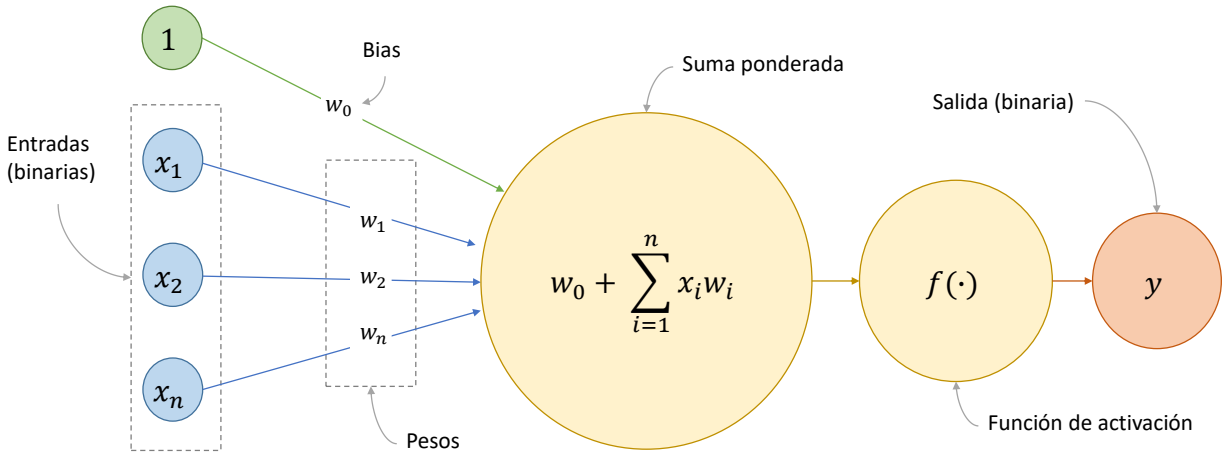


Figura 1-11: Esquema básico de una neurona artificial.

Entre los diferentes elementos que componen una neurona artificial, el único componente que es seleccionable por el diseñador es la función de activación la cual determina el comportamiento del nodo. A continuación destacamos las funciones de activación más frecuentes y estandarizadas, que están disponibles en cualquier librería:

- **Función sigmoide:** Muchos procesos naturales y curvas de aprendizaje de sistemas complejos muestran una progresión temporal desde unos niveles bajos al inicio, hasta acercarse a un clímax transcurrido un cierto tiempo. La transición se produce en una región caracterizada por una fuerte aceleración intermedia. La función sigmoide permite describir esta evolución. Su gráfica tiene una típica forma de "S"(Figura **1-12**). Su rango está entre 0 y 1.

- **Tanh (Tangente hiperbólica):** Corresponde al cociente entre el seno hiperbólico y el coseno hiperbólico. Como la función sigmoide, tiene forma de "S" y se considera una versión mejorada. Su rango se encuentra entre -1 y 1.

- **ReLu (Rectified Linear Unit):** Devuelve 0 si el input es negativo y devuelve el input si este es positivo.

- **SoftMax:** También conocida como función exponencial normalizada. Generalmente, se emplea en la última capa de la red neuronal ya que calcula la distribución de probabilidades de una clase sobre el total de categorías. Devuelve valores en el rango 0 a 1, por lo que se usa para clasificación.

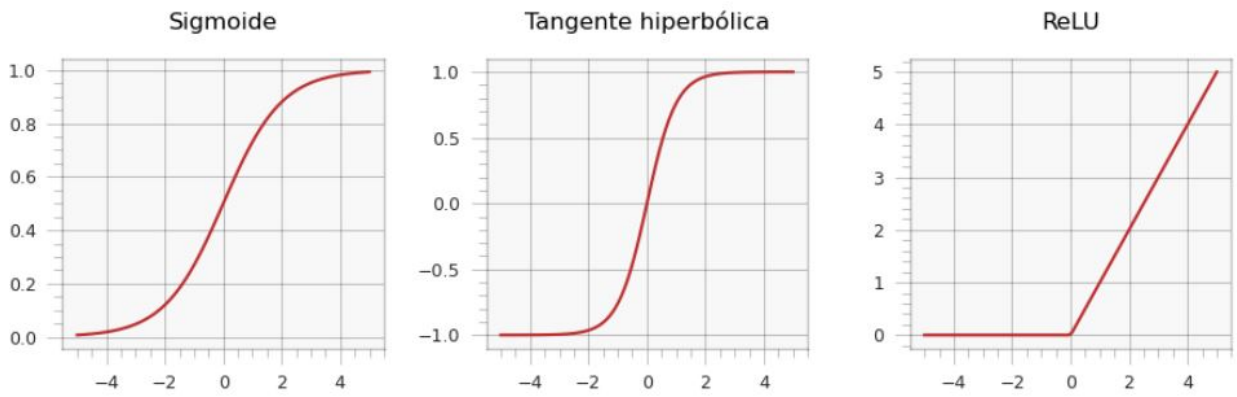


Figura 1-12: Repersentación de las funciones de activación más comunes.

Es posible encontrar una gran variedad de redes neuronales en función de cómo están distribuidos y conectados los diferentes nodos (o neuronas). Una red neuronal se podría subdividir así misma en diferentes capas, entre las cuales podemos destacar la capa de entrada, las capas ocultas y capa de salida. Los nodos de la capa de entrada carecen de pesos asociados a ella, simplemente transmiten la información de los datos de entrada a los siguientes nodos. A partir de la capa de salida se obtiene el resultado final de la red neuronal, estos nodos sí que adquieren pesos así como los de la capa oculta. Las capas de entrada y de salida son accesibles desde el exterior. Por el contrario, las capas ocultas que se encuentran entre la entrada y la salida, no son accesibles y de ahí reciben su nombre.

A lo largo de los últimos años, las redes neuronales han evolucionado desde estructuras más simples a más complejas, pudiendo clasificarlas en dos categorías según si poseen capa oculta o no: las redes de una sola capa (Figura **1-13**) y las de capas múltiples. Las redes neuronales de capas múltiples podemos dividirla a su vez en dos clases, las redes neuronales poco profundas o *shallow neural networks* (Figura **1-14**) y las profundas o *deep neural networks* (Figura **1-15**) en función de la cantidad de nodos que presenten en su capa oculta.

Las redes neuronales convolucionales (CNNs) tienen una gran variedad de aplicaciones. Como ejemplo, podrían ser entrenadas para predecir las ventas de una empresa en función de unas determinadas variables (inputs), en traumatología se pueden entrenar para diagnosticar si un hueso tiene algún tipo de rotura mediante las radiografías o también se podrían utilizar con el fin de identificar mediante el uso de cámaras aquellos productos que a simple vista no cumplen los requisitos necesarios para ser empaquetados y vendidos y en condiciones de trabajo reales.

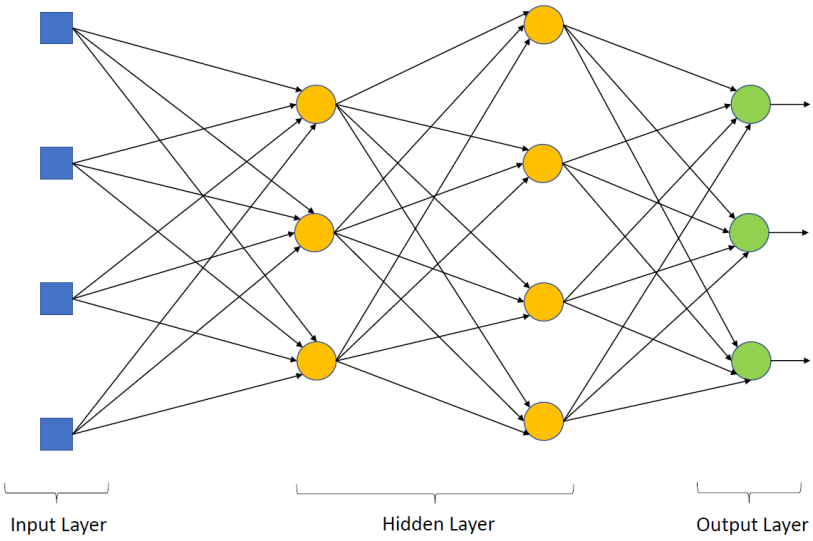


Figura 1-15: Deep Neural Network.

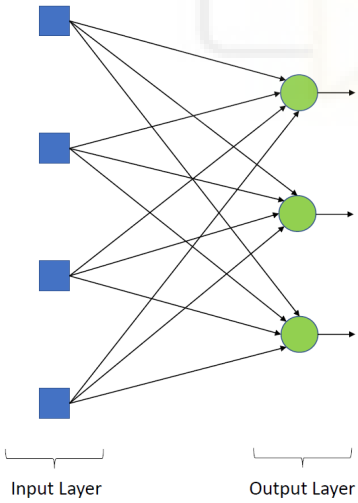


Figura 1-13: Single-layer Neural Network.

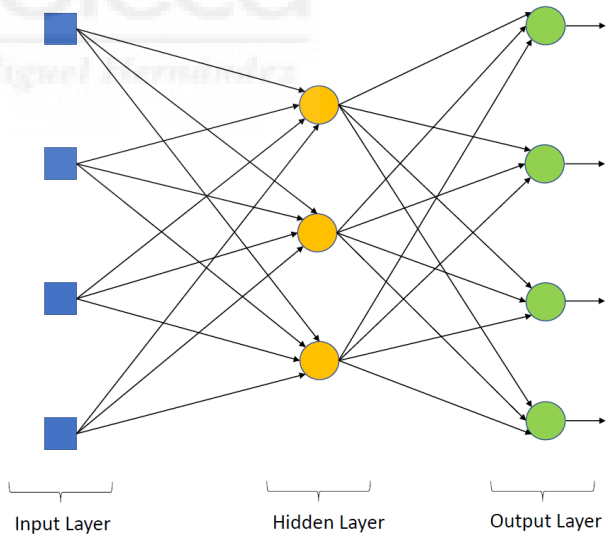


Figura 1-14: (shallow) Multi-layer Neural Network.

El objetivo del presente trabajo de investigación es el entrenamiento, optimización y validación de diferentes CNNs con el objetivo de resolver la tarea de localización, consiguiendo así no solo la identificación de la estancia en la que se encuentra el robot, sino también estimar las coordenadas en que se encuentra en el plano del suelo. Para realizar el entrenamiento y testeo de las redes neuronales se han empleado imágenes omnidireccionales capturadas en entornos de interior con diferentes condiciones de iluminación.

La localización de un robot autónomo móvil se puede abordar desde dos perspectivas, ambas han sido abordadas en el presente trabajo:

- **Localización global**: Consiste en localizar las coordenadas en las cuales se encuentra el robot sin dividir la tarea en diferentes fases. De esta manera se emplea toda la información de forma bruta, incrementando así el tiempo para llevar a cabo la localización.
- **Localización jerárquica**: Habrá un mapa que se estructura en varias capas, una o varias de alto nivel, que contienen poca información y que permiten una localización gruesa, en una zona del entorno, de manerá rápida, y una o varias capas de bajo nivel, que contienen más información, y permiten una localización más fina. De este modo, la localización se puede resolver en varias fases, estimando primero la posición de forma gruesa (en este trabajo se estimará primero la estancia en que está el robot) y, en la segunda fase, se estimará de manera más fina la posición, usando las capas de bajo nivel, pero restringiéndonos sólo a buscar en la zona o estancia recuperada en el anterior paso, lo cual es un proceso más eficiente.

El resto de la memoria se estructura de la siguiente manera, en el capítulo 2 se hará referencia a investigaciones previas de localización relacionadas con el uso del aprendizaje profundo y vectores descriptores. En el tercer capítulo comentaremos las diferentes herramientas empleadas como lo son el Transfer Learning, optimización de hiperparámetros mediante la Optimización Bayesiana, Data Augmentation y vectores descriptores de apariencia global. En el capítulo posterior se explicará de forma más detallada como se han implementado las diferentes técnicas y en el capítulo 5 se expondrán los resultados de los experimentos. Para acabar, en el capítulo 6 se comentarán las conclusiones extraídas del presente trabajo y se propondrán posibles líneas de investigación futuras.

2 Estado del arte

Los trabajos y predicciones de Von Neumann, Alan Turing y Wiener dieron lugar no solo a la computadora moderna sino también a la "Inteligencia Artificial", término establecido por John McCarthy en 1955 y que podríamos definir como el desarrollo de algoritmos con el fin de automatizar tareas que normalmente sólo el ser humano es capaz de realizar. Desde entonces se han llevado a cabo numerosas investigaciones en torno a esta temática.

La Inteligencia Artificial está presente en diferentes áreas del conocimiento. Por ejemplo, Wang *et al.*, [23] emplearon una Red Neuronal para el diagnóstico de los diferentes estados de enfermedades cardiovasculares. Alessandri *et al.*, [2] emplearon la Inteligencia Artificial para el control de un robot que realizaba tareas quirúrgicas. Otros autores han empleado el aprendizaje profundo para la predicción de terremotos (Bertrand *et al.*, [44]). Hasta incluso existen estudios que emplean algoritmos basados en Inteligencia Artificial para automatizar la agricultura mediante robots y drones (Talaviya *et al.*, [53]).

La Inteligencia Artificial ha supuesto una alternativa para el desarrollo de herramientas con el fin de abordar tareas propias de robots móviles, obteniendo resultados más eficientes. Recientemente, el uso de las CNNs ha atraído a más autores a la implementación de la Inteligencia Artificial junto con el uso de sensores de visión. Las tres tareas principales que aborda la robótica móvil son la navegación, la localización y el mapeo, todas ellas se pueden abordar implementando algoritmos basados en Inteliegencia Artificial. Por ejemplo, Tai *et al.*, [52] entrenaron una CNN para abordar el problema de la exploración evitando el impacto con obstáculos. La detección de obstáculos es un tema importante en la navegación, por lo que hay diferentes investigaciones que se centran exclusivamente en el desarrollo de algoritmos que lleven a cabo esta función (Abadi *et al.*, [1] y Menegatti *et al.*, [31]). Por otro lado, Amer *et al.*, [3] llevan a cabo una localización de UAVs en zonas urbanas. Sandino *et al.*, [46] también hicieron uso de UAVs y de la Inteligencia Artificial para llevar a cabo el mapeo aéreo de bosques afectados por diferentes patógenos.

Cabe destacar que las tareas propias de los robots móviles no solo han sido abordadas mediante el uso de la IA, sino que existen multitud de estudios que emplean otras técnicas. Además, el mapeo puede ir ligado con la localización de manera que se resuelven estos dos problemas de manera conjunta, esta técnica se conoce como SLAM (*Simutaneous Mapping And Localization*). Muchos autores han abordado este problema mediante el empleo de apro-

ximaciones probabilísticas (Thrun *et al.*, [56]). La creación de mapas ha sido adaptada a diferentes tipos de entornos, tanto de interior como de exterior, incluyendo información 2D o incluso 3D, y teniendo en cuenta situaciones reales, como posibles cambios en el entorno. Cuando el entorno es cambiante, resulta especialmente relevante la identificación de aquellos objetos que aparecen esporádicamente ya que no deberían de formar parte del mapa generado, consiguiendo así sistemas más precisos (Hahnel *et al.*, [22]). Posteriormente, Wolf *et al.*, [60] desarrollaron un algoritmo online capaz de generar un mapa de tres capas, la primera capa recoge el entorno estático, la segunda el entorno dinámico y la tercera los puntos de referencia estáticos que permiten la localización del robot. La combinación de estas tres capas proporcionan una descripción muy precisa del entorno del robot.

Algunos autores han abordado las tareas de mapeo y localización empleando la tecnología GPS (Tao *et al.*, [54]), otros han optado por la tecnología láser (Zhang *et al.*, [63]). Dentro de la tecnología laser, encontramos los sistemas LiDAR que son altamente empleados en robótica móvil, Sprunk *et al.*, [49] hicieron uso de estos sistemas para automatizar tareas de logística. Weiss *et al.*, [57] analizaron las ventajas y desventajas de los sensores LiDAR frente a herramientas tradicionales como la visión estereoscópica en el ámbito de la robótica agrícola. La tecnología sonar también es frecuentemente empleada para la creación de mapas y localización. En concreto, Feder *et al.*, [17] desarrollaron un mapeo estocástico mediante el uso del filtro de Kalman, el cual permite reducir el ruido blanco.

Los sensores de visión han sido ampliamente utilizados para tareas de mapeo, localización y navegación. Es posible combinar cámaras con otros tipos de sensores, por ejemplo Ohya *et al.*, [35] emplearon una cámara estándar junto a un sensor de ultrasonidos para llevar a cabo la navegación. Adicionalmente, Gallegos *et al.*, [20] combinaron la información obtenida por una cámara omnidireccional y por un láser 2D para realizar el mapeo y localización de forma simultánea. Otros han empleado los sistemas LiDAR junto con cámaras RGB-D para detectar y evitar los objetos que se encuentre el robot en su camino (Gatesichapakorn *et al.*, [21]).

Otros autores han propuesto recurrir únicamente al empleo de sensores de visión de los cuales las cámaras omnidireccionales son frecuentemente empleadas para resolver tareas propias de robots móviles debido a las grandes ventajas que presentan. El uso de este tipo de cámaras es una opción robusta y fiable para la realización del mapeo y localización tal y como se ha demostrado (Payá *et al.*, [39]). Este tipo de cámaras también se han incorporado en vehículos dónde además de desempeñar las tareas mencionadas anteriormente, se predice la trayectoria del automóvil con una alta precisión (Tardif *et al.*, [55]). Kuutti *et al.*, [25] evaluaron las técnicas de localización de vehículos más avanzadas e investigaron su aplicabilidad en vehículos autónomos, todas ellas basadas en sistemas de visión.

Cuando se trabaja con imágenes para resolver el problema de mapping y localización, es necesario extraer información relevante de dichas escenas, que sea útil para que el robot pueda estimar con robustez su pose (posición y orientación). Esta descripción de escenas se puede realizar mediante la extracción de características locales o descriptores de apariencia global. Por un lado, las características locales están basadas en la extracción de puntos, objetos o regiones características de la escena. Cada característica se describe en un descriptor con lo cual una escena es representada por un conjunto de descriptores. Por ejemplo, Andreasson *et al.*, [4] llevaron a cabo la localización de un robot móvil mediante la modificación del algoritmo SIFT (*Scale-Invariant Feature Transform*) de Lowe *et al.*, [28] para emparejar características extraídas de puntos de interés locales de imágenes panorámicas. Por otro lado, los vectores de apariencia global consisten en representar cada escena utilizando un único descriptor que recoge la información global de la imagen. Payá *et al.*, [36] demostraron de forma robusta que estos vectores pueden ser empleados para la localización y la creación de mapas.

En los últimos años, se han propuesto varios trabajos que ligan el uso de los sensores visuales a las redes neuronales convolucionales para resolver la tarea de localización en robots móviles. Por ejemplo, Xu *et al.*, [62] proponen un sistema de localización global en interiores basado en el entrenamiento de una CNN. En cambio, Chen *et al.*, [11] emplea una CNN para la extracción de vectores descriptores. El objetivo del presente trabajo es entrenar, optimizar y testar una red neuronal que, dada una imagen de entrada, cumpla un doble objetivo: (1) que sea capaz de identificar la estancia en la que se capturó esta imagen y (2) que estime con la mayor precisión posible las coordenadas del punto de dicha habitación en que se capturó la imagen.

Las redes neuronales juegan un papel fundamental dentro de la Inteligencia Artificial, para explicar su procedencia debemos remontarnos a 1943 cuando McCulloch y Pitts escriben un artículo en el que explican la computación de la actividad neuronal en el cerebro. En 1949, Hebb reforzó las ideas de McCulloh-Pits en su trabajo "*The organization of behavior: A neuropsychological theory*". El primer algoritmo de aprendizaje no se desarrolarría hasta 1958, cuando Rosenblatt [43] crea el denominado "Perceptrón". Más tarde, en 1969 Minsky y Papert abordaron aspectos sobre las redes neuronales simples en su libro "*Perceptrons: An introduction to computational geometry*". En 1975, Paul Werbos intoduciría el concepto de entrenamiento de redes neuronales mediante la retropropagación y hacia la decada de los noventa consiguiría un desarrollo avanzado de este método (Werbos, [58]).

Este tipo de redes se han diseñado con diferentes propósitos a lo largo de los años. En 1998, Lecun *et al.*, [26] propusieron una pionera red de 8 capas denominada LeNet-5 y caracterizada por el reconocimiento y clasificación de dígitos manuscritos. En la actualidad, hay numerosas CNNs entrenadas con un alto número de escenas para resolver diversos problemas, entre las

cuales podemos destacar AlexNet la cual fue diseñada por Krizhevsky *et al.*, [51] en 2012 para llevar a cabo una clasificación entre 1000 objetos. En 2015, GoogLeNet fue propuesta por Szegedy *et al.*, [50]. Al igual que AlexNet, esta red fue diseñada para llevar a cabo la clasificación de 1000 objetos. En este caso, aumentando a 22 el número de capas y reduciendo el número de parámetros de 60 millones (AlexNet) a 4 millones.

Todas estas redes neuronales se pueden utilizar como punto de partida para desarrollar diferentes herramientas con el fin de conseguir otros objetivos. Para ello, se puede hacer uso de una técnica conocida como *Transfer Learning* ya que permite aprovechar todo el conocimiento previo de la red, consiguiendo así resultados más robustos ahorrando tiempo de computación. Esta idea ya ha sido empleada por otros autores como Wozniak *et al.*, [61] que partió de la red neuronal VGG-F y la reentrenó para llevar a cabo la clasificación de 16 habitaciones mediante 8000 imágenes. El *Transfer Learning* resuelve dos grandes problemas dentro del *Machine Learning*. El primero de ellos es la falta de datos de entrenamiento, transfiriendo los conocimientos aprendidos originalmente para la realización de la nueva tarea. El segundo problema que resuelve esta técnica es la reducción de los tiempos de entrenamiento de las CNNs. Además, esta herramienta relaja la hipótesis de que los datos de entrenamiento deben ser independientes e idénticamente distribuidos con los datos de test.

