UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

"USO DE REDES NEURONALES DE AJUSTE DE DATOS MEDIANTE DESCRIPTORES HOLÍSTICOS PARA LOCALIZACIÓN VISUAL DE UN ROBOT MÓVIL EN CONDICIONES DE TRABAJO REALES"

TRABAJO FIN DE GRADO

Septiembre - 2021

AUTOR: Antonio Pinilla Riquelme

DIRECTOR/ES: Luis Payá Castelló

Sergio Cebollada López

Índice general

Índice de figuras

Capítulo 1

1. Introducción

Cada año hay un aumento de robots móviles en todos los ámbitos, ya sea para uso doméstico o industrial, y esto es debido a su gran versatilidad a la hora de resolver problemas. Este incremento es provocado por el descenso natural de los precios a lo largo de los años y al aumento de la capacidad computacional que permite adecuar este tipo de robots a un mayor número de entornos. Sin embargo, todavía queda un largo camino por recorrer por la falta de autonomía en dichos robots.

Para que un robot fuera totalmente autónomo tendría que ser capaz de navegar en todo tipo de entornos, conocidos o desconocidos, resolviendo cualquier problema u obstáculo que se le pudiera plantear. Para ello, deberá realizar tanto una tarea de *mapping*, es decir, construir un modelo del entorno, y una tarea de localización para conocer en que punto de ese entorno se encuentra, si ambas son realizadas al mismo tiempo se diría que el robot realiza una tarea de SLAM.

La mejor forma de solventar estas tareas a las que se enfrentan los robots es con el uso de sensores que permitan captar la información externa para realizar un mapeado y posteriormente procesar esa información e intentar localizar la posición de estos. El sensor usado en este trabajo y el más común es una cámara que nos proporciona imágenes omnidireccionales, las cuales serán tratadas con el fin de extraer su información para más tarde usar técnicas de aprendizaje automático (*machine learning*) como son las redes neuronales que serán entrenadas con esos datos. Esta extracción de información puede ser realizada de dos formas, ya sea con descriptores de apariencia local como pueden ser SIFT y SURF entre muchos otros, o con

descriptores de apariencia global como HOG, Firma de Fourier o GIST, siendo este último uno de los utilizados en este proyecto.

Por tanto, teniendo en cuenta todo lo anterior, este trabajo se basará en la creación de descriptores de apariencia global o holísticos a partir de una base de datos de imágenes omnidireccionales para su posterior uso en el entrenamiento de una red neuronal con el fin de conseguir la localización del robot con el menor error posible y un coste computacional aceptable. También se hará una comparación entre distintos descriptores y un estudio de los diferentes parámetros de entrenamiento y criterios de parada que posee la función de entrenamiento del programa informático Matlab con el que se realizarán todos los experimentos.

En este trabajo se ha utilizado la base de datos de Bielefeld que cuenta con imágenes omnidireccionales captadas por un sensor visual montado sobre el robot móvil apuntando a un espejo convexo hiperbólico. Dichas imágenes han sido tomadas durante distintos periodos del día y con la obstrucción de diferentes objetos tales como sillas o plantas, por tanto, la red neuronal entrenada con el descriptor extraído de estas imágenes será más robusto frente a oclusiones y cambios de iluminación, lo que proporcionará al robot una mayor autonomía en condiciones de trabajo reales.

Capítulo 2

2. Sensores en robótica móvil

Uno de los principales problemas en robótica es la determinación de un robot móvil en su entorno. Para realizar esta tarea tan importante es imprescindible el uso de sensores que nos permitan captar la información necesaria. En robótica los sensores se suelen dividir en exteroceptivos y propioceptivos dependiendo del lugar de donde obtienen la información.

Los sensores propioceptivos miden variables internas del robot como el nivel de batería, la carga y posición de la rueda o la velocidad del motor. Dentro de estos sensores encontramos los sensores odométricos como son los encoders, que convierten movimiento en una secuencia de pulsos digitales y pueden ser utilizados para medir velocidad, aceleración o posición. También se encuentran dentro de esta familia de sensores el acelerómetro que mide la aceleración generada cuando una masa es afectada por un cambio en su velocidad, y el giroscopio el cual sirve para conocer y mantener la orientación y la velocidad angular. Este tipo de sensores no suelen ser muy utilizados de forma individual debido a su falta de robustez, por lo que requieren de una combinación de más sensores para realizar un trabajo óptimo.