Cuando una CNN es lo suficientemente buena como para realizar una tarea específica, las diferentes capas de la red neuronal generan unos vectores descriptores de apariencia global empleados para dicha tarea, pero estos vectores pueden ser extraidos con el objetivo de darles una finalidad totalmente distinta, como puede ser la tarea de localización. Esta idea ya ha sido propuesta por diferentes autores como Mancini *et al.*, [29] los cuales realizaron una clasificación mediante el uso del clasificador Naïve Bayes. Más tarde Payá *et al.*, [38] emplearon los vectores descriptores de apariencia global para llevar a cabo una localización jerárquica. Otros autores como Cebollada *et al.*, [9] han empleado también la localización jerárquica teniendo en cuenta variaciones en la iluminación ambiente.

Con frecuencia suele ocurrir que no se dispone de un dataset de entrenamiento lo suficientemente grande como para entrenar la CNN. Para paliar este problema se hace uso del *Data Augmentation*, el cual consiste en generar artificialmente diversas imágenes de entrenamiento a partir de cada imagen inicial, incluyendo algunas modificaciones sobre ellas de manera que creamos nuevas imágenes de entrenamiento con leves distorsiones. Al conseguir un mayor número de imágenes de entrenamiento también se consigue a su vez evitar un sobreentrenamiento tal y como se ha demostrado por diferentes autores. Shorten *et al.*, [47]

hace uso de transformaciones geométricas, cambios de color, filtros kernel, mezcla de imágenes y borrado aleatorio entre otras técnicas para conseguir un data augmentation. Perez *et al.*, [40] proponen un método denominado Neural Augmentation para conseguir que una CNN aprenda qué distorsiones son las que mejoran la precisión del clasificador. Por ejemplo, Ding *et al.*, [14] utilizan tres métodos de aumento de datos para abordar un reconocimiento de objetivos por radar. El objetivo de este trabajo es hacer que la red sea robusta frente a la traslación entre otras perturbaciones. Salamon *et al.*, [45] proponen el aumento de datos de audio para superar el problema de la escasez de datos de sonido ambiental.

Con el fin de aumentar la precisión del modelo de puede recurrir a la optimización de los hiperparámetros de entrenamiento. En el ámbito del *machine learning*, los hiperparámetros son aquellos valores cuya configuración es externa y no se aprenden a partir de los datos. Por ejemplo, mientras que los pesos de las CNNs son parámetros (se aprenden durante el proceso de entrenamiento), el *Initial Learn Rate* es un hiperparámetro que controla cómo de rápido el modelo se adapta al problema (lo establece el arquitecto de la CNN y no se aprende durante el proceso de entrenamiento). Este tipo de parámetros suelen ser especificados por el programador y ajustados en función de la tarea a desempeñar. El diseñador de la red no puede conocer de antemano los mejores valores de los hiperparámetros ante un problema determinado. Por ello, puede utilizar reglas empíricas como copiar valores utilizados en otros problemas o buscar el mejor valor por ensayo y error. El correcto entrenamiento de un modelo depende en gran medida de la configuración de los hiperparámetros y, por tanto, del método utilizado para establecerlos. Los métodos de optimización como la búsqueda en cuadrícula y la búsqueda aleatoria han demostrado superar a los métodos clásicos para este problema (Bergstra *et al.*, [6]). Estos métodos han sido capaces de obtener ajustes de hiperparámetros similares o mejores que los establecidos por los expertos del machine learning (Bergstra *et al.*, [7] y Kotthoff *et al.*, [24]). Como resultado, la optimización de hiperparámetros se ha convertido en una importante área de investigación (Falkner *et al.*, [16] y Feurer *et al.*, [18]). Durante los últimos años, la optimización Bayesiana ha surgido como una eficiente alternativa, logrando mejores resultados (Snoek *et al.*, [48] y Domhan *et al.*, [15]).

3 Herramientas usadas

Durante el desarrollo de esta investigación se han empleado diferentes herramientas y métodos como son la Visión omnidireccional, redes neuronales convolucionales, *Transfer Learning*, *Data Augmentation* y optimización de hiperparametros mediante la optimización Bayesiana.

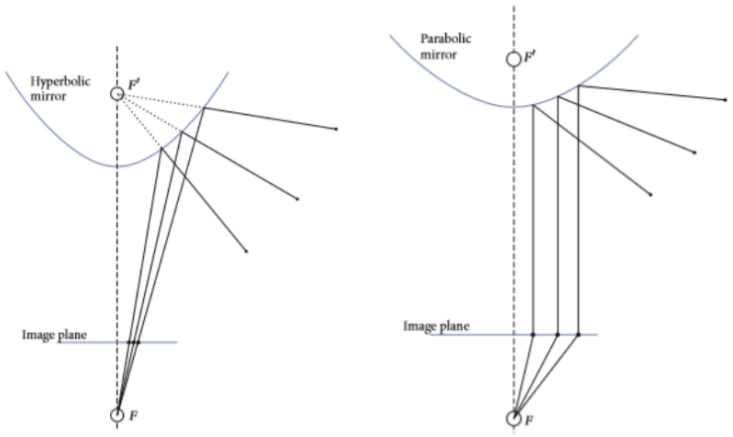
3.1. Visión omnidireccional

Los sensores de visión son ampliamente utilizados para la resolución de tareas de robótica móvil debido a la gran cantidad de información que se puede extraer de ellos.

En concreto, el uso de las cámaras omnidireccionales para la tarea de localización ha ido incrementando debido a su amplio campo de visión, obteniendo información con un campo de visión de 360º alrededor del robot con una sola imagen. Esta característica repercute directamente en la cantidad de imágenes que tiene que procesar el robot y por tanto en el coste computacional de la tarea de localización. Entre las diferentes ventajas que conlleva el uso de este tipo de cámaras, se podría destacar la detección de la posición del robot sin importar la orientación del mismo, la mayor estabilidad de las imágenes o el menor coste de este tipo de cámaras comparado con otro tipo de sensores.

Existen diferentes configuraciones de sensores de visión omnidireccionales entre los cuales podemos destacar el sistema catadióptrico por su gran simplicidad. La reflexión de la luz sobre el espejo cóncavo permite la creación de este tipo de imágenes. La curvatura del espejo puede ser hiperbólica (Figura **3-1** (a)) o parabólica (Figura **3-1** (b)) en función de si la lente de la cámara es convencional o ortográfica.

Hay una gran cantidad de trabajos de investigación en los que se han utilizado cámaras omnidireccionales. Por ejemplo, Cebollada *et al.*, [8] entrenaron una red neuronal no solo para llevar a cabo la clasificación en estancias sino que también la emplearon para obtener los vectores descriptores de apariencia global. Abordaron la tarea de localización mediante dos enfoques, globalmente y jerárquicamente. Para la localización global emplearon la técnica del vecino más cercano comparando los vectores descriptores de las imágenes de entrenamiento y de testeo. Para la localización jerárquica, las imágenes test son clasificadas y seguidamente se comparan sus vectores descriptores con los de las imágenes training aplicando también la técnica del vecino más cercano.



(a) (b)

Figura 3-1: Tipos de espejo para el sistema catadióptrico: (a) Espejo hiperbólico, (b) Espejo parabólico.

Es común realizar una transformación de las imáganes omnidireccionales a panorámicas para trabajar con ellas. Sin embargo, el presente trabajo trata de utilizar directamente imágenes omnidireccionales a color para comprobar su rendimiento respecto a otros experimentos realizados. Esta metodología no ha sido ampliamente estudiada hasta la fecha. Tan solo encontramos pequeños estudios en trabajos previos (Cespedes, [10]) donde se demostró las ventajas que podía presentar el uso de imágenes omnidireccionales. A priori, la ventaja más directa de utilizar imágenes omnidireccionales es el ahorro de tiempo que conllevaría tener que realizar la transformación de omnidireccional a panorámica.

3.1.1. Base de datos COLD

En el presente trabajo se ha tomado como punto de partida las imágenes de la base de datos COLD (COsy Localization Database) la cual está constituida por imágenes omnidireccionales que se caracterizan por tener 3 capas (RGB) de 470x470 píxeles. Existen varios datasets en función de las diferentes condiciones de iluminación (Figura **3-2**) y también con diferentes trayectorias.

Las imágenes son capturadas mediante un sistema catadióptico omnidireccional incorporado en un robot (Figura **3-3**) que realiza una ruta por las diferentes estancias del edificio cuya planta se puede apreciar en la (Figura **3-4**). En este trabajo se ha seleccionado la ruta roja de las mostradas en la (Figura **3-4**) ya que el robot visita un mayor número de estancias. Además, se escoge el dataset de nublado (seq2_cloudy3) para realizar el entrenamiento ya que es la condición de iluminación más estándar, y que permite ver un mayor número de detalles en las imágenes. Asimismo, presenta un menor contraste de iluminación entre el interior y el exterior del edificio. Este dataset de entrenamiento se ha muestreado para

conseguir una separación entre imágenes de 40 centímetros apróximadamente para ser comparable con otros estudios que emplean mapeado de rejilla realizados por el Laboratorio de Automatización, Robótica y Visión por Computador de la Universidad Miguel Hernández de Elche.

(a) (b) (c)

Figura 3-2: Ejemplo de imágenes pertenecientes a la base de datos COLD para las tres condiciones de iluminación: (a) nublado, (b) soleado y (c) noche. Las imágenes contenidas en este dataset pueden ser descargadas a través de su página web https://www.cas.kth.se/COLD/.

Además, se contará con datasets de nublado (seq2_cloudy2), soleado (seq2_sunny4) y noche (seq2_night1) para comprobar el buen funcionamiento de la red neuronal.

El edificio de la base de datos COLD consta de nueve estancias de las cuales el pasillo (CR-A) hace de vínculo de unión y por tanto es la estancia con más imágenes (Tabla **3-1**). Cabe resaltar que las redes neuronales son capaces de aprender la probabilidad de que aparezca una imagen de una estancia u otra en función de la cantidad de imágenes de entrenamiento que tengamos de cada estancia. En concreto, en esta tarea de clasificación esta característica de las redes neuronales es de especial interés tal y como se verá más adelante.

Estancia	1P0-A	2P01-A	2P02-A	CR-A	KT-A	LO-A	PA-A	ST-A	TL-A
Nº imágenes	44	46	31	238	46	26	57	30	38

Tabla 3-1: Cantidad de imágenes de entrenamiento.

Figura 3-3: Ejemplo de robot movil autónomo.

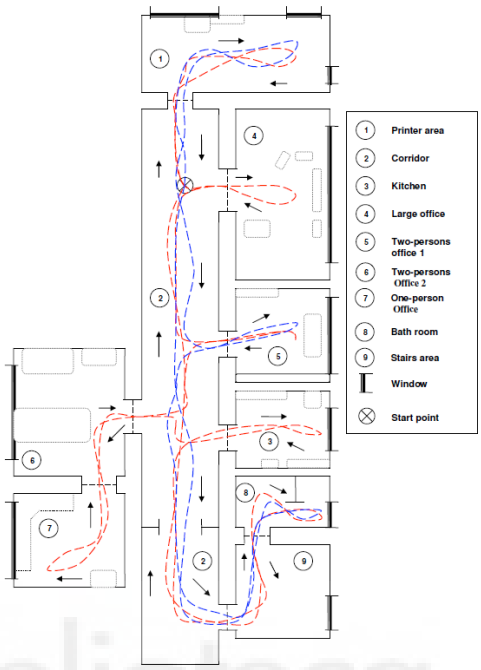


Figura 3-4: Planta edificio Friburgo de la base de datos COLD.

3.2. Convolutional Neural Network(CNN)

El presente trabajo propone resolver las tareas de mapeo y localización en robótica móvil mediante redes neuronales convolucionales que pertenecen a la rama del Deep Learning dentro de la Inteligencia Artificial (AI). Las redes neuronales convolucionales estás compuestas por una serie de capas que transforman las imagen de entrada (input) en una predicción (output). Existen diferentes tipos de capas que combinadas entre sí conforman una CNN. Las capas más frecuentes, que están presentes en la mayoría de redes neuronales convolucionales:

- **Convolutional Layer (capa convolucional):** Esta capa es empleada para extraer las distintas características de las imágenes de entrada. En ésta se realiza la operación matemática de convolución entre la imagen de entrada y un filtro de un tamaño determinado MxM (kernel size). Al pasar el filtro sobre la imagen de entrada, se obtiene una imagen desalida denominada mapa de características, que nos proporciona información sobre la imagen, como las esquinas y los bordes. Más tarde, este mapa de características se alimenta a otras capas para aprender otras características de la imagen de entrada.

- **Fully Connected Layer (FC):** También conocida como capa totalmente conectada, se utiliza para conectar las neuronas entre dos capas diferentes. Estas capas suelen

situarse antes de la capa de salida y forman las últimas capas de una arquitectura CNN. En ella, la imagen de entrada de las capas anteriores se aplana y se introduce en la capa FC. A continuación, el vector aplanado se somete a unas cuantas capas FC en las que suelen tener lugar operaciones matemáticas. En esta etapa comienza el proceso de clasificación.

- **Dropout:** El dropout consiste en ignorar una fracción aleatoria de las neuronas que tiene esta capa como entrada. El objetivo principal de esta capa es ahorrar tiempo de entrenamiento por medio de un modelo más sencillo. Al despreciar ciertas neuronas, se reduce el número de parámetros y por tanto los cálculos de los pesos asociados a cada una de estas neuronas. Además, este tipo de capas también permiten reducir el sobreajuste de los modelos.

- **Batch normalization:** También conocido como Normalización por lotes. Tiene como función normalizar las entradas de una capa para cada mini lote (mini-batch). Esto tiene el efecto de estabilizar el proceso de aprendizaje y reducir drásticamente el número de épocas de entrenamiento necesarias para entrenar redes neuronales profundas.

- **Pooling Layer:** El objetivo principal de esta capa es disminuir el tamaño del mapa de características, generado en la convolucional, para reducir los costes computacionales. Para ello, se reducen las conexiones entre las capas y se opera de forma independiente en cada mapa de características el cual es dividido en bloques de un tamaño determinado NxN. Dependiendo del método utilizado, existen varios tipos de operaciones de Pooling (Figura **3-5**):

  - Max Pooling: Para cada uno de los bloques, se selecciona el mayor de los valores como output y se ignora el resto.

  - Average Pooling: Para cada uno de los bloques, se toma el valor medio de los elementos del bloque como output y se ignora el resto.

  - Sum Pooling: Para cada uno de los bloques, se toma la suma total de los elementos del bloque como output y se ignora el resto.

  La capa de pooling suele servir de puente entre la capa convolucional y la capa de totalmente conectada.

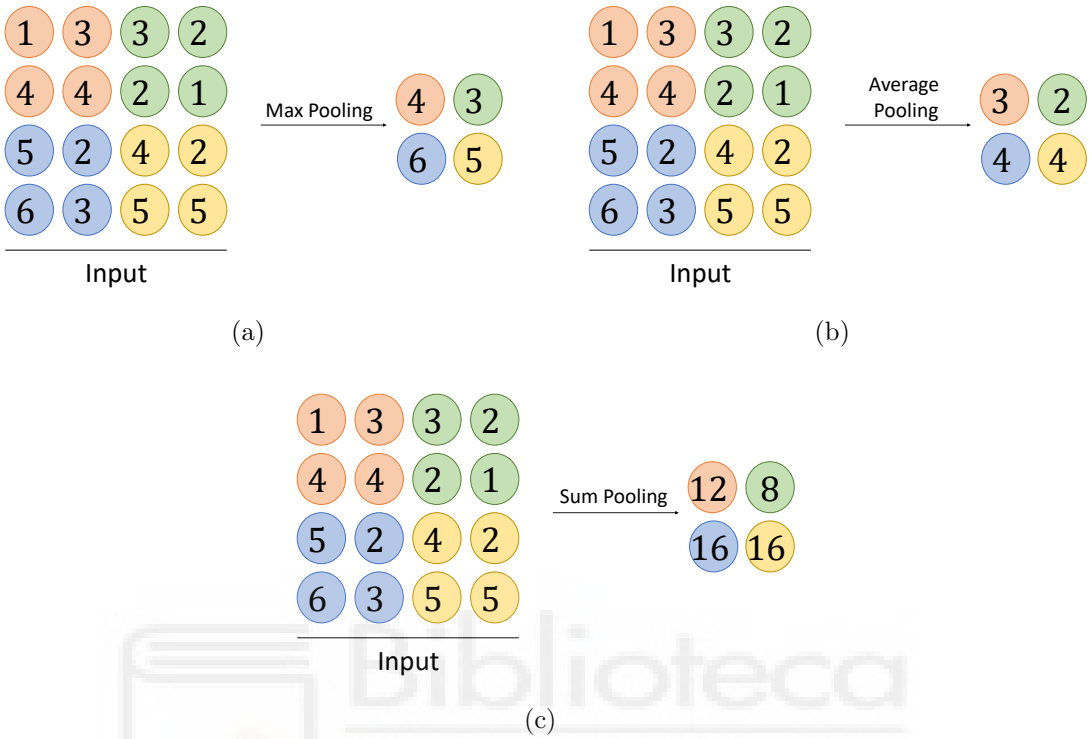


Figura 3-5: Ejemplo de operaciones de Pooling: (a) Max Pooling, (b) Average Pooling y (c) Sum Pooling.

En función de cómo estén interconectadas las capas entre sí y las funciones de activación que se utilicen, se pueden generar infinidad de arquitecturas para aplicaciones totalmente distintas. En la actualidad, hay ya diferentes CNNs ampliamente entrenadas para la tarea de clasificación de objetos con una precisión muy alta, como pueden ser AlexNet o GoogLeNet.

3.2.1. Arquitectura de Alexnet

En esta investigación se ha tomando como punto de partida AlexNet para llevar a cabo la tarea de clasificación y regresión. AlexNet se caracteriza por tener ocho capas (cinco capas convolucionales y tres capas completamente conectadas) con un softmax final que permite clasificar imágenes en 1000 categorías de objetos. Una representación esquemática de su arquitectura sería la que encontramos en la (Figura **3-6**).

El diseño y entrenamiento de una red neuronal es una tarea relativamente sencilla, pero podemos obtener mejores resultados y ahorrar tiempo si creamos una red neuronal a partir de otra, conservando los conocimientos de la red original y modificando la finalidad de la red. Esta técnica es conocida como *Transfer Learning*.

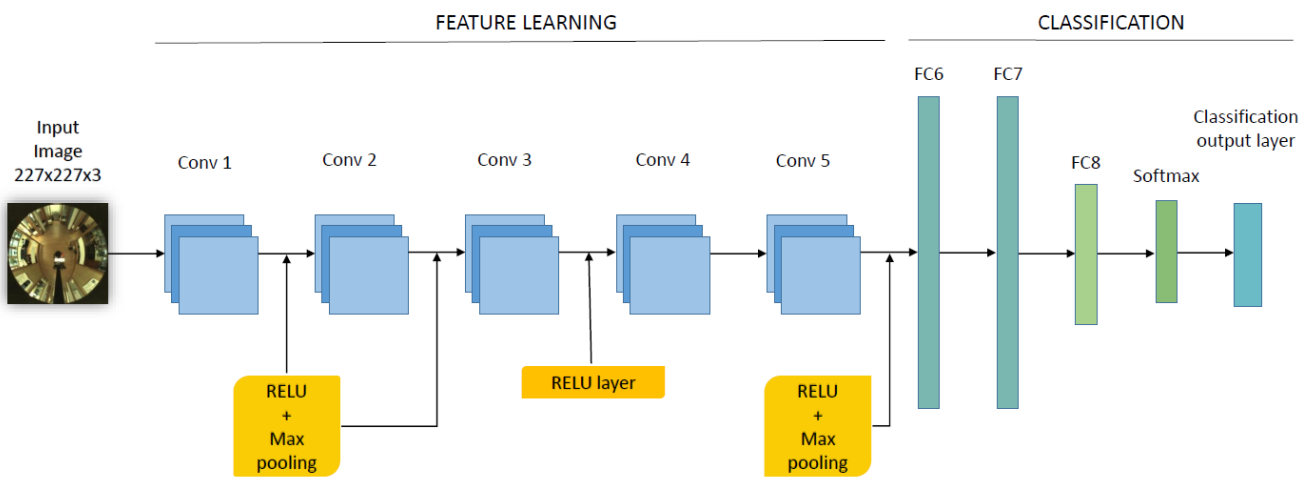


Figura 3-6: Arquitectura simplificada de AlexNet.

3.2.2. Transfer Learning

El Transfer Learning permite reutilizar redes neuronales ya entrenadas aprovechando los conocimientos adquiridos previamente. En nuestro caso hemos partido de Alexnet y hemos realizado una serie de modificaciones según la tarea a realizar.

- **Tarea de clasificación.** Con el fin de clasificar las estancias del laboratorio de la base de datos Friburgo, hemos modificado la última capa, Classification output layer, modificando el número de nodos de salida (de 1000 a 9). Estos 9 nodos de salida corresponden a las 9 estancias del laboratorio.

- **Tarea de regresión.** Alexnet es una red entrenada para la tarea de clasificación, pero podemos hacer que funcione para regresión con la siguiente arquitectura. Nos quedamos hasta la capa fc7 de AlexNet, se añade una FullyConnected Layer con dos nodos para la coordenada X e Y respectivamente y por último, se introduce una regressionLayer. La arquitectura quedaria tal y como se representa en la (Figura **3-7**)

Además, el inputsize de Alexnet es 227x227x3 mientras que el tamaño de nuestras imágenes es de 470x470x3, por lo que si queremos que la red aproveche todos los píxeles de nuestras imágenes, tenemos que modificar el tamaño de la capa de entrada. Más adelante veremos los resultados obtenidos de los diferentes experimentos realizados.

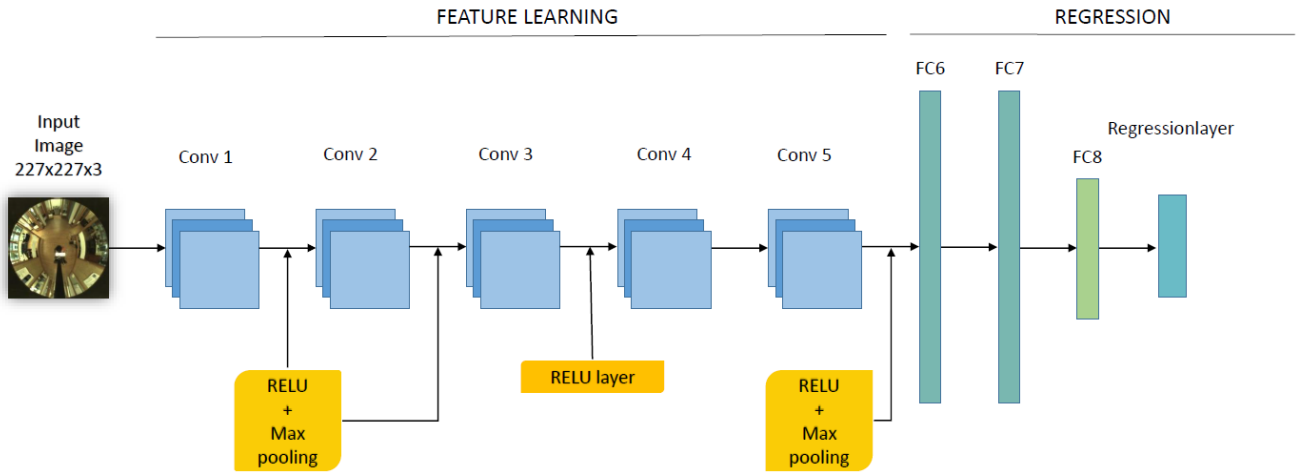


Figura 3-7: Arquitectura Alexnet modificada para regresión.

3.2.3. Vectores descriptores de apariencia global

Se ha empleado una técnica denominada localizacion jerárquica la cual consiste en revolver la tarea de localización en diferentes fases con diferente nivel de granularidad. En este caso, primero se estima en qué habitación se encuentra el robot y posteriormente, se utiliza la técnica del vecino más cercano mediante el uso de vectores descriptores de aparencia global.

Los vectores descriptores de apariencia global son muy populares para resolver tareas de mapeo y localización visual y consisten en extraer información visual global de una imagen de manera que caractericen a ésta. Comúnmente, se han utilizado descriptores basados en métodos analíticos (*hand-crafted*), pero en los últimos años han emergido los descriptores basados en aprendizaje profundo (*deep descriptors*). En este caso, se propone utilizar esta nueva herramienta. Para ello, los vectores se obtienen de las capas intermedias de la red neuronal. El propósito de estos vectores es resolver la tarea para la cual está entrenada la CNN, en nuestro caso para la clasificación de estancias, pero pueden ser extraídos y utilizados para una finalidad distinta como puede ser para la localización del robot mediante la búsqueda del vecino más cercano.

Este trabajo propone utilizar la CNN como modelo jerárquico con el objetivo de: (a) abordar la localización aproximada como problema de búsqueda de habitaciones (capa de alto nivel) partiendo de la imagen de test y (b) obtener descriptores holísticos de las imágenes de entrada. Los descriptores de las imágenes de entrenamiento formarán la capa de bajo nivel, y permiten resolver una localización fina, abordado como un problema de detección de la imagen más similar, con los descriptores holísticos de las imágenes test (también obtenidos mediante la CNN).

3.3. Data Augmentation

Un gran conjunto de datos de entrenamiento es crucial para el rendimiento del modelo. Sin embargo, a veces, el conjunto de datos de entrenamiento disponible es más pequeño de lo necesario y entonces, el modelo no puede ser entrenado adecuadamente para alcanzar la solución deseada. Para resolver este problema, se ha propuesto la técnica de Data Augmentation como método para mejorar el rendimiento del modelo aumentando el número de instancias de entrenamiento y evitando el sobreajuste. El aumento de datos consiste básicamente en la creación de nuevos datos (imágenes) mediante la aplicación de diferentes efectos sobre las imágenes originales. Algunos autores ya han utilizado el aumento de datos para resolver sus tareas de aprendizaje profundo.

Las tranformaciones que se han utilizado en este trabajo son las siguientes:

1. Random Rotation: Consiste en introducir una rotación cualquiera a la imagen, en el caso particular de las imágenes omnidireccionales, esta modificación es muy interesante dado que permite emular lo que ocurre en una situación real cuando el robot cambia su orientación en el plano del suelo.

2. Oscuridad y brillo: Los valores de baja intensidad son reajustados (incremento) para crear imágenes con más brillo. Por otro lado, para crear un efecto de oscuridad, los valores bajos de intensidad son reducidos. Cabe mencionar que dichos efectos nunca son aplicados simultáneamente sobre las imágenes.

3. Ruido gaussiano con varianza de 1e-06 a la imagen en escala de grises.

4. Oclusiones: Con el fin de simular situaciones reales con personas.Este efecto simula los casos cuando ciertas partes de la imagen no se pueden visualizar porque están ocultos por algún motivo, como por ejemplo que alguna persona u objeto se posicione en frente de la cámara. En este trabajo, simulamos dicho efecto introduciendo imágenes geométricas en escala de grises en lugares aleatorios de la imagen.

5. Reflexión: Obtener la fotografía espejo.

6. Blur effect o efecto movimiento. Este efecto ocurre cuando la imagen se capturó en movimiento.

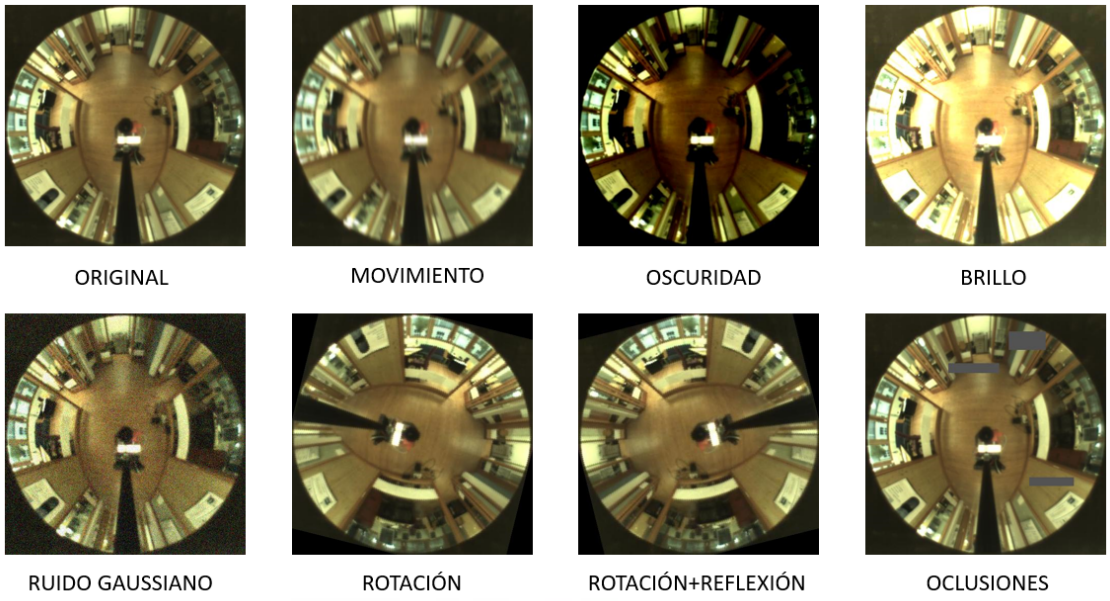


Figura 3-8: Efectos Data Augmentation sobre una imagen ejemplo de la base de datos COLD.

3.4. Optimización Bayesiana

En el campo del aprendizaje máquina (o machine learning), el entrenamiento de una red neuronal no es más que el proceso de aprendizaje de dicha red. El aprendizaje consiste en la identificación de patrones mediante el uso de algoritmos, es por ello que no solo es importante la calidad y el volumen de datos (imágenes) de las que dispone la red, sino también la elección del algoritmo más adecuado.

Las redes neuronales tienen una serie de parámetros que son aprendidos durante el proceso de entrenamiento. Un ejemplo de éstos son los pesos, que van tomando valores según va aprendiendo la red. Estos valores no son configurados por el programador, sino que son adquiridos por la propia red.

Además, existen los hiperparámetros, que sí que son modificables por el programador y que en función de estos podemos obtener mejores o peores resultados para el caso que se esté tratando. Mientras que los parámetros del modelo especifican cómo transformar los datos de entrada en la salida deseada, los hiperparámetros definen cómo se estructura realmente nuestro modelo. Por tanto, un aspecto muy importante a la hora de realizar el entreno de redes neuronales es el de configurar de manera correcta los valores de los hiperparámetros. Por desgracia, estos hiperparámetros dependen unos de otros en formas desconocidas. Lo que hace aún más difícil la búsqueda de los valores óptimos que minimicen la función de pérdida

(*loss function*). La función de pérdida es una herramienta cuyo objetivo es optimizar la capacidad predictiva de la red neuronal, para lo cual se busca minimizar la diferencia entre el *ouput* producido por la misma y el dato real que estemos tratando de predecir. En el presente trabajo se ha utilizado como función de pérdida la *Cross Entropy Loss*.

El diseñador de la red no conoce los mejores hiperparámetros a priori y por tanto es muy común utilizar los mismos valores que se han empleado para problemas previos. Otra alternativa es buscar los mejores valores por ensayo y error, pero no es aconsejable realizar la selección de hiperparámetros a través de prueba y error, ya que cada entrenamiento puede tener una duración considerablemente alta. Por ejemplo, en el presente trabajo, el entrenamiento de la CNN dura de media 20 min, por consiguiente, todo esto supondría un tiempo de computación difícilmente aceptable de asumir.

Por otro lado, se puede recurrir a la búsqueda de rejilla (*grid search*) o búsqueda aleatoria (*random search*). Estas técnicas son ligeramente mejores que el ajuste manual porque el algoritmo realiza una búsqueda de forma autónoma, explorando una serie de valores de hiperparámetros. Sin embargo, incluso estos métodos son bastante ineficientes porque no eligen los siguientes hiperparámetros a evaluar basándose en los resultados anteriores y, en consecuencia, suelen pasar mucho tiempo evaluando hiperparámetros que no mejoran el rendimiento de la red. Los planteamientos bayesianos, en contraste con la búsqueda aleatoria o en cuadrícula, mantienen un historial de resultados de evaluaciones anteriores que utilizan para formar un modelo probabilístico que asigna a los hiperparámetros una probabilidad de ser el óptimo de la función objetivo.

En consecuencia, la optimización de hiperparámetros se ha convertido en un campo activo de investigación. Bergstra *et al.*, [6] proponen la elección de hiperparámetros de forma aleatoria y Falkner *et al.*, [16] combinan las técnicas de optimización Bayesiana y de *Hyperband*. Entre las opciones propuestas por diferentes expertos del campo de *machine learning*, una opción muy aceptada es la utilización de la optimización Bayesiana.

La optimización Bayesiana pertenece a una clase de algoritmos de optimización conocida como *Sequential Model-Based Optimization (SMBO)* que permiten utilizar los resultados de nuestra iteración anterior para mejorar nuestro método de muestreo del siguiente experimento. La optimización Bayesiana en concreto se basa en el teorema de Bayes, el cual es una forma de calcular probabilidades condicionadas a un dato conocido. Esto significa que el dato es fijo y la hipótesis es aleatoria. En el contexto de la optimización de hiperparámetros la aproximación bayesiana se beneficia de la información que nuestro modelo aprende durante el proceso de optimización.

La idea consiste en escoger alguna hipótesis previa sobre cómo se comportarán nuestros

hiperparámetros. Esto lo utilizaremos como datos de partida. A partir de aquí, se buscarán los parámetros óptimos, reforzando y actualizando nuestra hipótesis previa. Esta actualización de hiperparámetros se basa en una medición continua.

La optimización bayesiana debe presentar un compromiso entre la exploración y la explotación, asegurando que se comprueban los rangos de valores más relevantes. Una vez encontrada la región candidata a tener el mínimo, se procederá a buscar el valor óptimo de ella, lo cual, presenta un método de búsqueda más eficiente que los propuestos hasta el momento.

3.4.1. Hiperparámetros

El presente trabajo propone la optimización de hiperparámetros cuya variación puede resultar crucial para llevar a cabo el entrenamiento de CNNs tanto para clasificación como para regresión. A continuación, se indican los hiperparámetros que se han considerado variar a través de la optimización bayesiana:

- **Initial Learn Rate.** Valor inicial del ritmo de aprendizaje o *learn rate*. El *learn rate* controla cómo de rápido el modelo se adapta al problema. Un valor muy pequeño puede provocar que no se llegue al valor óptimo, necesitando un mayor número de épocas. Un valor muy grande puede provocar una convergencia en un mínimo local. Ya que al ser los pasos más grandes se podría dar el caso de saltarse el mínimo global. Para ajustar este hiperparámetro de forma progresiva se hace uso del *Learn Rate Drop Period* y *Learn Rate Drop Factor*.

- **Learn Rate Drop Period.** Número de épocas para las cuales se realiza una reducción del *learn rate*.

- **Learn Rate Drop Factor.** Factor por el cual se quiere reducir el *learn rate* con la frecuencia establecida por el *Learn Rate Drop Period*.

- **Momentum.** Permite suavizar el proceso de aprendizaje acumulando el gradiente de los pasos anteriores para determinar la dirección a seguir. De manera que puede mantener la tendencia global de los puntos, evitando que un mal dato desencadene en un mínimo local.

- **Max Epochs.** Número máximo de épocas del entrenamiento. Un número elevado de épocas de entrenamiento provocará un sobreajuste o *overfitting* en la red neuronal. Es decir, cuando un modelo aprende los detalles y el ruido de los datos de entrenamiento hasta el punto de que repercute negativamente en el rendimiento del modelo con los nuevos datos o datos de test. Por otro lado, si el número de épocas es bajo puede ocurrir lo que se conoce como *underfitting*, cuando un modelo no puede modelar los datos de entrenamiento ni generalizar a los nuevos datos.

- **L2 Regularization.** Factor de regularización de la L2 (decadencia de peso). Valor escalar positivo que añade un término de regularización que penaliza (o disminuye el coeficiente de) todos los pesos de la función de pérdida, consiguiendo que el modelo generalice los datos y no se sobreajuste.
- **Gradient Decay Factor (β_1).** Factor de disminución del gradiente o tasa de descenso de la media variable para Adam solver.
- **Squared Gradient Decay Factor (β_2).** Factor de disminución del gradiente al cuadrado para Adam y RMSProp solvers.
- **Epsilon.** Compensación del denominador para Adam y RMSProp solvers. Es un valor muy pequeño que previene de dividir entre cero en la implementación.

3.4.2. Algoritmos de resolución

Además de los hiperparametros, el programador puede elegir entre diferentes algoritmos de resolución, como pueden ser:

Stochastic Gradient Descent with Momentum (sgdm)

El algoritmo de descenso de gradiente estocástico puede oscilar a lo largo del camino de descenso más rápido hacia el óptimo. Añadir un término de momento a la actualización de los parámetros es una forma de reducir esta oscilación. La actualización del descenso de gradiente estocástico con el momento es:

$$\theta_{\ell+1} = \theta_\ell - \alpha \nabla E(\theta_\ell) + \gamma(\theta_\ell - \theta_{\ell-1})$$

dónde ℓ es el número de la iteración, $\alpha > 0$ es el learning rate, θ es el vector de parámetros, $\nabla E(\theta)$ es el gradiente de la función de perdida y γ determina la contribución del gradiente del punto anterior a la iteración actual. El término del momento $\gamma(\theta_\ell - \theta_{\ell-1})$ permite reducir la oscilación a lo largo de la trayectoria de descenso hacia el óptimo.

Root Mean Square Propagation (RMSProp)

El descenso de gradiente estocástico con momento utiliza una única tasa de aprendizaje para todos los parámetros. Otros algoritmos de optimización tratan de mejorar el entrenamiento de la red utilizando tasas de aprendizaje que difieren según el parámetro y que pueden adaptarse automáticamente a la función de pérdida que se está optimizando. El RMSProp (propagación cuadrada media de la raíz) es uno de esos algoritmos. Mantiene una media variable de los cuadrados de los elementos de los gradientes de los parámetros,

$$\nu_\ell = \beta_2 \nu_{\ell-1} + (1 - \beta_2) \left[\nabla E\left(\theta_\ell\right)\right]^2$$

dónde ν es la media variable y β_2 su tasa de descenso (Squared Gradient Decay Factor).

El algoritmo RMSProp utiliza este promedio móvil (ν) para normalizar las actualizaciones de cada parámetro individualmente,

$$\theta_{\ell+1} = \theta_\ell - \frac{\alpha \nabla E(\theta_\ell)}{\sqrt{v_\ell} + \epsilon}$$

dónde la división se realiza por elementos. El uso de RMSProp disminuye efectivamente las tasas de aprendizaje de los parámetros con grandes gradientes y aumenta las tasas de aprendizaje de los parámetros con gradientes pequeños. Epsilon (ϵ) es una pequeña constante añadida para evitar la división por cero cuyo valor se puede modificar puesto que es un hiperparámetro.

Adam

Adam utiliza una actualización de los parámetros similar a la de RMSProp, pero con un término de momento añadido. Mantiene una media móvil en función de los elementos tanto de los gradientes de los parámetros como de sus valores cuadrados:

$$m_\ell = \beta_1 m_{\ell-1} + (1 - \beta_1) \nabla E(\theta_\ell)$$

$$\nu_\ell = \beta_2 \nu_{\ell-1} + (1 - \beta_2) [\nabla E(\theta_\ell)]^2$$

dónde m es otra media variable. Se pueden especificar los factores de disminución β_1 y β_2 usando los hiperparametros Gradient Decay Factor y Squared Gradient Decay Factor, respectivamente. Adam usa los promedios móviles para actualizar los parámetros de la red como:

$$\theta_{\ell+1} = \theta_\ell - \frac{\alpha m_\ell}{\sqrt{v_\ell} + \epsilon}$$

Si los gradientes a lo largo de muchas iteraciones son similares, entonces el uso de un promedio móvil del gradiente permite que las actualizaciones de los parámetros tomen momento en una cierta dirección. Si los gradientes contienen principalmente ruido, entonces el promedio móvil del gradiente se hace más pequeño, y así las actualizaciones de los parámetros también se hacen más pequeñas.

4 Métodos desarrollados

Como ya se ha explicado con anterioridad, los experimentos han consistido en obtener una red neuronal utilizando las diferentes técnicas explicadas en el capítulo anterior con el objetivo de calcular la posición en la cual la imagen omnidireccional fue capturada dentro de un entorno.

4.1. Introducción a la tarea de localización

El proceso de localización del robot se ha abordado desde dos perspectivas:

- **Localización jerárquica.** En la cual se resuelve la tarea de localización en dos etapas. En la primera de ellas se lleva a cabo un reconocimiento de la estancia en la que se encuentra el robot para lo cual se hace uso de una CNN entrenada para desempeñar la tarea de clasificación tal y como se describe en la sección 4.2. En la segunda etapa, se estiman las coordenadas dentro de la estancia preseleccionada. Para llevar a cabo esta etapa se hace uso de descriptores globales extraídos de la CNN y la técnica del vecino más cercano tal y como se explica en la sección 4.6.

- **Localización global.** La localización global consiste en determinar las coordenadas en las que se encuentra el robot dentro de una trayectoria completa, sin dividir el problema en diferentes etapas. Para ello, en la sección 4.3 se entrena una CNN para desempeñar la tarea de regresión la cual se caracteríza por tener como etiqueta las coordenas donde se capturó la imagen.

Cabe destacar que se ha empleado la CNN para un objetivo u otro en función del tipo de localización a realizar, alterando aquellas capas necesarias para la consecución del propósito.

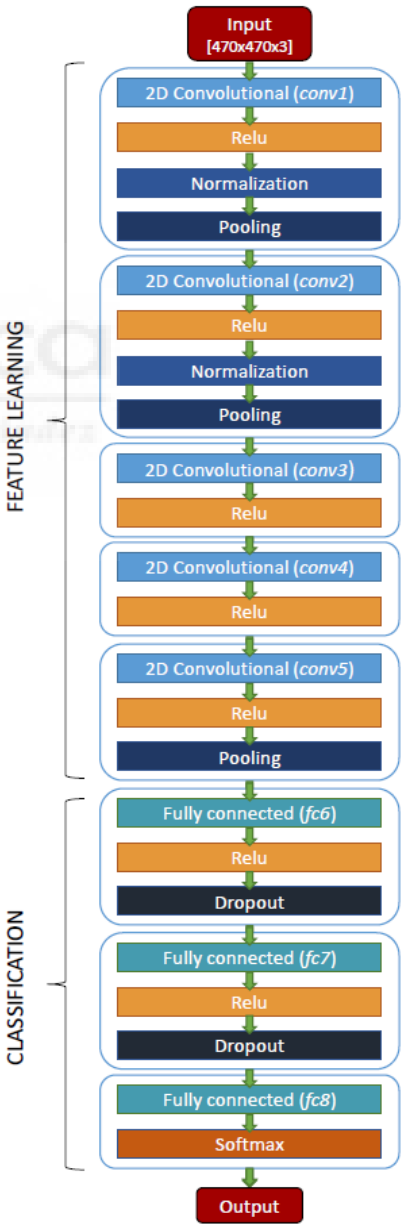


Figura 4-1: Arquitectura detallada de Alexnet.

4.2. Entrenamiento de la CNN para la tarea de clasificación

4.2.1. CNN obtenida a partir de la adaptación de la arquitectura Alexnet y entrenamiento desde cero

El objetivo es obtener una CNN que sea capaz de identificar la estancia dentro de la cuál se capturó la imagen de entrada. Como punto de partida, se toma la red Alexnet y, en primer lugar, se adapta su arquitectura (Figura **4-1**), modificando su capa inicial para adaptarla al tamaño de las imágenes del dataset usado (470x470x3 píxeles) y las capas totalmente conectada (*fc8*) y *softmax*, de modo que la CNN modificada sea capaz de clasificar la imagen de entrada en la estancia correspondiente (el dataset que se usará para el entrenamiento fue capturado en un edificio con 9 estancias).