Por otro lado, tenemos los sensores exteroceptivos los cuales adquieren información del entorno del robot como distancia a objetos, intensidad de la luz o la amplitud de sonido. Dentro de estos sensores podemos encontrar los sensores de rango entre los que se encuentran los infrarrojos, sensores láser y ultrasonido, como el radar y el sonar, también sensores como los sistemas de posicionamiento global (GPS), y sensores de visión como las cámaras. Este tipo se sensores son más importantes en tareas de navegación de robots móviles puesto que la información del entorno es imprescindible para poder moverte en él, por lo que, se explicarán en mayor detalle algunos de los más usados.

El sonar (*Sound Navigation and Ranging*) es un sensor especialmente usado en entornos subacuáticos que utiliza la propagación del sonido en dichos entornos para determinar la ubicación, velocidad y distancia, entre otras características, de objetos, formaciones rocosas y el lecho submarino. Este sensor transmite una señal ultrasónica de la que es conocida su velocidad y mide el tiempo que tarda el reflejo de esa señal sobre una superficie en regresar. El sonar es usado para reemplazar al radar en el ámbito submarino debido a que este último hace uso de ondas electromagnéticas que no son viables en el agua debido a la alta conductividad del medio acuático.

Los sensores láser son otro tipo de sensores de rango que se utilizan para detectar la posición de los objetos. Estos sensores miden la distancia al objetivo emitiendo un rayo láser y calculando el tiempo de vuelo (TOF), es decir, el tiempo que tarda en retornar ese rayo al sensor.

Sin embargo, en tareas de navegación, sobre todo en interiores, todos los sensores comentados anteriormente no son tan utilizados como los que se explicarán a continuación.

2.1. Sensores de visión

Otra opción frente a los sensores exteroceptivos vistos anteriormente son los sensores de visión que cada vez son más utilizados por las grandes ventajas que aportan con respecto al resto. Algunas de estas ventajas son la gran cantidad de información que son capaces de extraer y que esa información puede ser tridimensional. Estos sensores también son relativamente más baratos y eficientes al presentar un menor uso de batería, además, a diferencia de otros sensores estos son bastante versátiles puesto que pueden ser utilizados tanto en interiores como en exteriores.

Estos sensores de visión o cámaras son herramientas de formación de imágenes que resultan de la proyección de la luz en una superficie sensible a esta. Se pueden obtener diferentes configuraciones de cámaras dependiendo de la geometría de la superficie, las propiedades ópticas y distribución geométrica de los fotorreceptores y si dispone de lentes simples o múltiples afectando a la forma en la que la luz se obtiene y se proyecta.

Los sensores de visión también pueden ser configurados dependiendo del número de cámaras usadas y el campo de visión que estas ofrecen. Existen sistemas de visión basados en configuraciones monoculares, binoculares y también trinoculares. Estas dos últimas configuraciones tienen la ventaja de que permiten medir la profundidad de las imágenes y, aunque son capaces de obtener la información completa del entorno se requieren varias imágenes para realizar esa tarea.

En los últimos años, en contraste con las configuraciones anteriormente mencionadas han surgido sistemas que son capaces de obtener información visual omnidireccional. Estos sistemas han ganado popularidad gracias a la gran cantidad de información que provee, ya que poseen un campo de visión de 360º entorno al robot.

2.2. Sistemas de visión omnidireccionales

Como se ha mencionado en la sección anterior estos sistemas de visión omnidireccionales son más usados frente al resto de sistemas de visión convencionales debido a su amplio campo de visión las características que aparecen en las imágenes obtenidas con estos sistemas son más estables, ya que, las imágenes permanecen más tiempo cuando el robot se mueve. Además, también proporcionan información que permite calcular la posición del robot independientemente de su orientación. Estos sistemas ofrecen una información más completa sobre el entorno con un número reducido de imágenes.

Existen muchas configuraciones que permiten adquirir información visual omnidireccional como por ejemplo los sistemas *ladybug* que están constituidos por varios sensores de visión sobre una cámara apuntando en distintas direcciones con lo que capturan imágenes desde distintos puntos para más tarde reconstruir la imagen omnidireccional.

Figura 1. Sistema de visión omnidireccional *ladybug.*

Otro sistema de visión que captura un amplio campo de visión son las cámaras con lentes de ojos de pez. Estas lentes son capaces de proporcionar un campo de visión superior a 180º, por lo que con la utilización de dos de estas lentes se es capaz de lograr un campo de visión esférico de 360º como sucede en la cámara Gear 360 de Samsung.

Figura 2. Cámara Gear 360 de Samsung.