Una vez disponible la red modificada, se entrena desde cero (sin considerar los pesos y umbrales necesarios de Alexnet) con las imágenes de entrenamiento de dicho dataset, de modo que la red resultante de este entrenamiento sea capaz de resolver este problema de clasificación. Durante el proceso de entrenamiento, se utilizan las imágenes omnidireccionales como datos de entrada de la red y la habitación donde fueron capturadas como etiqueta. De esta manera se llevaría a cabo una actualización de los pesos de cada nodo con cada una de las imágenes de entrada (Figura **4-2**). Este método de entrenamiento se conoce como aprendizaje supervisado al ir las imágenes asociadas a una etiqueta con la salida correcta, en función del error cometido en la predicción de la CNN se actualizarán los pesos de cada uno de los nodos.

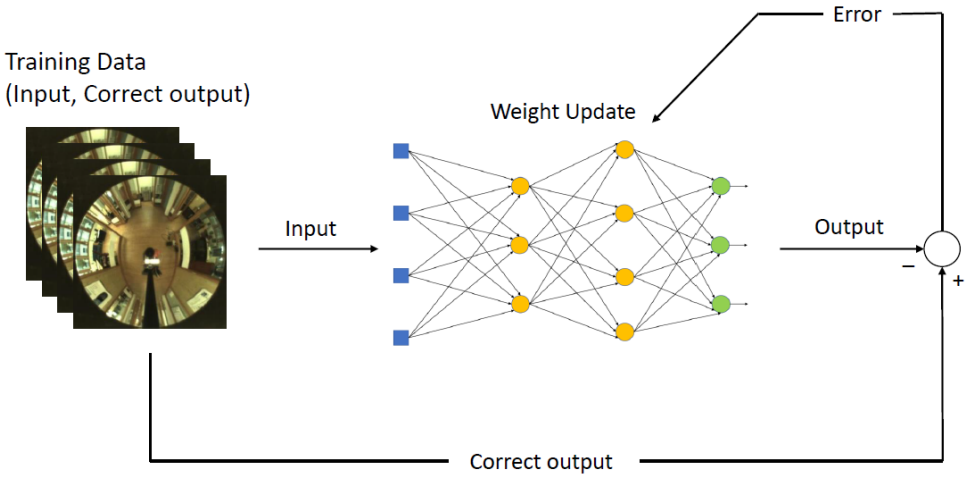


Figura 4-2: Aprendizaje supervisado de una red neuronal.

4.2.2. CNN obtenida a partir de la adaptación de la arquitectura Alexnet y transfer learning

Sin modificar la capa de entrada

Para utilizar Alexnet sin modificar su primera capa, debemos redimensionar las imágenes de entrada de 470x470x3 a 227x227x3 píxeles ya que es el input size que viene determinado por la primera capa de Alexnet.

Para ello utilizamos una variable auxiliar Train y Validation donde guardamos las imágenes de entrenamiento y de validación redimensionadas. Posteriormente, se realiza el transfer learning de Alexnet, con el nuevo número de categorias a clasificar (9 estancias). Para ello, es necesario modificar la capa totalmente conectada (*fc8*) para 9 categorías y la capa de salida (*output*) siendo esta la capa que lleva a cabo la clasificación (Figura **4-3**). Una vez esté modificada la arquitectura se lleva a cabo el reentrenamiento, partiendo de los pesos iniciales de Alexnet, y actualizándolos con el conjunto de entrenamiento de nuestro dataset para que la red aprenda a resolver la nueva tarea (clasificación en 9 estancias). Utilizando de nuevo el aprendizaje supervisado, método que será empleado en cada uno de los entrenamientos.

Modificando la capa de entrada

Modificamos la capa de entrada con el objetivo de aprovechar toda la resolución de las imágenes, también la capa totalmente conectada (*fc8*) para 9 categorías y la capa de salida (*output*). Para adaptar correctamente los tamaños de salida de las capas completamente conectadas debemos reestablecer la capa totalmente conectada (*fc6*) (Figura **4-4**).

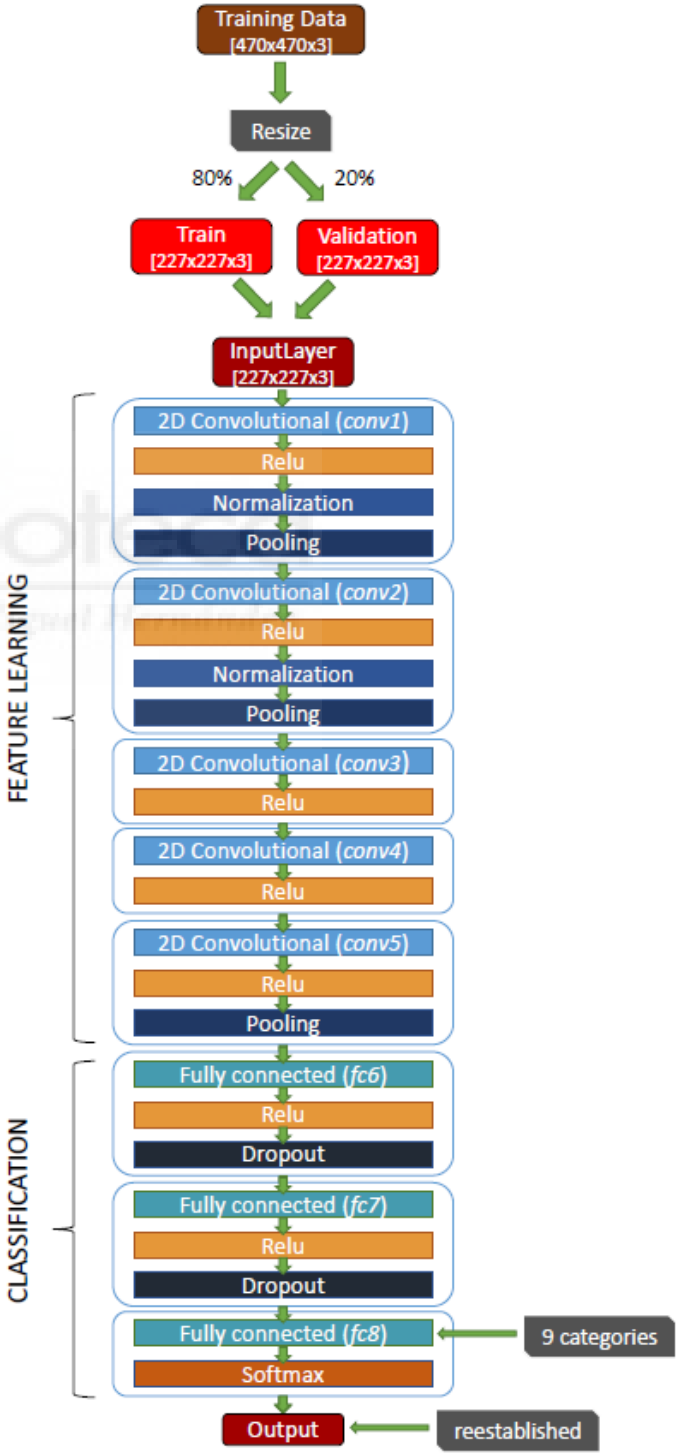


Figura 4-3: Sin modificar capa de entrada.

Es importante remarcar que la capa de entrada es distinta al resto ya que simplemente se encarga de transmitir las señales de entrada a los siguientes nodos, de manera que carecen de pesos asociados a ellas. Por lo que el mayor número de píxeles de la capa de entrada se traducirá en una mayor información a procesar con las ventajas y desventajas que ello conlleva.

Normalizando la cantidad de datos de cada clase usados para el entrenamiento

Esta vez vamos a partir de la mejor red obtenida hasta el momento pero vamos a cambiar los datos de entrada.

La diferencia en el número de imágenes capturadas para cada habitación puede llevarnos a error ya que la CNN podrá aprender qué probabilidad hay de que el robot esté en un lugar u otro. Para evitar este problema, se realiza una normalización en la cantidad de datos de entrenamiento correspondientes a cada estancia, estableciendo como criterio que el número de imágenes de cada habitación sea similar.

A priori, en una tarea de clasificación de objetos, no nos interesa que la CNN aprenda la probabilidad de que aparezaca un objeto de una clase o otra. Pero en la clasificación de estancias, puede que si, ya que tendremos más imágenes de las estancias más visitadas por el robot, como es el caso del pasillo que es el enlace de unión de las diferentes estancias.

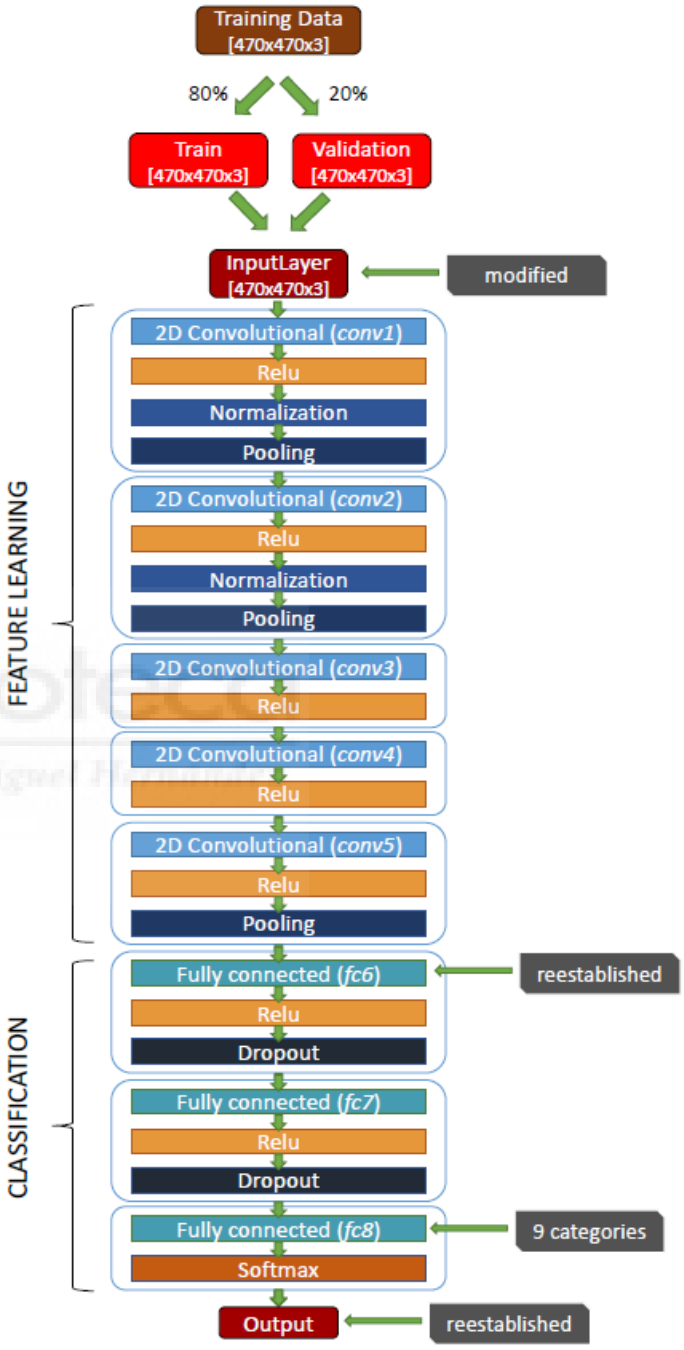


Figura 4-4: Modificando capa de entrada.

Por otro lado, normalizar la cantidad de imágenes nos obliga a prescindir de una gran cantidad de información que puede ser crucial para la buena localización del robot. Aún así se ha realizado este experimento para comprobar la incidencia de la normalización de datos en tareas de mapeo y localización.

Estancia	1P0-A	2P01-A	2P02-A	CR-A	KT-A	LO-A	PA-A	ST-A	TL-A
Orig.	44	46	31	238	46	26	57	30	38
Norm.	22	23	31	27	23	26	29	30	19

Tabla 4-1: Cantidad de imágenes de entrenamiento originales frente a normalizadas.

4.3. Entrenamiento de la CNN para la tarea de regresión

Se toma como punto de partida la mejor CNN para clasificación obtenida en el apartado anterior, con el objetivo de realizar un transfer learning para adaptarla a la tarea de regresión. Para ello, se modifican las tres últimas capas de Alexnet para realizar la nueva tarea. Es decir, se cambia la capa (*fc8*), se elimina la *Softmax* y se reemplaza la classificationLayer (*Output*) por una RegressionLayer (Figura **4-5**).

La principal diferencia entre regresión y clasificación es que en clasificación se trabaja con categorias mientras que en regresión con valores numéricos, en nuestro caso, las coordenadas cartesianas del plano bidimensional por el cual navega el robot. Nuestra CNN se caracteriza por tener dos nodos en la capa totalmente conectada (*fc8*), correspondientes a las coordenadas x e y, aunque también sería interesante crear una CNN para cada coordenada cartesiana. En clasificación el objetivo era obtener la máxima precisión posible, en regresión es minimizar el error medio que consiste en la suma cuadrática de los errores de las coordenadas x e y.

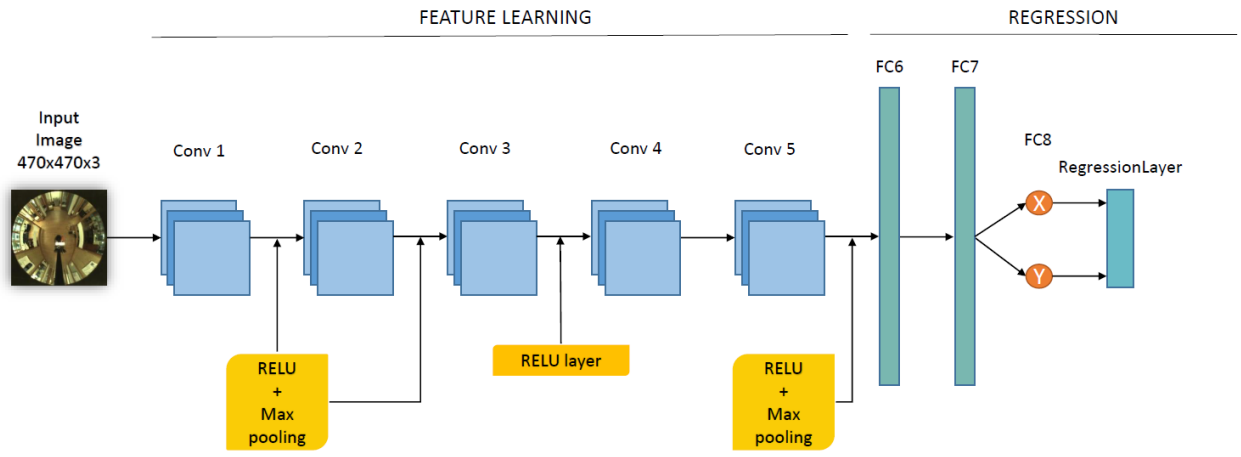


Figura 4-5: Arquitectura para regresión.

4.4. Data Augmentation

El Data Augmentation tiene como objetivo generar un mayor número de imágenes de entrenamiento. Se parte del dataset de entrenamiento y se realizan perturbaciones sobre las imágenes. Se aplican un total de siete efectos los cuales fueron definidos en la sección 3.3. Estos efectos se pueden combinar entre sí dando lugar a todas las combinaciones posibles, lo cual en nuestro caso dará lugar a 384 imágenes por cada imagen de entrenamiento. El dataset de entrenamiento cuenta con 556 imágenes y el obtenido tras el data augmentation con 213.504 imágenes. En el efecto rotación se realizan rotaciones de 45 grados desde 45 hasta 315 grados. Este efecto es de especial interés en las imágenes omnidireccionales ya que simula la propia rotación del robot, no teniedo importancia la orientación del mismo para determinar el lugar en el que se encuentra.

El Data Augmentation realizado en el presente trabajo consiste en la aplicación de una serie de efectos a las imágenes de entrenamiento (rotación, brillo u oscuridad, ruido gaussiano, oclusiones, reflexión y efecto movimiento). En la siguiente figura **4-6** se representa el algoritmo que lleva a cabo el data augmentation. Partiendo de las imágenes originales de entrenamiento se aplica a cada una de las imágenes una rotación de 45 grados de forma sucesiva hasta llegar a 315 grados y se guardan en el nuevo dataset. Posteriormente, a las imágenes anteriores se les aplica efecto brillo y oscuridad, cabe destacar que dichos efectos nunca son aplicados simultáneamente sobre las mismas imágenes. Las nuevas imágenes generadas son almacenadas, tanto estas imágenes como las anteriores son pasadas por un filtro de ruido gaussiano y se vuelven a guardar. Se sigue la misma metodología con todos los efectos restantes (oclusiones, reflexión y efecto movimiento) de manera que al final del algoritmo tendremos un dataset con las imágenes originales y todas las combinaciones posibles de las perturbaciones presentadas.

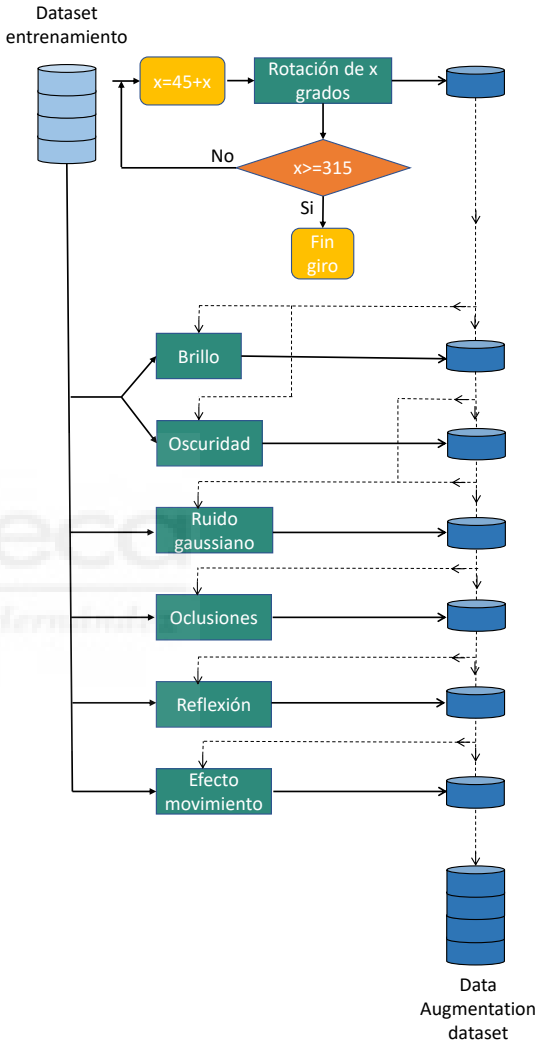


Figura 4-6: Realización del Data Augmentation.

4.5. Optimización Bayesiana

El objetivo de la optimización bayesiana es encontrar la configuración de hiperparámetros óptima para realizar el entrenamiento de una red neuronal convolucional y un dataset de entrenamiento determinado. Esta técnica permite automatizar la búsqueda de aquellos valores de hiperparámetros más idóneos según la tarea a desempeñar. La tarea de localización se puede abordar desde diferentes vertientes, en este trabajo se ha hecho uso de la localización jerárquica en la que primero se que clasifica la estancia en la que se encuentra el robot y posteriormente se emplea la técnica del vecino más cercano mediante el uso de vectores descriptores de apariencia global y la regresión que consiste en entrenar una red neuronal convolucional mediante el uso de imágenes y las coordenadas en las que se encuentra el robot.

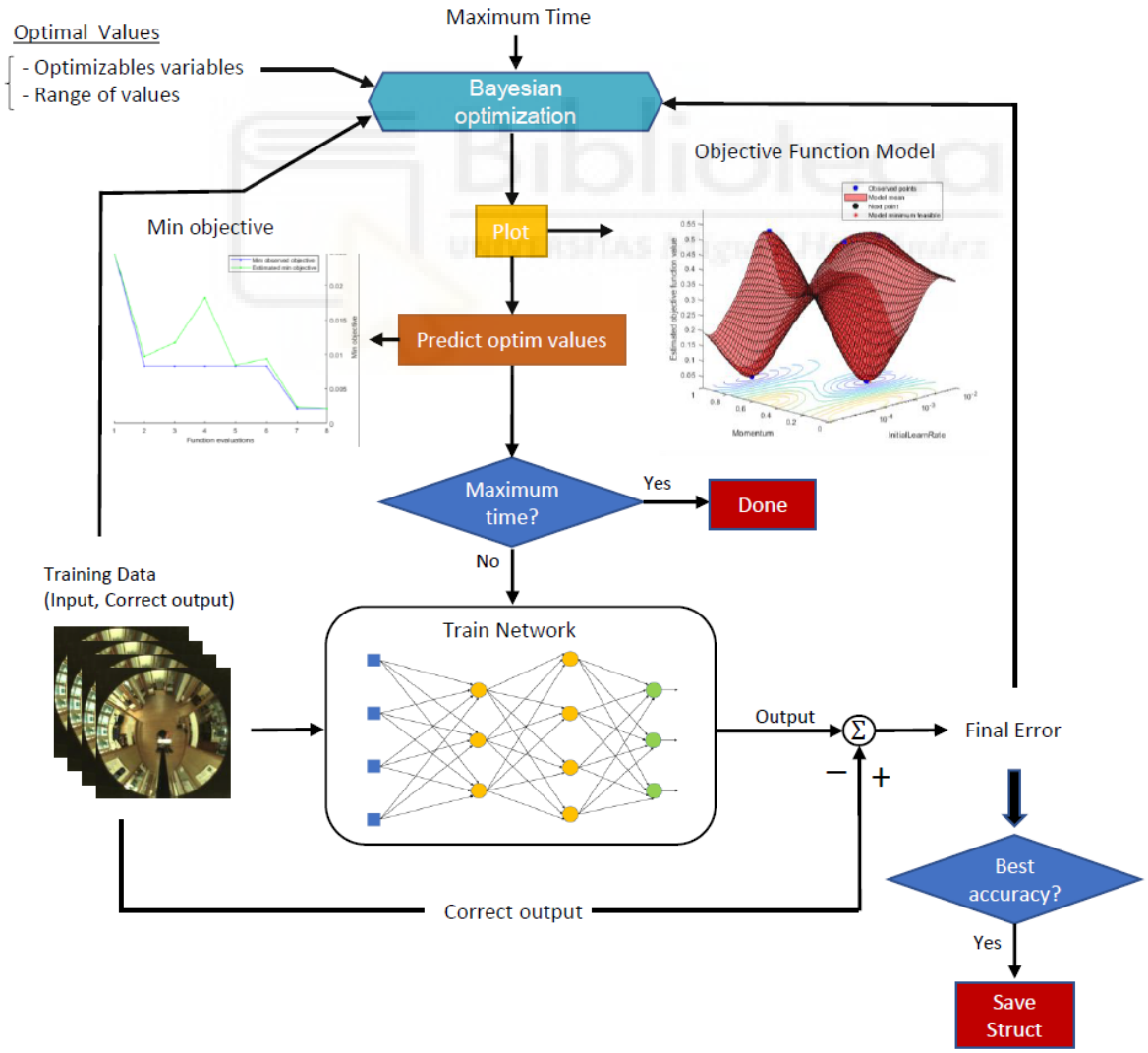


Figura 4-7: Diagrama de flujo de la optimización bayesiana.

El algoritmo empleado para llevar a cabo la optimización bayesiana viene representado por el diagrama de flujo (Figura **4-7**). Las entradas necesarias para la optimización bayesiana son las imágenes con sus etiquetas, las variables a optimizar con su rango de valores, el máximo tiempo de ejecución del algoritmo (este parámetro está ligado al número de puntos explorados) y por último el error de entrenamiento asociado a esos hiperparámetros con esos valores. Los errores asociados a los valores de los hiperparámetros son representados en el gráfico del modelo de la función objetivo con el fin de encontrar el mínimo error. Este gráfico solo es posible si tenemos una (Figura **4-8**) o dos variables (Figura **4-9**) a optimizar, en el caso de más hiperparámetros se encontrarán los valores óptimos, pero no se podrán visualizar. A partir del gráfico del modelo de la función objetivo el algoritmo es capaz de elegir el próximo punto de exploración que considera candidato a óptimo y representa el valor del mínimo observado respecto al mínimo estimado (Figura **4-10**).

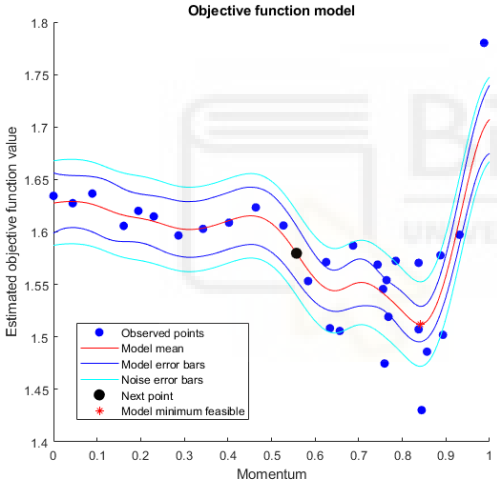


Figura 4-8: Una variable a optimizar.

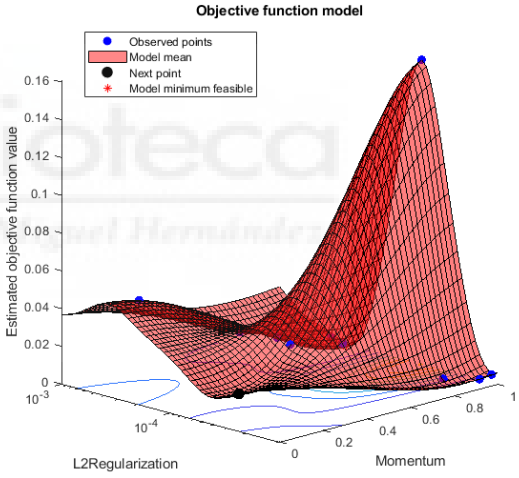


Figura 4-9: Dos variables a optimizar.

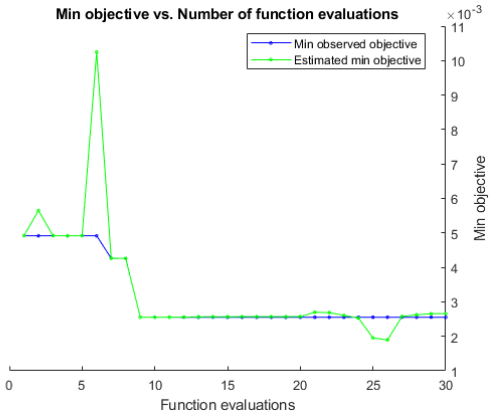


Figura 4-10: Mínimo observado frente estimado.

4.6. Vectores descriptores de apariencia global

Los vectores descriptores de apariencia global permiten representar una imagen con un solo vector descriptor, aportando información de toda la escena. Existen diferentes algoritmos para la creación de vectores descriptores, por ejemplo Payá *et al.*, [37] emplean un descriptor basado en la firma de Fourier con el fin de realizar tareas de localización usando un algoritmo Monte Carlo. Otros métodos para describir la apariencia global de las escenas pueden ser la transformada de Radon descrita inicialmente en [41] y HOG (Histogram of Oriented Gradients) descrito inicialmente por [13] para la detección de personas. Ambas técnicas han sido ampliamente empleadas para desarrollar tareas propias de robots móviles. Berenguer *et al.*, [5] demuestran la eficacia y robustez de la transformada Radon creando un metodo de localizacion 2D usando un descriptor de apariencia global, además, Fernández *et al.*, [37] proponen el uso de HOG para llevar a cabo la localización al aire libre empleando la base de datos de Google Street View.

El empleo de las redes neuronales convolucionales no solo se limitan a realizar tareas de clasificación o regresión, sino que también se pueden emplear para la descripción de la apariencia global de las imágenes. En cada una de las capas de una red neuronal se generan unos vectores descriptores que sirven para llevar a una tarea específica, estos vectores pueden ser extraídos y ser empleados como vectores descriptores de apariencia global. Esta técnica ha sido empleada por diferentes autores (Cebollada *et al.*, [8]) que han demostrado su precisión y fiabilidad.

La tarea de localización se puede abordar desde diferentes técnicas entre las cuales podemos destacar la clasificación en estancias como una localización "gruesa", la regresión y el uso de vectores descriptores como una localización "fina"donde se trata de encontrar la posición donde se encuentra el robot de manera más precisa. La localización gruesa y la localización fina se combinan para dar lugar a la localización jerárquica propuesta en este trabajo de fin de grado.

Pero cabe recordar que se propone realizar la localización de dos formas distintas:

- Forma 1: Localización global en la cual se determinan las coordenadas en las que se capuró la imagen mediante una CNN entrenada para la tarea de regresión. Es decir, la red es entrenada para predecir la posición del robot tal y como se explica en la sección 4.3.

- Forma 2: Localización jerárquica con el objetivo de: (a) abordar la localización aproximada como problema de búsqueda de habitaciones (capa de alto nivel) partiendo de la imagen de test y (b) obtener descriptores holísticos de las imágenes de entrada. Los descriptores de las imágenes de entrenamiento formarán la capa de bajo nivel, y

permiten resolver una localización fina, abordado como un problema de detección de la imagen más similar, con los descriptores holísticos de las imágenes test (también obtenidos mediante la CNN).

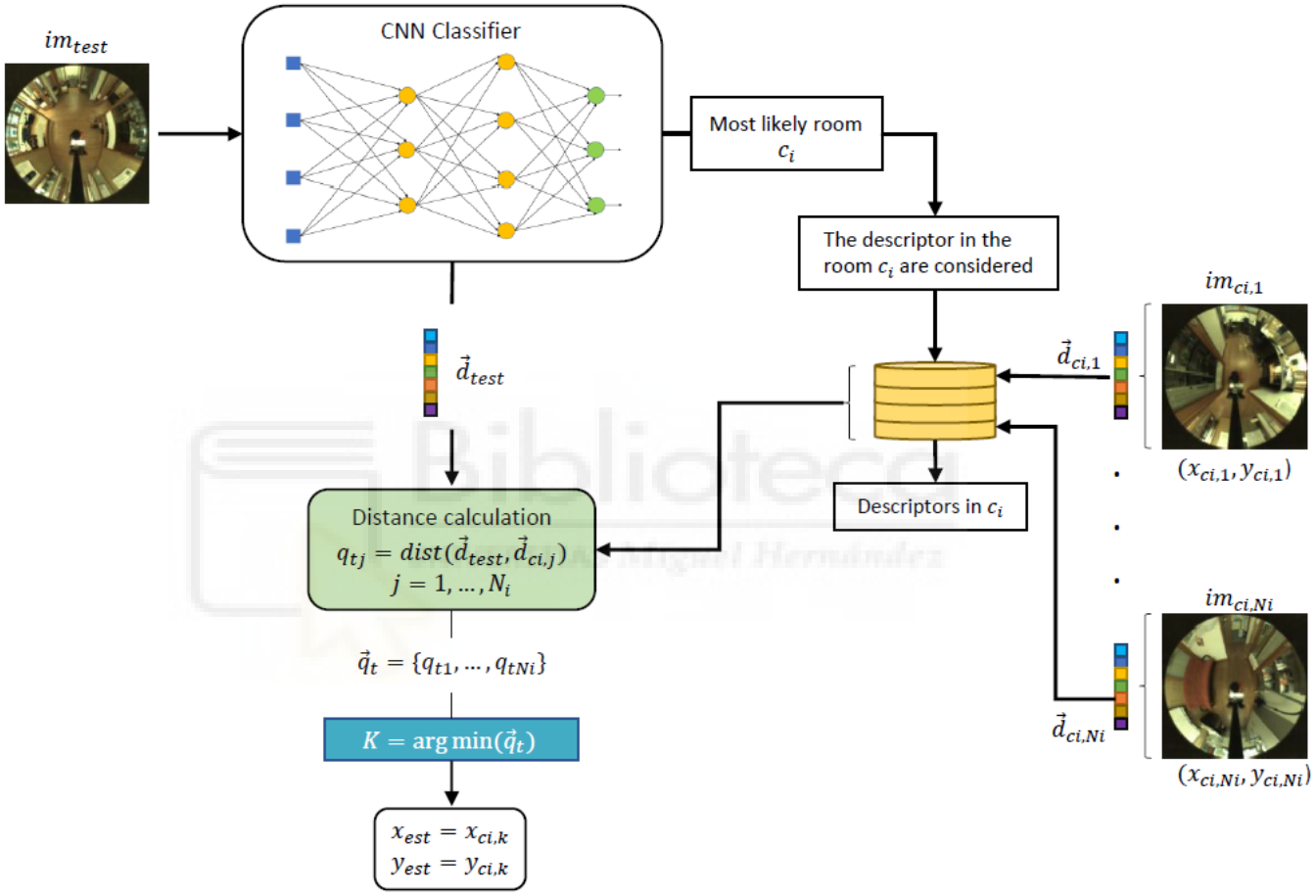


Figura 4-11: Diagrama de la localización jerárquica.

En cuanto a la localización jerárquica, las capas de alto nivel permiten una **localización gruesa** y las capas de bajo nivel una **localización fina**. El paso grueso proporciona una localización más rápida y el paso fino considera información más precisa que se utiliza para realizar la localización fina. La localización jerárquica propuesta se lleva a cabo tal y como muestra el diagrama de la fig. **4-11**. En primer lugar (paso de localización gruesa), se introduce una imagen de test im_{test} en la CNN y se estima la habitación más probable c_i en la que se capturó la imagen a partir de la información de las capas de salida. Al mismo tiempo, la CNN reentrenada también es capaz de proporcionar descriptores holísticos a partir de las capas intermedias. Posteriormente, tras identificar la habitación, se lleva a cabo una localización más precisa (etapa de localización fina). En esta etapa se selecciona uno

de los descriptores $\vec{d}_{test}$ y se compara con los descriptores $D_{c_i} = \{\vec{d}_{c_i,1}, \vec{d}_{c_i,2}, ..., \vec{d}_{c_i,N_i}\}$ del conjunto de datos de entrenamiento que pertenecen a la habitación seleccionada c_i mediante la distancia euclídea y se guarda el descriptor más similar $\vec{d}_{c_i,j}$. Por último, la posición en la que se capturó la imagen de prueba se estima como las coordenadas en las que se capturó $im_{c_i,k}$.

5 Resultados de los experimentos

A continuación, se mostrarán los detalles y resultados de los experimentos descritos en el capítulo 4. Así como se especificarán los conjuntos de entrenamiento y testeo de la red.

5.1. Conjunto de datos de entrenamiento y test

Estancia	**1P0-A**	**2P01-A**	**2P02-A**	**CR-A**	**KT-A**	**LO-A**	**PA-A**	**ST-A**	**TL-A**
Conjunto entreno 1	44	46	31	238	46	26	57	30	38
Conjunto entreno 2	22	23	31	27	23	26	29	30	19
Conjunto entreno 3	16896	17664	11904	91392	17664	9984	21888	11520	14592
Conjunto test 1	155	230	135	1040	254	177	222	133	249
Conjunto test 2	171	222	116	1139	244	182	309	172	252
Conjunto test 3	138	296	127	1110	272	143	313	220	257

Tabla 5-1: Tabla resumen del conjunto de datos de entrenamiento y testeo.

El dataset de nublado se muestrea con el objetivo de obtener un conjunto de datos resultante con una distancia media de 40 cm entre imágenes consecutivas, lo cual dará como resultado el conjunto de entrenamiento 1, con 519 imágenes. El conjunto de entrenamiento 2, se obtiene normalizando la cantidad de imágenes de cada estancia del conjunto de entrenamiento 1, obteniendo así un total de 230 imágenes. Adicionalmente, al conjunto de entrenamiento 1 se le realizará el aumento de datos descrito en la sección 4.4, dando lugar al conjunto de entrenamiento 3, que contiene 213.504 imágenes. Estos conjuntos de datos se usarán, individualmente, para entrenar las CNNs. De este modo, será posible conocer el efecto del aumento de datos en el desempeño de la red. Por su parte, se consideran diferentes conjuntos de test: el conjunto test 1, que contiene imágenes capturadas en condiciones de nublado (ruta del robot diferente a la de entrenamiento 1), con un total de 2.595 imágenes;

el conjunto test 2, que contiene todas las imágenes capturadas en condiciones soleado (2.807 imágenes) y el conjunto test 3, con todas las imágenes capturadas por la noche (2.876 imágenes). Por tanto, el entrenamiento de la red se realiza, en todo caso, con imágenes capturadas en condiciones de iluminación nublado, y el test se realizará en tres condiciones distintas: nublado, soleado o noche, con lo que será posible testar la robustez de la red ante este tipo de cambios de iluminación en el entorno.

A modo resumen, en la Tabla 5.1 se muestra el número de imágenes en cada estancia de cada uno de los conjuntos de datos mencionados anteriormente. Además, cabe recordar que los conjuntos de entrenamiento son subdivididos en entrenamiento (80 %) y validación del entrenamiento (20 %).

5.2. CNN para clasificación

De forma general y antes de entrar en más detalle con los experimentos. Cabe mencionar que para todos los entrenamientos de clasificación, se entrena con pares imagen entrenamiento - etiqueta (número) de habitación, y durante la fase de test se considera acierto si la red devuelve como habitación más probable aquella en la que realmente se capturó la imagen de test (de acuerdo con el ground truth del dataset). De este modo, el accuracy(%) o exactitud que aparece en las gráficas expresa el porcentaje de veces que la red ha clasificado correctamente la imagen test de entrada.

5.2.1. Experimentos iniciales

Con el fin de encontrar la mejor red neuronal convolucional para realizar la clasificación en estancias se han realizado diferentes experimentos, cada uno de los entrenamientos va a realizar 90 iteraciones en 30 epochs y con un *Initial Learn Rate* de 0.001. El algoritmo de resolución empleado en cada uno de los experimentos para clasificación es el *Stochastic Gradient Descent with Momentum* (SGDM). Además, cabe destacar que en estos experimentos se prescinde del Data Augmetation y de la optimización Bayesiana. Como se ha descrito

en la sección 4.1.1, en el primer experimento se genera una CNN a partir de la arquitectura Alexnet, modificando su capa inicial (para adaptarla a la resolución real de nuestras imágenes) y finales. Además, se prescinde del aprendizaje previo de esta red, comenzando el proceso de entreamiento desde cero. De este modo, este modelo es entrenado con el dataset de entrenamiento 1 el cual cuenta con 556 imágenes en condición lumínica de nublado (Tabla 5.1, conjunto de entreno 1). La exactitud obtenida en dicho entrenamiento fue de 95.5 % con un tiempo de computación de 14 minutos aproximadamente. En el testeo de dicho modelo obtenemos un buen resultado para imágenes capturadas de noche (94,09 %), en cambio para las imágenes obtenidas en condiciones soleadas la exactitud baja a un 59,92 %. Los resultados de este experimento aparecen en la Figura **5-1** con la etiqueta expC1.

El segundo experimento se parte de Alexnet utilizando la técnica del transfer learning y se modifican únicamente las capas finales. De este modo, es necesario redimensionar las imágenes de entrada de 470x470x3 a 227x227x3. Con este cambio, el tiempo necesario para el entrenamiento desciende notablemente a 3 minutos y medio con una exactitud de entrenamiento del 98.2 % y una exactitud de testeo mucho mejor a la anterior tal y como se puede observar en la Figura **5-1** con la etiqueta expC2.

Otro experimento interesante consiste en modificar la capa de entrada de la red neuronal de 227x227x3 a 470x470x3 para no tener que reducir el tamaño las imágenes de entrada. De esta forma la CNN dispondrá de una cantidad mayor de datos (píxeles) con los que entrenar. En este el tercer experimento también se hará uso del transfer learning a partir de Alexnet. De esta forma se consigue una exactitud de entrenamiento del 96.4 % que pese a ser inferior a la del segundo experimento los resultados de testeo son un poco mejores tal y como aparece en la Figura **5-1** con la etiqueta expC3. Cabe destacar que la principal desventaja de aumentar la cantidad de píxeles con los que tiene que trabajar la red neuronal se traduce en un aumento del tiempo computacional (14 minutos), pero a su vez también aumenta la exactitud de las imágenes de testeo. Además, si comparamos estos resultados con los de la red neuronal desde cero, demostramos que aunque modifiquemos la capa de entrada, los conocimientos de las capas aguas abajo no desaparencen.

Por otro lado, si se normaliza la cantidad de imágenes de entrenamiento de cada una de las clases (estancias) y se parte de la anterior CNN obtenemos que en este caso la exactitud final es similar a la del experimento 3, tal y como aparece en la Figura **5-1** con la etiqueta expC4. Se puede apreciar que el hecho de normalizar el número de imágenes no mejora el resultado, sino que lo empeora ligeramente. Para concluir esta primera sección de experimentos se puede concluir que los mejores resultados para clasificación empleando redes neuronales desde cero o haciendo uso del transfer learning los hemos obtenido en el tercer experimento (expC3), en el que se realiza un transfer learning de Alexnet y se modifica la capa de entrada, pero estos resultados se pueden mejorar empleando otras técnicas como pueden ser el Data Augmentation y/o la optimización de hiperparámetros.

A continuación, se resumen los experimentos realizados para que queden claras las diferencias entre ellos:

- Experimento 1 (expC1): CNN obtenida a partir de la adaptación de la arquitectura Alexnet, adaptación de la capa de entrada al tamaño original de las imágenes y entrenamiento desde cero con el conjunto de entrenamiento 1.
- Experimento 2 (expC2): CNN obtenida a partir de la adaptación de la arquitectura Alexnet, transfer learning, reducción de la resolución de las imágenes de entrada y reentrenamiento con el conjunto de entrenamiento 1.

- Experimento 3 (expC3): CNN obtenida a partir de la adaptación de la arquitectura Alexnet, transfer learning, adaptación de la capa de entrada al tamaño original de las imágenes y reentrenamiento con el conjunto de entrenamiento 1.
- Experimento 4 (expC4): Normalización del número de imágenes de entrenamiento por estancia. CNN obtenida a partir de la adaptación de la arquitectura Alexnet, transfer learning, adaptación de la capa de entrada al tamaño original de las imágenes y reentrenamiento con el conjunto de entrenamiento 2.

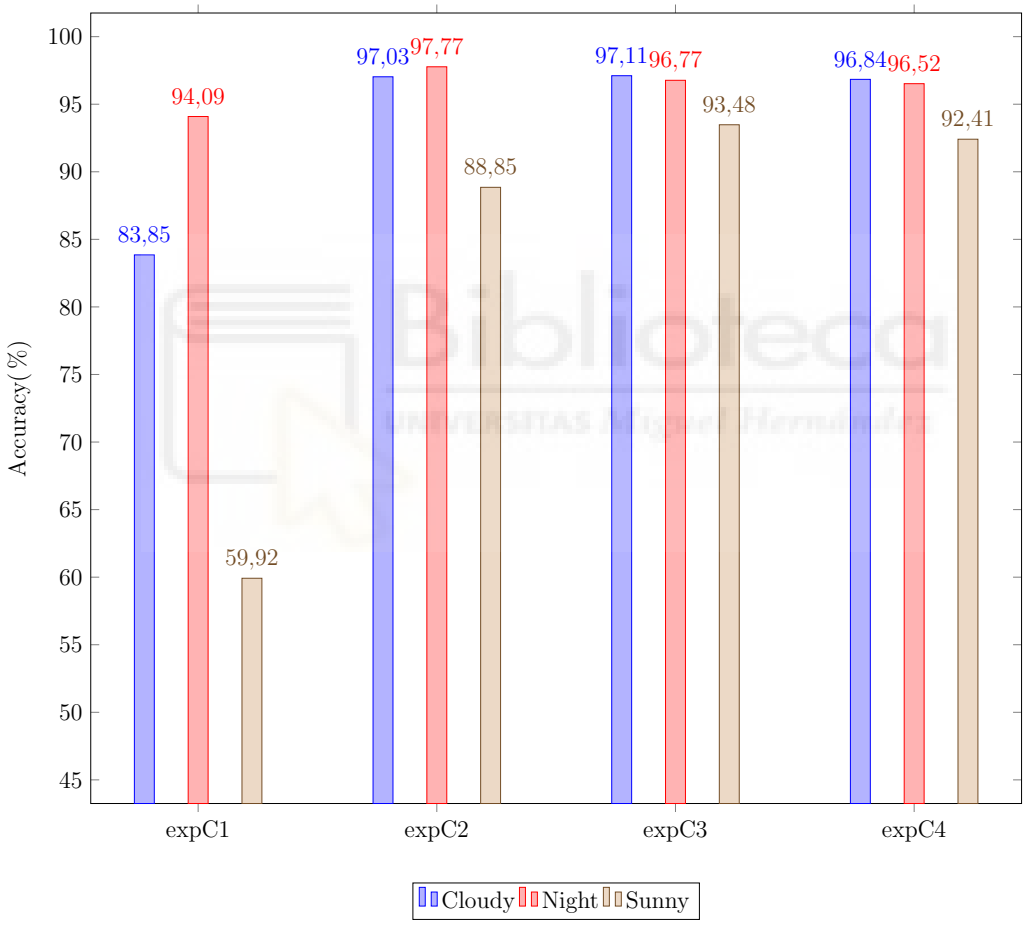


Figura 5-1: Exactitud de los experimentos inciales para imágenes test.

5.2.2. Experimentos con data augmentation

Realizando el Data Augmentation de las imágenes de entrenamiento se puede conseguir una red neuronal convolucional más robusta y fiable debido al mayor número de imágenes de entrenamiento. Esto tiene una repercusión directa en el coste computacional por lo que es interesante encontrar un equilibrio entre exactitud y tiempo de computación. De aqui en adelante se va a utilizar la configuración que mejores resultados ha dado hasta el momento, correspondiente al experimento 3 anterior y que consiste en Alexnet modificando la capa de entrada para aprovechar todo los píxeles de las imágnenes. Con el fin de no realizar un mayor entrenamiento del necesario se va a evaluar la CNN en cada epoch para encontrar la de mayor exactitud. Cada epoch consta de 1334 iteraciones y evaluaremos un total de 3 epochs. Además, el *Initial Learn Rate* toma un valor de 0.001. Como se puede apreciar en la (Figura **5-3**), en la segunda epoch obtenemos los mejores resultados. Aunque en realidad, si observamos el proceso de entrenamiento (Figura **5-2**), cuando finaliza la primera epoch la exactitud ya tiene un valor prácticamente del 100 %. En la epoch 3 la red neuronal ha entrenado más pero los resultados de testeo son peores, esto se debe a que la CNN ha comenzado a aprender las imágenes muy al detalle, tanto que ya no es efectivo. Este fenómeno se conoce como sobreajuste debido a un sobreentrenamiento del modelo.

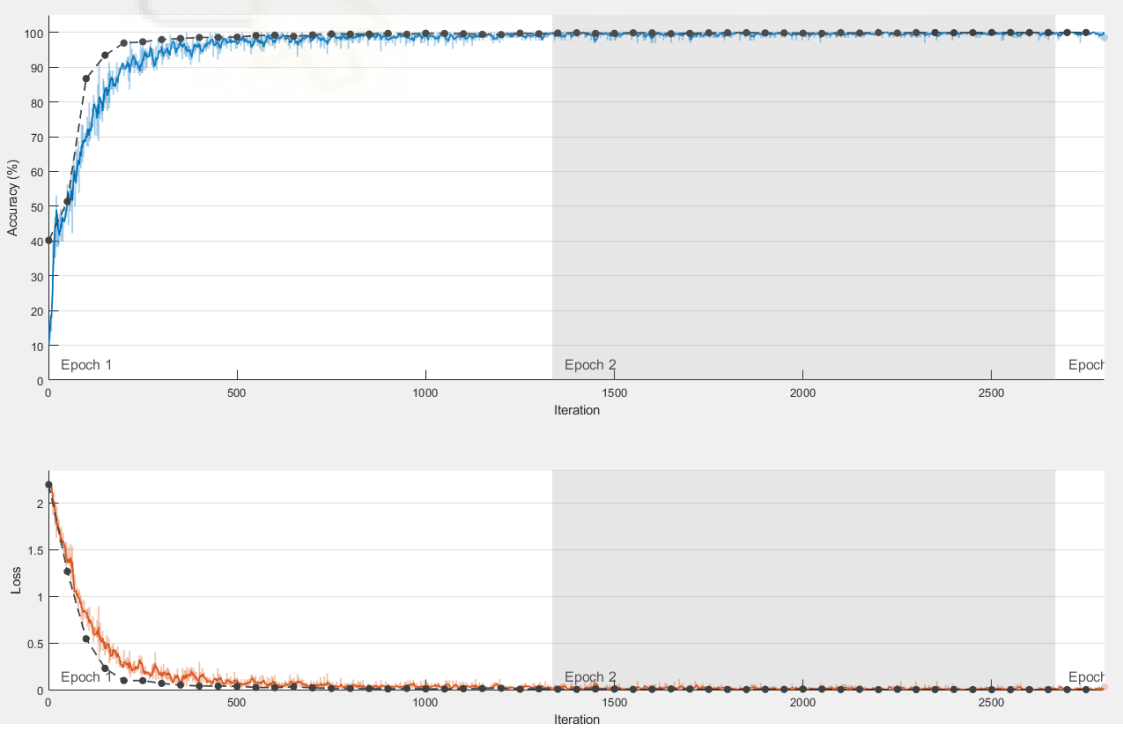


Figura 5-2: Entrenamiento para clasificación (expC5). En el gráfico superior se muestra la evolución de la exactitud durante el entrenamiento de la CNN y en el gráfico inferior la función de pérdida.

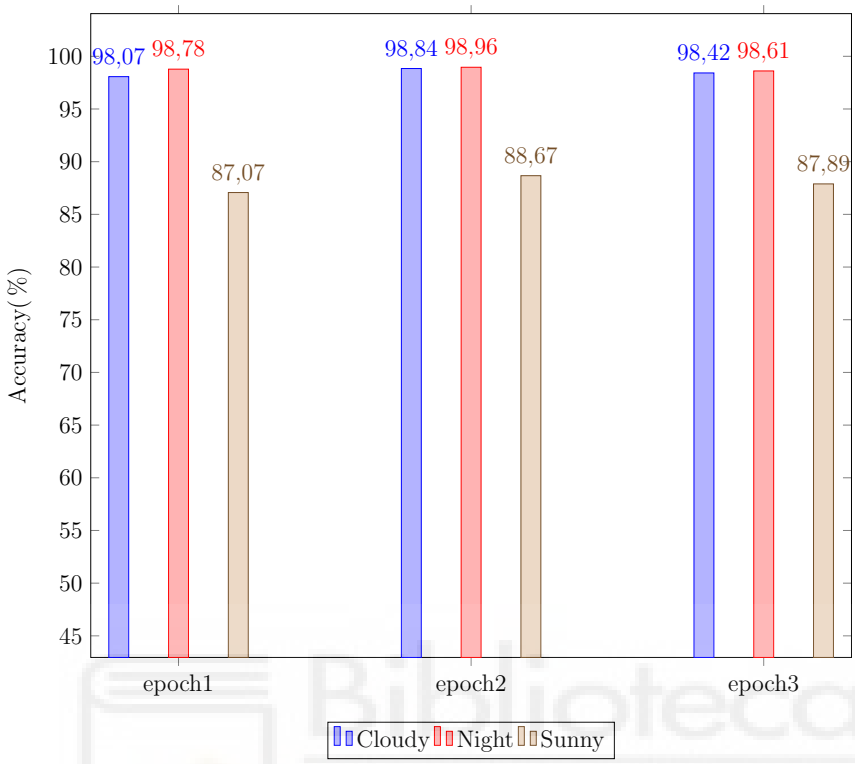


Figura 5-3: Exactitud testeo para imágenes con Data Augmentation (expC5).

Si comparamos las Figuras **5-1** y **5-3** se puede apreciar el buen rendimiento del Data Augmentation para nublado y noche. En cambio, los resultados para soleado son peores con el Data Augmentation. Además, si partimos del dato de que para alcanzar la segunda epoch se necesitaron unos 1372 minutos, el coste computacional del Data Augmentation es 98 veces superior a si entrenamos la red neuronal a partir de las imágenes originales, esto se debe a que antes la red debía procesar 556 imágenes y ahora 213.504 imágenes.

5.2.3. Optimización bayesiana

Aunque los resultados para clasificación ya son suficientemente buenos, aún se pueden mejorar un poco más con la selección idónea de los hiperparámetros, para ello se han realizado los experimentos mostrados en la (Tabla **5-2**).

En el experimento 6 (expC6) se parte del dataset de imágenes de entrenamiento con condiciones de iluminación ambiente nublado y se realiza la optimización bayesiana. En cada iteración del algoritmo se explora un punto del espacio tetradimensional formado por el InitialLearnRate, el Momentum, el L2Regularization y el valor estimado de la función objetivo. Posteriormente, en el experimento 7 (expC7) se parte de los hiperparámetros óptimos obtenidos en el experimento 6 y se realiza el entrenamiento de la red neurnonal con las imágenes del Data Augmentation. En el experimento 8 (expC8) se realiza una optimización del hiperparámetro Momentum teniendo el resto de hiperparámtros con el valor fijo, establecido por defecto en el *Training Options* de *Matlab*. El modelo de la función objetivo del experimento 8 se puede apreciar en la (Figura **5-4**). En el experimento 9 (expC9) las variables a optimizar son el Momentum y el InitialLearnRate, y su función objetivo viene dada por la (Figura **5-5**). Por último, en el experimento 10 (expC10) se ha aumentado el número de puntos explorados y se han optimizado el Momentum y el L2Regularization (Figura **5-6**). La tabla **5-2** resume los diferentes experimentos realizados en esta sección. Los resultados para las imágenes test de los anteriores experimentos los podemos visualizar en el diagrama de barras de la Figura **5-8**.

Exp	Tiempo (min)	Puntos explorados	Dataset entrenamiento	Hiperparámetros	Rango de valores	Valor óptimo
6	631	30	Dataset original	InitialLearnRate	[1e-4 1]	0.0060208
				Momentum	[0.5 1]	0.53961
				L2Regularization	[1e-10 1e-2]	3.8728e-9
7	1375	-	Dataset aumentado	InitialLearnRate	0.0060208	-
				Momentum	0.53961	-
				L2Regularization	3.8728e-9	-
8	2345	8	Dataset aumentado	InitialLearnRate	1e-3	1e-3
				Momentum	[0 1]	0.91105
				L2Regularization	1e-4	1e-4
9	2349	8	Dataset aumentado	InitialLearnRate	[1e-5 1e-2]	0.0072652
				Momentum	[0 1]	0.38491
				L2Regularization	1e-4	1e-4
10	8863	30	Dataset aumentado	InitialLearnRate	1e-3	1e-3
				Momentum	[0 1]	0.97983
				L2Regularization	[1e-5 1e-3]	0.00030623

Tabla 5-2: Experimentos optimización bayesiana para clasificación.

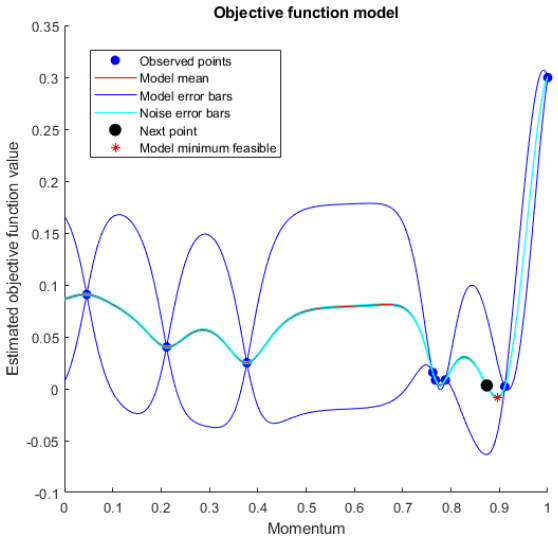


Figura 5-4: Grafica del modelo de la función objetivo del expC8, la cual tiene como propósito minimizar la función de pérdidad. En el eje X se representan los valores del hiperparámetro Momentum, y en el eje Y el valor de la función objetivo correspondiente.

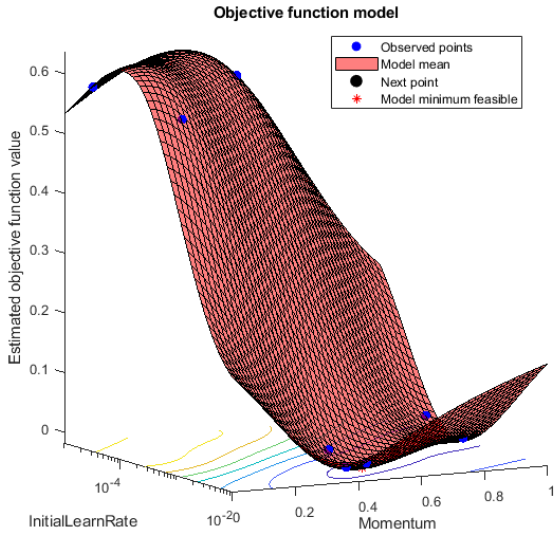


Figura 5-5: Grafica del modelo de la función objetivo del expC9, la cual tiene como propósito minimizar la función de pérdidad. En el eje X y Z se representan los valores de los hiperparámetros Momentum e Initial Learn Rate, y en el eje Y el valor de la función objetivo correspondiente.

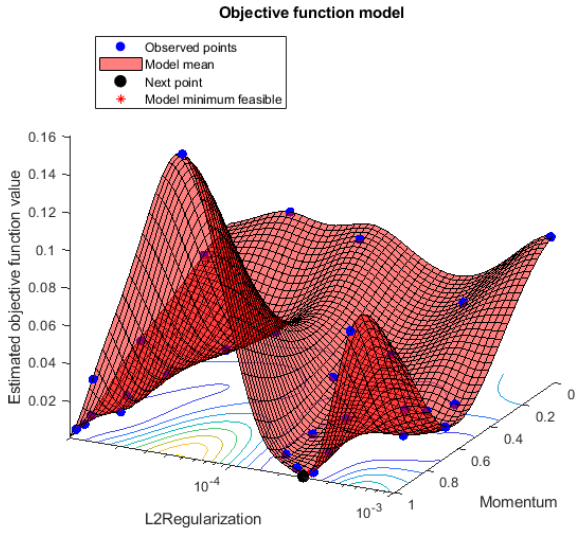


Figura 5-6: Grafica del modelo de la función objetivo del expC10, la cual tiene como propósito minimizar la función de pérdidad. En el eje X y Z se representan los valores de los hiperparámetros Momentum y L2 Regularization, y en el eje Y el valor de la función objetivo correspondiente.

Tal y como se puede apreciar en la Figura **5-8**, el mejor resultado para soleado lo obtenemos en el experimento 6 (expC6) que es el único que no ha empleado las imágenes del Data Augmentation, en realidad no es un hecho aislado ya que como se vió en la (Figura **5-3**) se obtenian peores resultados para soleado con este dataset. Para noche y nublado los mejores resultados se encuentran en el experimento 7 (expC7), aunque si tenemos en cuenta las tres condiciones de iluminación el que mejores resultados presenta es el experimento 8 (expC8).

Si representamos los valores medios de la exactitud de testeo de cada uno de los experimentos anteriores, considerando conjuntamente las tres condiciones de iluminación durante el test (Figura **5-8**), se puede concluir que el que mejores resultados presenta es el experimento 8 (expC8) con una exactitud promediada de 96.3 % que se obtiene de hacer la media de los resultados de las tres condiciones de iluminación. Por otro lado, es importante no solo tener en cuenta la exactitud media, sino también el tiempo de computación de manera que si comparamos la exactitud obtenida respecto al tiempo de entrenamiento (Figura **5-9**) obtenemos que el experimento 2 (expC2) es el que guarda la mejor relación. Esto se debe a que la exactitud del experimento 2 es notablemente buena (94.55 %) pero su tiempo de computación es extremadamente bajo, tan solo 3 minutos y medio.

De este modo, en función de cuál sea nuestro objetivo será más interesante realizar un experimento u otro, si deseamos una exactitud muy alta aceptando un tiempo de computación elevado, el empleo del data augmentation y de la optimización bayesiana es más que válido. En cambio, si el tiempo de entrenamiento del modelo juega un papel fundamental en la elección del algoritmo a emplear, el transfer learning de Alexnet sin modificar el tamaño de píxeles de la capa de entrada es la opción más idónea.

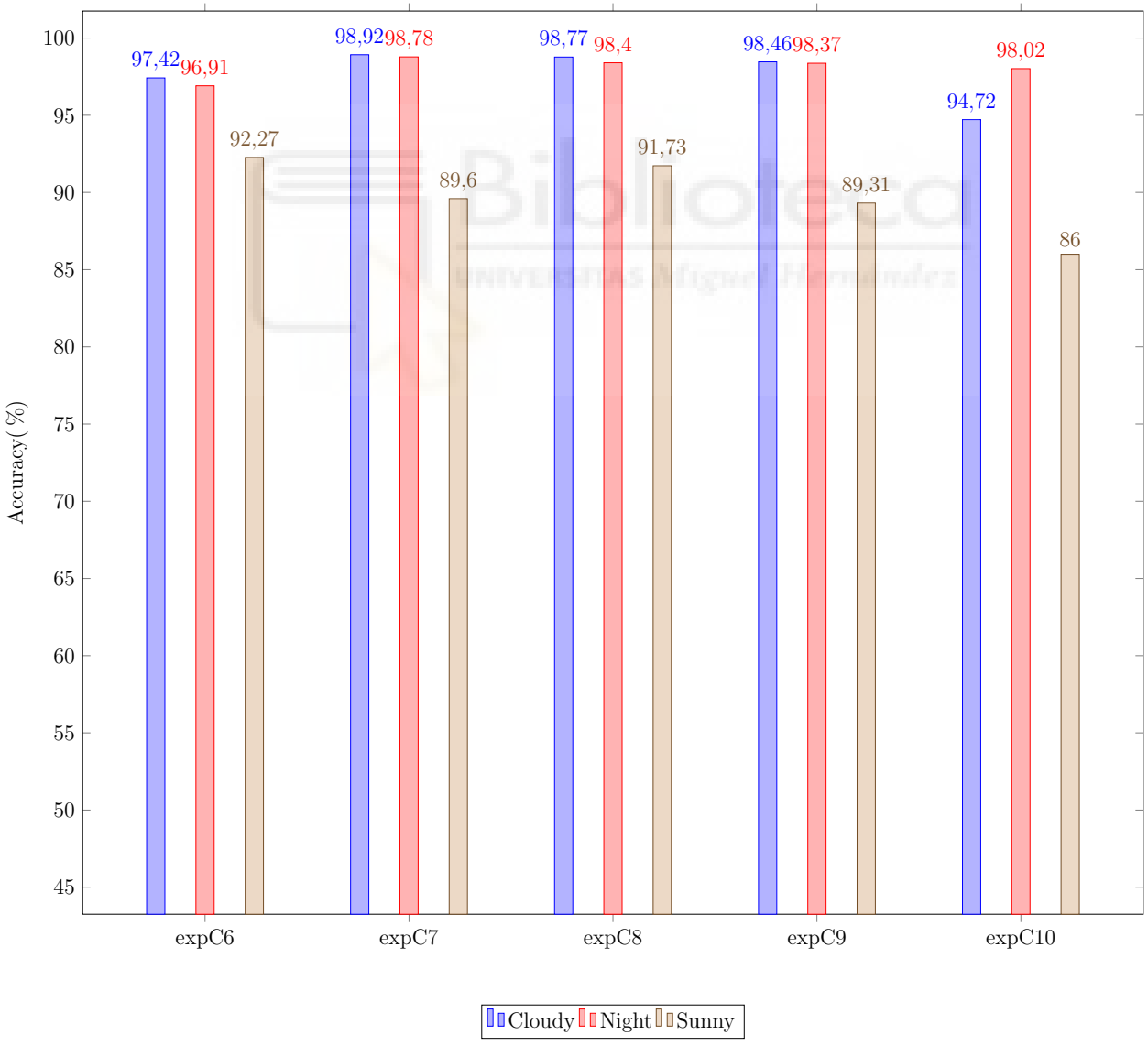


Figura 5-7: Exactitud de los experimentos para imágenes test.

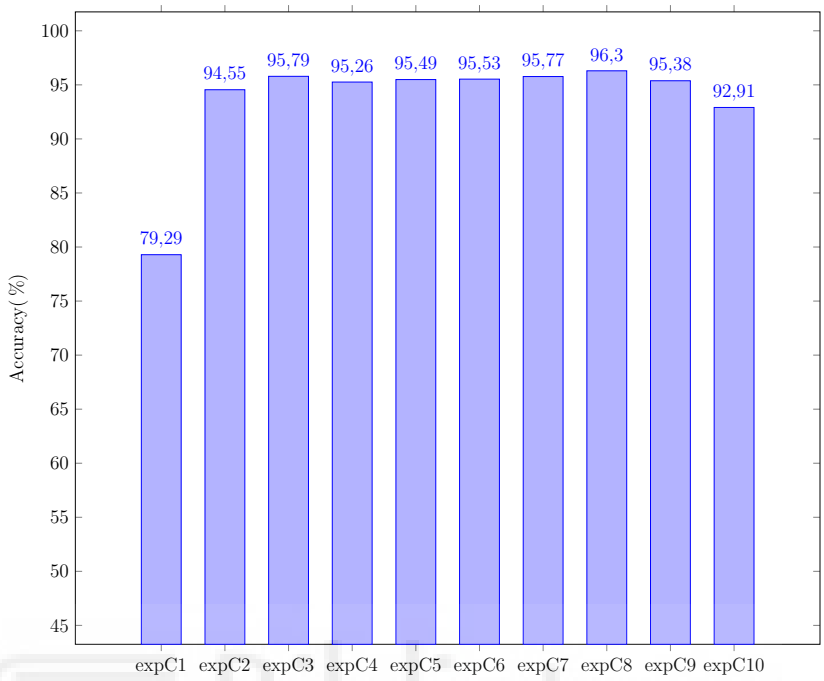


Figura 5-8: Exactitud media de los experimentos de clasificación para imágenes test.

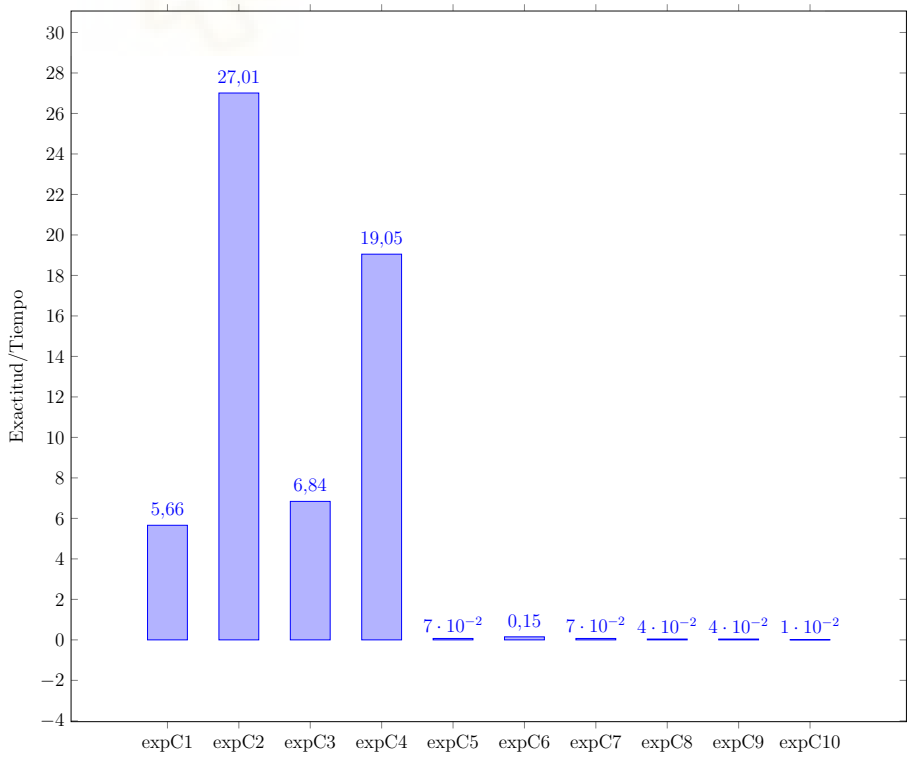


Figura 5-9: Exactitud/tiempo de los experimentos de clasificación para imágenes test.

Una herramienta muy empleada en el campo del machine learning para evaluar y analizar los resultados de la clasificación son las matrices de confusión. Las matrices de confusión permiten dar una idea de cómo está clasificando nuestro algoritmo, representando los aciertos y errores en la predicción de cada categoría. Las matrices de confusión tienen un orden igual al número de categorías, de manera que en las filas se disponen las clases reales y en las columnas las predichas. Las matrices de confusión para el experimento 8 (expC8) las encontramos en las (Figuras **5-10**, **5-11**, **5-12**).

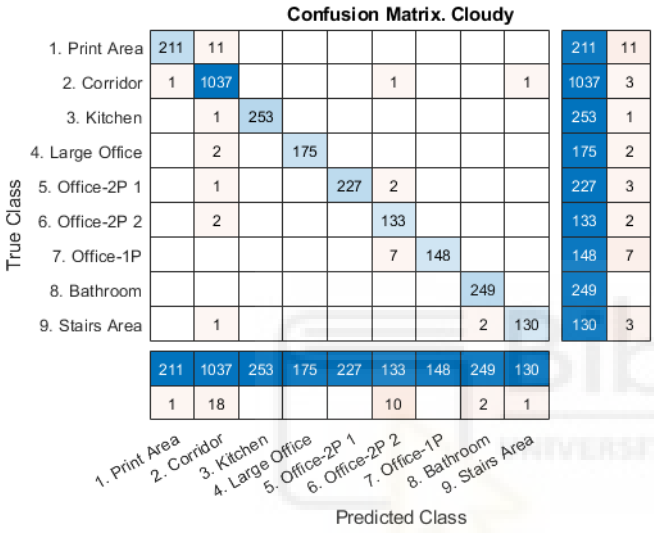


Figura 5-10: Matriz de confusión del expC8 para condiciones de nublado.

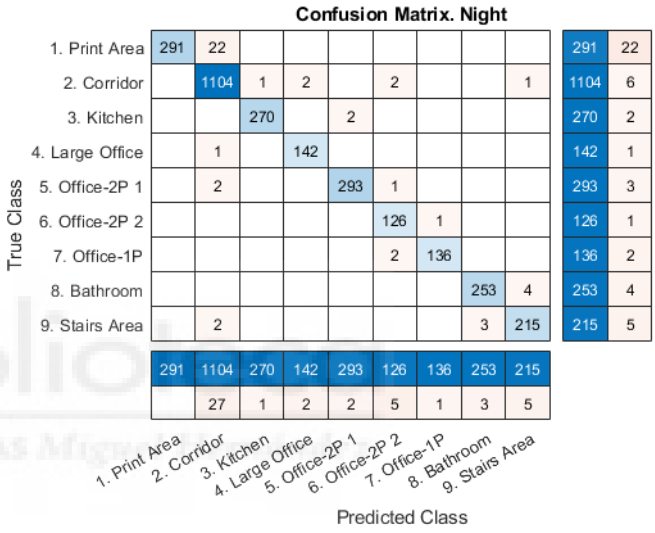


Figura 5-11: Matriz de confusión del expC8 para condiciones de noche.

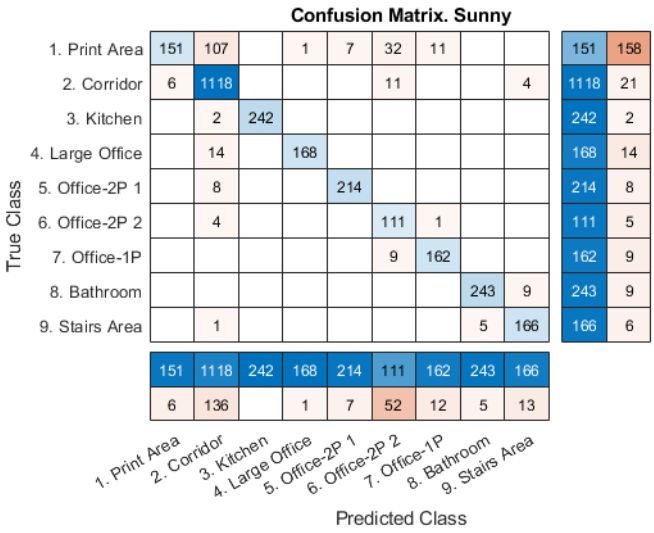


Figura 5-12: Matriz de confusión del expC8 para condiciones de soleado.

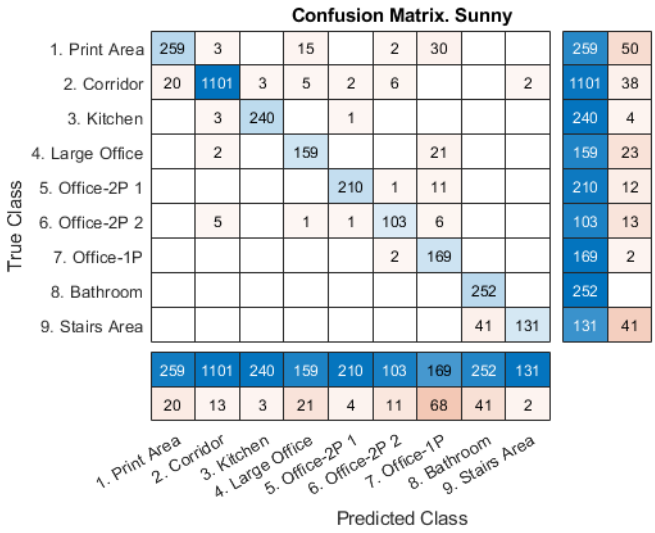


Figura 5-13: Matriz de confusión del expC2 para condiciones de soleado.

La matriz de confusión para día soleado (Figura **5-12**) indica que con el data augmentation la estancia que más lleva a error es la primera "Printer Area", siendo confundida casi la mitad de las veces por la segunda estancia "Corridor". Si comparamos los resultados con la matriz de confusión de día soleado del experimento 2 (Figura **5-13**), los errores de confusión a partir del data augmentation son menores en la mayoría de estancias por lo que el problema no debe de estar en que los efectos del data augmentation funcionen mal en general con las imágenes en un día soleado. Si analizamos la disposición de las estancias en la (Figura **5-14**) se puede apreciar que la primera estancia es la única que tiene ventanas en orientación norte relativa al plano. Si se visualizan las imágenes correspondientes a esa estancia se observa que los rayos de sol atraviesan esas ventanas de manera oblicua debido a que está amaneciendo. Esto junto con los efectos del data augmentation, provocan la confusión en el algoritmo.

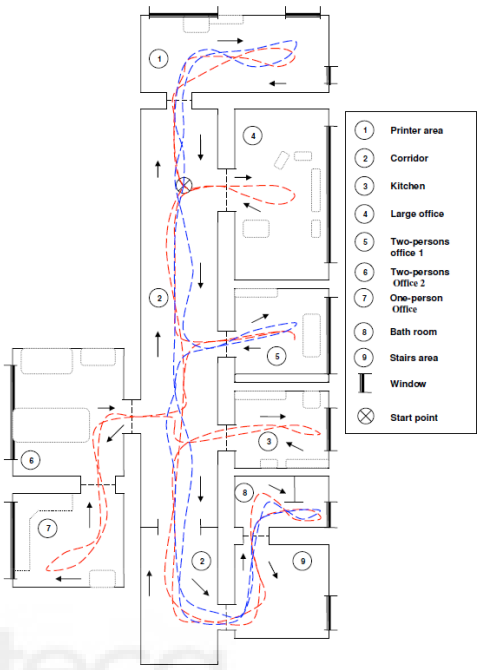


Figura 5-14: Planta del dataset utilizado (Friburgo).

Por otro lado, en todas las matrices de confusión anteriores se puede apreciar que el pasillo es confundido en todas las predicciones de estancias salvo en dos, la estancia número 7 "One-person Office", y la octava "Bath room". Estas estancias son las únicas que no lindan con el pasillo, pero en el resto existe una dificultad a la hora de establecer el límite exacto entre ellas.

5.3. CNN para regresión

La tarea de regresión del presente trabajo tiene por objetivo obtener la localización del robot de manera más precisa. Para abordar este problema se parte de la CNN del experimento tres para clasificación, mediante transfer learning se crea una CNN para regresión y se realiza el entrenamiento (Figura **5-15**). Se emplea como datos de entrenamiento el conjunto de entrenamiento 1 que se presenta al principio del capítulo (Tabla 5.1) y que cuenta con 556 imágenes para condiciones de nublado y las etiquetas son las coordenadas (x, y) del punto de captura de cada imagen, dadas por el ground truth del dataset. Por lo que la salida esperada ya no es una estancia, sino dos valores numéricos correspondientes a la coordenadas x e y del plano. Recordemos que del total de imágenes de entrenamiento el 20 % se emplea para validación del mismo.

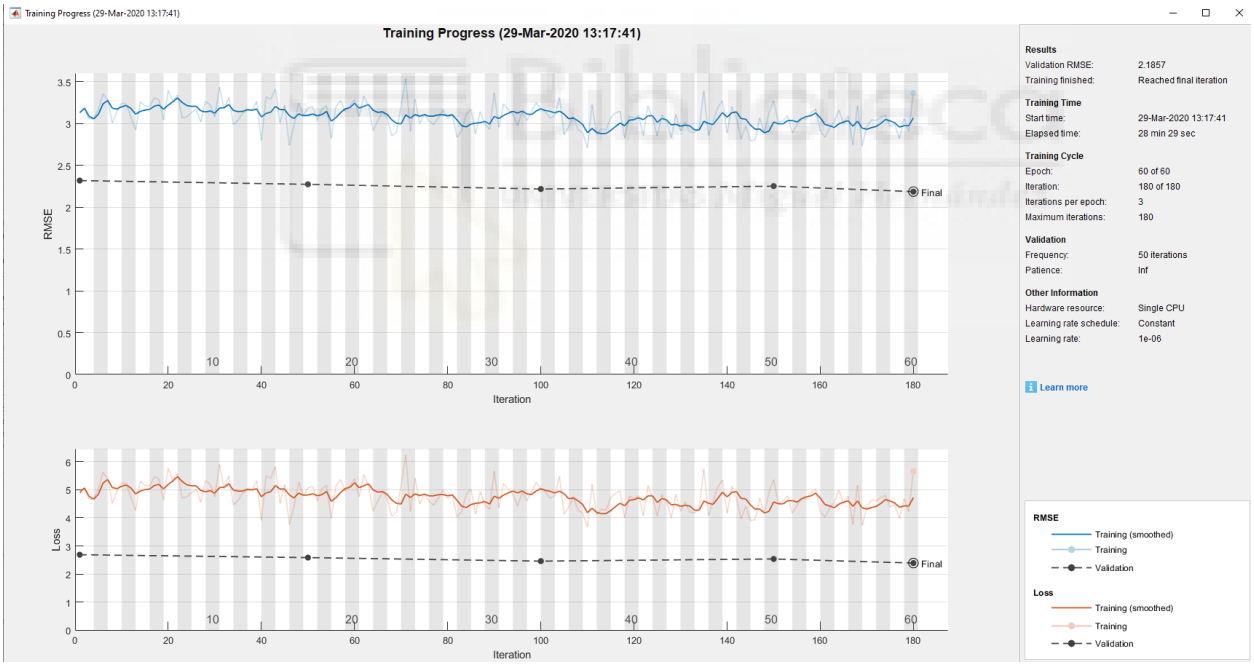


Figura 5-15: Entrenamiento para regresión. En la figura superior se representa el Error Cuadrático Medio (RMSE) durante la evolución del entrenamieto, y en el gráfico inferior se aprecia la evolución de la función de péridida.

Este entrenamiento tuvo un tiempo de computación de unos 29 minutos obteniendo un error de posición medio de 1,797 metros. En la figura **5-15** se aprecia como se producen oscilaciones sin reducir el error, esto se debe a una mala configuración de los hiperparámetros de entrenamiento por lo que debemos optimizarlos mediante la optimización bayesiana.

5.3.1. Optimización bayesiana

Se han realizado 10 entrenamientos empleando diferentes algoritmos de resolución (sgdm, rmsprop y adam), partiendo de diferentes redes neuronales (Alexnet con sus pesos, modificando la capa inicial y las finales o la CNN del experimento 3 de clasificación (expC3)) y se han optimizado los hiperparámetros pertenecientes a cada algoritmo de resolución. En todos los experimentos se han realizado entrenamientos de 60 epochs salvo en el experimento 9 (expR9) que se entrenó con 100 epochs y es por ello que su tiempo de computación es mayor al resto. En el experimento 4 (expR4), se emplea un data augmentation propio de matlab que consiste en reflexiones tanto en el eje x como en el y, rotaciones de 0 a 360 grados y traslaciones de píxeles en el eje x e y en un rango de 20x20 píxeles. Estos efectos se hacen sobre las imágenes del conjunto de entrenamiento 1 antes de ser pasadas a la red para el entrenamiento. La tabla **5-3** presenta la información referente a los entrenamientos llevados a cabo.

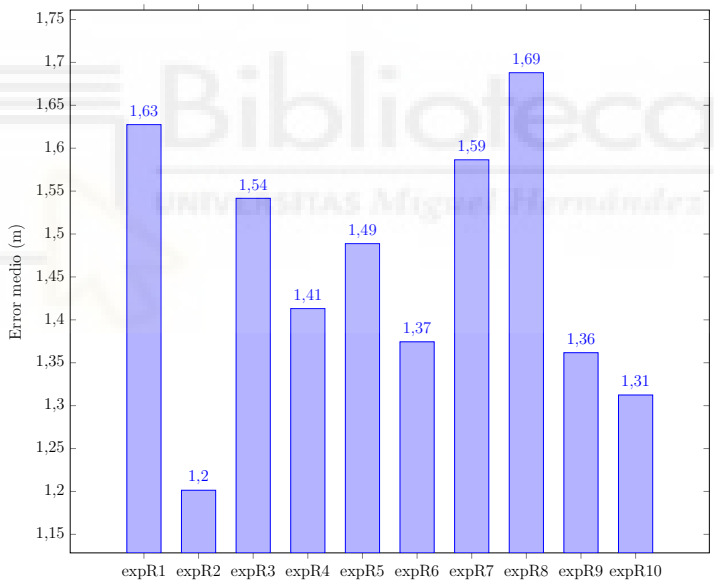


Figura 5-16: Error medio de los experimentos de regresión en test para noche.

En los valores de los hiperparámetros óptimos no se puede establecer un patrón claro ya que estos parámetros dependen unos de otros en formas desconocidas. En la figura **5-16** tenemos el error medio en metros de cada uno de los experimentos evaluados con imágenes test en condiciones nocturnas. El mejor valor lo encontramos en el segundo experimento reduciendo así en 60 centímetros el error medio obtenido sin el empleo de la optimización. Aunque hemos mejorado los resultados, estos no son lo suficientemente prometedores para poder llevar a cabo la tarea de localización visual, teniendo en cuenta que la distancia media entre imágenes capturadas es de 40 cm. Es por ello que en el siguiente apartado se propone el uso de descriptores de apariencia global o descriptores holísticos para llevar a cabo una localización más precisa del robot.

Exp	Tiempo (min)	CNN de partida	SolverName	Hiperparámetros	Rango de valores	Valor óptimo
1	586	Alexnet	sgdm	InitialLearnRate	[1e-7 1]	0,00020467
				Momentum	[0 1]	0,87
				L2Regularization	[1e-10 1e-2]	1,1e-10
2	578	CNN exp 3 clasificación	sgdm	InitialLearnRate	[1e-7 1]	0,000036909
				Momentum	[0 1]	0,97093
				L2Regularization	[1e-10 1e-2]	2,57e-10
3	590	CNN exp 3 clasificación	adam	InitialLearnRate	[1e-7 1]	9,29e-6
				SquaredGradientDecayFactor	[0.9 1]	0,97544
				GradientDecayFactor	[0.9 1]	0,92312
				L2Regularization	[1e-10 1e-2]	2,3609e-10
4	581	CNN exp 3 clasificación	sgdm + DA matlab	InitialLearnRate	[1e-7 1]	1,7401e-6
				Momentum	[0 1]	0,1835
				L2Regularization	[1e-10 1e-2]	1,2384e-10
5	588	CNN exp 3 clasificación	rmsprop	InitialLearnRate	[1e-7 1]	1,0481e-5
				SquaredGradientDecayFactor	[0.5 1]	0,50129
				L2Regularization	[1e-10 1e-2]	3,3116e-3
6	600	CNN exp 3 clasificación	adam	InitialLearnRate	[1e-7 1]	1,6e-5
				SquaredGradientDecayFactor	[0 1]	0,018458
				GradientDecayFactor	[0 1]	7,436e-2
				L2Regularization	[1e-10 1e-2]	3,6304e-9
				Epsilon	[1e-12 1e-2]	2,869e-4
7	600	CNN exp 3 clasificación	rmsprop	InitialLearnRate	[1e-12 1]	1,3158e-5
				SquaredGradientDecayFactor	[0 1]	0,9224
				Epsilon	[1e-12 1]	3,4677e-11
				L2Regularization	[1e-12 1]	1,7268e-3
8	590	CNN exp 3 clasificación	sgdm	InitialLearnRate	[1e-7 1]	7,0942e-5
				Momentum	[0 1]	0,032441
				L2Regularization	[1e-10 1e-2]	1,4819e-8
				LearnRateDropPeriod	[0 80]	36,906
				LearnRateDropFactor	[0 1]	0,95738
9	844	CNN exp 3 clasificación	rmsprop	InitialLearnRate	[1e-7 1e-4]	9,4319E-05
				Momentum	[0.8 1]	0,92828
				LearnRateDropFactor	[0 1]	0,2522
				L2Regularization	[1e-12 1e-8]	1,0568-11
10	598	CNN exp 3 clasificación	sgdm	InitialLearnRate	0,000036909	0,000036909
				Momentum	[0.9 1]	0,96455
				L2Regularization	2,57e-10	2,57e-10

Tabla 5-3: Experimentos optimización bayesiana para regresión.

5.4. Localización mediante descriptores holísticos

En esta última sección se ha empleado la técnica de localización jerárquica en la que se utiliza una red neuronal convolucional para llevar a cabo una localización gruesa mediante la clasificación en estancias y posteriormente una localización más fina del robot mediante la técnica del vecino más cercano. Los descriptores holísticos o descriptores de apariencia global son extraídos de las capas de la red neuronal, estos vectores pueden proceder de diferentes capas y es posible emplear una red distinta a la utilizada para clasificación para la extracción de los mismos.

ExpD	CNN empleada para clasificación	CNN empleada para la extracción	Capa de extracción del vector descriptor
1	Red experimento 3 clasificación	Red experimento 3 clasificación	fc6
2	Red experimento 3 clasificación	Red experimento 3 clasificación	fc7
3	Red experimento 5 clasificación	Red experimento 5 clasificación	fc6
4	Red experimento 6 clasificación	Red experimento 6 clasificación	fc6
5	Red experimento 7 clasificación	Red experimento 7 clasificación	fc6
6	Red experimento 8 clasificación	Red experimento 8 clasificación	fc6
7	Red experimento 9 clasificación	Red experimento 9 clasificación	fc6
8	Red experimento 10 clasificación	Red experimento 10 clasificación	fc6
9	Red experimento 3 clasificación	Alexnet	fc6

Tabla 5-4: Experimentos vectores descriptores.

En los dos primeros experimentos de la tabla **5-4** se comprueba a partir de qué capas se obtienen mejores resultados, entre las diferentes capas se ha decidido estudiar las totalmente conectadas 6 y 7. Se han obtenido unos mejores resultados a partir de la capa totalmente conectada 6 la cual es anterior a la fc7. Los vectores provenientes de la capa fc6 están más orientados a la extracción de patrones y geometrías de las imágenes mientras que los de la fc7 están más orientados a la clasificación de las estancias. En los experimentos se han empleado las mejores redes para clasificación, siendo la CNN del experimento 8 de clasificación (expC8) la mejor obtenida, lo cual no se ha traducido en el mínimo error de localización. Esto se debe

a que esta CNN no funcionaba tan bien en días soleados provocando un error de clasificación que se traduce en un aumento del error medio de la localización.

El mínimo error medio de localización teniendo en cuenta los resultados de soleado, nublado y noche se da en los experimentos 1 (expD1) y 9 (expD9) (Figura **5-17**), en los cuales se ha empleado la CNN del experimento 3 de clasificación (expC3) que era la segunda mejor. En el primer experimento (expD1) se utiliza la misma red para llevar a cabo la clasificación y la extracción de los vectores mientras que el experimento nueve se utiliza Alexnet solo para la extracción de los vectores.

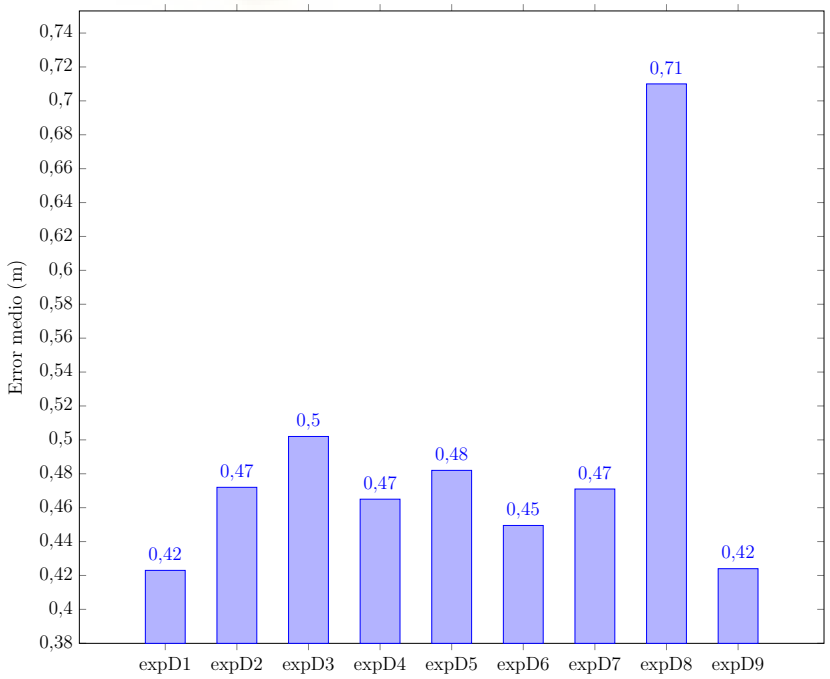


Figura 5-17: Error medio de los experimentos de localización jerárquica de las imágenes test.

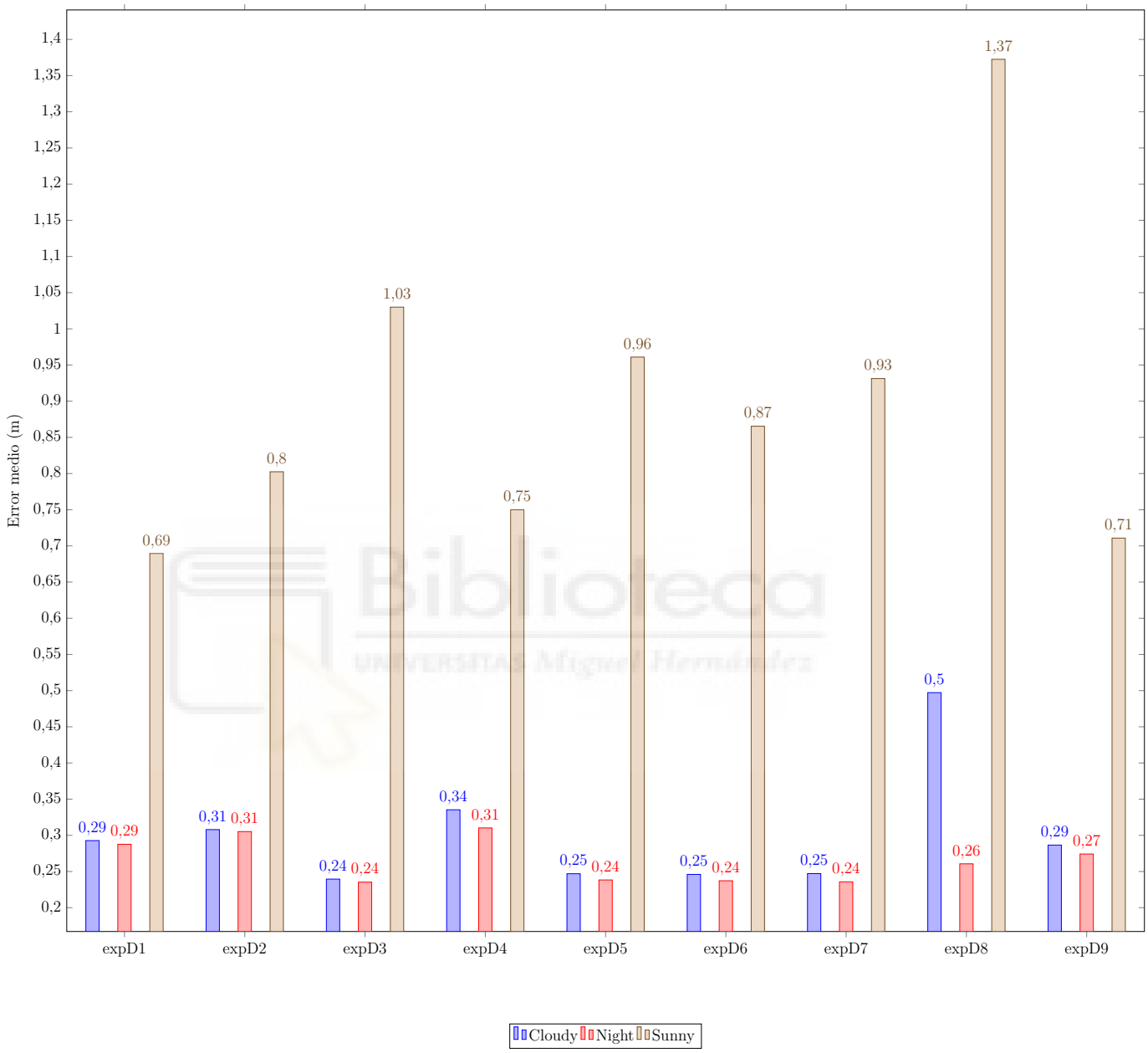


Figura 5-18: Error de posición para imágenes test empleando la localización jerárquica.

6 Conclusiones y líneas de trabajo futuras

En el presente trabajo se han realizado diferentes experimentos para abordar la tarea de localización, la cual, junto con el mapping, es una de la tareas fundamentales en robótica móvil autónoma. Para ello se ha hecho uso de la red neuronal convolucional conocida como Alexnet, ya sea aprovechando su arquitectura y/o su entrenamiento previo. En los distintos experimentos de clasificación, el empleo del tranfer learning de Alexnet ha dado resultados satisfactorios.

Con el objetivo de mejorar la precisión, se emplean técnicas como el data augmentation y la optimización bayesiana. El data augmentation incrementa la precisión de testeo para noche y nublado, pero no mejora sustancialmente los resultados para soleado. Por otro lado, el empleo de la optimización bayesiana ha conseguido mejorar los resultados pero el tiempo de ejecución de estos algoritmos es muy alto y el incremento de precisión que aportan es bajo, además los valores óptimos encontrados son difícilmente analizables ya que la forma en la que dependen los hiperparámetros unos de otros es desconocida. Aunque se podrían implementar otras técnicas de optimización como pueden ser la búsqueda aleatoria de hiperparámetros o la búsqueda por cuadrícula.

En términos generales, las condiciones de iluminación soleadas han mostrado ser las más desafiantes en este problema de localización. De este modo, resulta especialmente relevante analizar en detalle los porcentajes de acierto de la fase de test con este tipo de iluminación. La red que ha proporcionado los mejores resultados es la resultante del tercer experimento de clasificación (expC3), una red sencilla cuyo entrenamiento conllevó unos 14 minutos y se basa en la aplicación del transfer learning a Alexnet y la modificación de la capa de entrada. Una posible ampliación de este estudio sería analizar con mayor profundidad qué capas son las más idóneas para la extracción de los descriptores.

Desde otra perspectiva, abordar la tarea de localización mediante la regresión no ha aportado soluciones que puedan considerarse satisfactorias, obteniendo errores de localización de entorno a 1 metro. En cambio, la localización jerárquica mediante el empleo de descriptores de apariencia global ha permitido conseguir errores en torno a los 24 centímetros en condiciones de iluminación de nublado y noche, mientras que para soleado el mínimo error de

localización es de 69 centímetros.

Como líneas de investigación futuras relacionadas con la localización de robots autónomos móviles, se podría probar el uso de otras redes neuronales como puede ser el caso de GoogLeNet, NetVLAD o Inception v3 entre otras arquitecturas. Incluso se podría reducir el tamaño de redes neuronales con el fin de que los tiempos de computación sean menores tal y como ha demostrado Renda *et al.*, [42], eliminando aquellas partes de la red neuronal que no nos interesan y consiguiendo los mismos resultados con la red reducida.

Con el fin de obtener un Data Augmentation que funcione mejor en condiciones de iluminación soleada. Sería importante analizar, por separado, qué efectos son los más relevantes para obtener un buen porcentaje de acierto en esta aplicación concreta de localización, o incluso estudiar otro tipo de efectos que puedan ser más útiles para emular situaciones reales.

Un estudio interesante a desarrollar sería intentar mejorar la precisión en clasificación analizando en mayor profundidad en qué momento del entrenamiento una red neuronal empieza a aprender con demasiado detalle dando lugar a un sobreajuste, es decir, se está sobreentrenando la red neuronal produciendo un rendimiento inferior.

Una posible línea de investigación podría ser la de llevar a cabo la localización mediante Redes Neuronales Siamesas. Este tipo de redes se caracterizan por contar con dos subredes idénticas o gemelas, las cuales están unidas en sus salidas. Estas subredes además de tener la misma arquitectura comparten pesos. Cabe destacar que la red siamesa recibe un par de imágenes y una etiqueta, la cual será cero o uno en función de si las imágenes de entrada son iguales o diferentes respectivamente. Cada subred recibe una de las imágenes y generan un par de vectores los cuales son comparados con el objetivo de determinar si las imágenes de entrada son iguales o diferentes. Entre las aplicaciones más comunes de este tipo de redes encontramos el reconocimiento facial (Cui *et al.*, [12]), aunque se puede extrapolar a la localización de robots móviles (Leyva-Vallina *et al.*, [27]).

Nuestro planteamiento utilizando Redes Siamesas sería el siguiente. Se entrenaría la red con el fin de que pudiera predecir con una alta precisión si el par de imágenes de entrada pertenecen a la misma estancia (etiqueta con un cero) o a diferentes estancias (etiqueta con un uno). Si esta tarea se realizase de forma exitosa el objetivo sería llevar a cabo una localización fina. De manera que una de las imágenes de entrada de la red se correspondería con una imagen de entrenamiento, cuyas coordenadas serían conocidas. Y por otro lado tendriamos la imagen test, que sería comparada con todas las imágenes de entrenamiento. La imagen de entrenamiento más parecida a la imagen test, es decir, la imagen cuya salida de la red diera un valor más cercano a cero según el etiquetado establecido, nos daría unas coordenadas y por tanto una localización aproximada de la imagen test que se esté evaluando.

En esta misma línea se podría estudiar la implementación de Tripletas, que se basarían en el mismo principio que el método anterior pero en vez de dos subredes tendríamos tres. Fierro *et al.*, [19] demuestran que las arquitecturas múltiples solucionan los tres retos más importantes de los sistemas de recuperación de imágenes, como lo son la brecha semántica, el aprendizaje de similitud y el espacio de almacenamiento, los cuales no habían sido resueltos en trabajos anteriores.

Finalmente, otros posibles trabajos serían la ampliación del uso de las técnicas de aprendizaje profundo para la localización mediante el uso de diferentes herramientas como los autoencoders o las redes LSTM. Por último, también nos gustaría crear y evaluar enfoques de localización basados en CNNs en entornos exteriores.

Bibliografía

[1] M.H. Abadi, M.A. Oskei, and A. Fakharian. Mobile robot navigation using sonar vision algorithm applied to omnidirectional vision. *AI Robotics (IRANOPEN), IEEE*, pages 1–6, 2015.

[2] E. Alessandri, A. Gasparetto, and R. Valencia Garcia. An application of artificial intelligence to medical robotics. *Journal of Intelligent and Robotic Systems*, 41:225–243, 2005.

[3] K. Amer, R. Samy, M. ElHakim, M. Shaker, and M. ElHelw. Convolutional neural network-based deep urban signatures with application to drone localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[4] H. Andreasson, A. Treptow, and T. Duckett. Localization for mobile robots using panoramic vision, local features and particle filter. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3348–3353, 2005.

[5] Y. Berenguer, L. Payá, M. Ballesta, and O. Reinoso. Position estimation and local mapping using omnidirectional images and global appearance descriptors. *Journal of Sensors*, 15(10):26368, 2015.

[6] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

[7] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. 2013.

[8] S. Cebollada, L. Payá, M. Flores, V. Román, A. Peidró, and O. Reinoso. A deep learning tool to solve localization in mobile autonomous robotics. *ICINCO 2020: 17th Intl. Conf. On Informatics in Control, Automation and Robotics*, 2020.

[9] S. Cebollada, L. Payá, V. Román, and O. Reinoso. Hierarchical localization in topological models under varying illumination using holistic visual descriptors. *IEEE Access*, 7, 2019.

[10] O. Cespedes. Localización de un robot móvil utilizando información visual y redes neuronales convolucionales. *(Trabajo de Fin de Grado). Universidad Miguel Hernández de Elche, España.*, 2020.

[11] Y. Chen, R. Chen, M. Liu, A. Xiao, D. Wu, and S. Zhao. Indoor visual positioning aided by cnn-based image retrieval: Training-free, 3d modeling-free. *Sensors*, 18(8), 2018.

[12] W. Cui, W. Zhan, J. Yu, C. Sun, and Y. Zhang. Face recognition via convolutional neural networks and siamese neural networks. In *2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, pages 746–750, 2019.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1(Jun):886–893, 2005.

[14] J. Ding, B. Chen, H. Liu, and M. Huang. Convolutional neural network with data augmentation for sar target recognition. *IEEE Geoscience and remote sensing letters*, 13(3):364–368, 2016.

[15] T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[16] S. Falkner, A. Klein, and F. Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. *arXiv preprint arXiv:1807.01774*, 2018.

[17] H.J.S. Feder, J.J. Leonard, and C.M. Smith. Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18(7):650–668, 1999.

[18] M. Feurer and F. Hutter. Hyperparameter optimization. In *Automated Machine Learning*, pages 3–33. Springer, 2019.

[19] A.N. Fierro, M. Nakano, K. Yanai, and H.M. Pérez. Redes convolucionales siamesas y tripletas para la recuperación de imágenes similares en contenido. *Información tecnológica*, 30:243 – 254, 12 2019.

[20] G. Gallegos and P. Rives. Indoor slam based on composite sensor mixing laser scans and omnidirectional images. *IEEE International Conference on Robotics and Automation*, 2010.

[21] S. Gatesichapakorn, J. Takamatsu, and M. Ruchanurucks. Ros based autonomous mobile robot navigation using 2d lidar and rgb-d camera. In *2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP)*, pages 151–154, 2019.

[22] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. *IEEE International Conference on Robotics and Automation*, 2003.

[23] N. Kannathal, U. R. Acharya, C. M. Lim, P. Sadasivan, and S. Krishnan. Classification of cardiac patient states using artificial neural networks. *Robotics and Autonomous Systems*, 8(4):206–211, 2003.

[24] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown. Auto-weka: Automatic model selection and hyperparameter optimization in. *Automated Machine Learning: Methods, Systems, Challenges*, page 81, 2019.

[25] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis. A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications. *IEEE Internet of Things Journal*, 5(2):829–846, 2018.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(Nov):2278–2324, 1998.

[27] M. Leyva-Vallina, N. Strisciuglio, and N. Petkov. Place recognition in gardens by learning visual representations: Data set and benchmark analysis. In Mario Vento and Gennaro Percannella, editors, *Computer Analysis of Images and Patterns*, pages 324–335. Springer International Publishing, 2019.

[28] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, page 1150, USA, 1999. IEEE Computer Society.

[29] M. Mancini, S.R. Bulò, E. Ricci, and B. Caputo. Learning deep nbnn representations for robust place categorization. *IEEE Robotics and Automation Letters*, 2(3)(May):1149–1801, 2017.

[30] E. Martín, J.L. Lázaro, F.J. Meca, D. Salido, F. Espinosa, and L. Pallarés. Infrared sensor system for mobile-robot positioning in intelligent spaces. *Sensorial Systems Applied to Intelligent Spaces*, 11(5):5416–5438, 2011.

[31] E. Menegatti, A. Pretto, A. Scarpa, and E. Pagello. Random search for hyper-parameter optimization. *IEEE Transactions on Robotics*, 22(Jun):523–535, 2006.

[32] L. Moreno, J.M. Armingol, S. Garrido, A. de la Escalera, and M.A. Salichs. A genetic algorithm for mobile robot localization using ultrasonic sensors. *Journal of Intelligent and Robotic Systems*, 34:135–154, 2002.

[33] D. Murray and C. Jennings. Stereo vision based mapping and navigation for mobile robots. *Proceedings of International Conference on Robotics and Automation*, 1997.

[34] K. Ohno, T. Tsubouchi, B. Shigematsu, and S. Yuta. Differential gps and odometry-based outdoor navigation of a mobile robot. *Advanced Robotics*, 18:611–635, 2004.

[35] I. Ohya, A. Kosaka, and A. Kak. Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. *IEEE Transactions on Robotics and Automation*, 14(6)(Dec):969–978, 1998.

[36] L. Payá, F. Amorós, L. Fernández, and O. Reinoso. Performance of global-appearance descriptors in map building and localization using omnidirectional vision. *Sensors*, 14(2):3033–3064, 2014.

[37] L. Payá, L. Fernández, L. Gil, and O. Reinoso. Map building and monte carlo localization using global appearance of omnidirectional images. *Journal of Sensors*, 10(12):11468–11497, 2010.

[38] L. Payá, A. Peidró, F. Amorós, D. Valiente, and O. Reinoso. Modeling environments hierarchically with omnidirectional imaging and global-appearance descriptors. *Remote sensing*, 10(4):522, 2018.

[39] L. Payá and O. Reinoso. A state-of-the-art review on mapping and localization of mobile robots using omnidirectional vision sensors. *Journal of sensors*, 2017.

[40] L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[41] J. Radon. Uber die bestimmung von funktionen durch ihre integralwerte langs gewisser mannigfaltigkeiten. *Berichte Sachsische Akademie der Wissenschaften*, 69(1):262–277, 1917.

[42] A. Renda, J. Frankle, and M. Carbin. Comparing rewinding and fine-tuning in neural network pruning. *ICLR 2020*, 2020.

[43] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[44] B. Rouet-Leduc, C. Hulbert, N. Lubbers, K. Barros, C.J. Humphreys, and P.A. Johnson. Machine learning predicts laboratory earthquakes. *Geophysical Research Letters*, 44(18):9276–9282, 2017.

[45] J. Salamon and J. P. Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, March 2017.

[46] J. Sandino, G. Pegg, F. Gonzalez, and G. Smith. Aerial mapping of forests affected by pathogens using uavs, hyperspectral sensors, and artificial intelligence. *Sensors*, 18(4), 2018.

[47] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[48] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180, 2015.

[49] C. Sprunk, G. D. Tipaldi, A. Cherubini, and W. Burgard. Lidar-based teach-and-repeat of mobile robot trajectories. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3144–3149, 2013.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Ravinovich. Going deeper with convolutions. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[51] L. Tai and M. Liu. Imagement classification with deep convolutional neural networks. *In Advances in neural information processing systems*, pages 1097–1105, 2012.

[52] L. Tai and M. Liu. Mobile robots exploration through cnn-based reinforcement learning. *Robotics and Biomimetics*, 2016.

[53] T. Talaviya, D. Shah, N. Patel, H. Yagnik, and M. Shah. Implementation of artificial intelligence in agriculture for optimisation of irrigation and application of pesticides and herbicides. *Artificial Intelligence in Agriculture*, 4:58 – 73, 2020.

[54] Z. Tao, P. Bonnifait, V. Frémont, and J. Ibañez-Guzman. Mapping and localization using gps, lane markings and proprioceptive sensors. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 406–412, 2013.

[55] J.P. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[56] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5:253–271, 1998.

[57] U. Weiss and P. Biber. Plant detection and mapping for agricultural robots using a 3d lidar sensor. *Robotics and Autonomous Systems*, 59(5):265 – 273, 2011.

[58] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[59] R.W. Wolcott and R.M. Eustice. Robust lidar localization using multiresolution gaussian mixture maps for autonomous driving. *The International Journal of Robotics Research*, 36(3):292–319, 2017.

[60] D.F. Wolf and G.S. Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Auton Robot*, 19:53–65, 2005.

[61] P. Wozniak, H. Afrisal, R. G. Esparza, and B. Kwolek. Scene recognition for indoor localization of mobile robots using deep cnn. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[62] S. Xu, W. Chou, and H. Dong. A robust indoor localization system integrating visual localization aided by cnn-based image retrieval with monte carlo localization. *Sensors*, 19(2), 2019.

[63] L. Zhang and B. K. Ghosh. Line segment based map building and localization using 2d laser rangefinder. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 3, pages 2538–2543, 2000.