Estos dos sistemas anteriores realizan la tarea deseada de una forma eficaz, pero presentan un gran inconveniente, y es que la creación de una imagen esférica a partir de dos cámaras no es una tarea sencilla, ya que cada cámara posee unas condiciones de luz distintas.

Otro sistema de visión omnidireccional es el sistema catadióptrico que utiliza una cámara apuntando a un espejo convexo esférico, cónico, parabólico o hiperbólico, consiguiendo así aumentar el campo de visión y obteniendo un ángulo de visión de 360º.

Las características más importantes para tener en cuenta en estos sistemas en tareas de *mapping* y localización son la calibración y el centro único de proyección. Gracias al centro único de proyección del sistema todos los rayos de luz que llegan al espejo convergen en un solo punto haciendo posible la obtención de imágenes sin distorsión. También hay que tener en cuenta que muchas tareas de *mapping* y localización requiere del conocimiento de todos los parámetros de la cámara del sistema catadióptrico, por lo que será necesario hacer una calibración de este. Además, cabe destacar que la base de datos utilizada para este proyecto fue tomada con uno de estos sistemas, concretamente, una cámara enfocando a un espejo convexo hiperbólico.

Figura 3. Sistema de visión catadióptrico con espejo convexo hiperbólico.

Capítulo 3

3. Descriptores en robótica móvil

Como se ha comentado en la sección anterior el método más utilizado para la descripción del entorno son las imágenes omnidireccionales. Estas imágenes son muy cambiantes dependiendo del movimiento del robot, las condiciones de iluminación o incluso la variación de la posición de ciertos objetos. Por tanto, es muy necesario extraer la información más relevante para conseguir un sistema de *mapping* y localización muy robusto. Existen dos tipos de métodos para realizar esta tarea: los métodos basados en la extracción y descripción de características locales y métodos basados en apariencia global.

3.1. Métodos basados en características locales

Estos métodos se basan en la extracción de puntos, objetos o regiones de la escena. Una vez elegidas las características de la imagen, se obtiene un vector por cada una de ellas que es invariante a cambios en la posición o rotación del robot, todos esos vectores son combinados para obtener así el descriptor de esa imagen.

Existen diferentes métodos que siguen este procedimiento para describir una escena, entre ellos, cabe destacar que los más populares y utilizados son el SIFT y SURF. En cuanto al descriptor SIFT (Scale Invariant Feature Transform), fue desarrollado por David Lowe y publicado en 1999. Este descriptor obtiene características que son invariantes a rotación, escalado, cambios en el punto de vista de la cámara y cambios de iluminación. La obtención de puntos característicos de SIFT reside en la detección de regiones de la imagen en las que se producen cambios de gradiente significativos a ambos lados de esos puntos. Además, es capaz de detectar regiones en una imagen en la que algunas de sus propiedades se mantienen constantes, a estas regiones se las conoce como *blob*.

Por otro lado, el descriptor SURF (Speeded-Up Robust Features) fue desarrollado por Herbert Bay y presentado en 2006. Este descriptor está inspirado en SIFT, siguiendo sus mismos principios, pero tiene un coste computacional menor y una mayor robustez frente a transformaciones de imágenes.

Sin embargo, en los últimos años estos métodos basados en características locales no están siendo tan utilizados en tareas de *mapping* y localización debido a las ventajas en dichas tareas de los métodos que se van a detallar en la siguiente subsección.

3.2. Métodos basados en apariencia global

A diferencia de los anteriores, los descriptores holísticos o de apariencia global trabajan con la imagen completa sin extraer ninguna información local y se basan en la obtención de un solo vector por imagen que contiene toda la información de dicha imagen. Al trabajar con un único vector estos métodos presentan una gran ventaja puesto que, consiguen hacer mas fáciles las tareas de *mapping* y localización, llevándolas en algunos casos, a resolverse mediante una simple comparación de pares de imágenes disminuyendo así el coste computacional.

Los descriptores holísticos fueron elegidos para su utilización en este proyecto debido a la gran cantidad de ventajas, pero también debemos tener en cuenta que presentan ciertos inconvenientes sobre todo cuando se trabaja en entornos de interiores debido al *visual aliasing*, fenómeno que ocurre en ambientes con estructuras visuales repetitivas. Estos descriptores pueden ser divididos en dos grupos: descriptores de apariencia global basados en métodos analíticos, estos fueron los primeros descriptores que se utilizaron en visión por computador, y descriptores de apariencia global basados en técnicas de *deep learning.*

3.2.1. Descriptores de apariencia global basados en métodos analíticos

Los descriptores holísticos conseguidos mediante métodos analíticos se basan principalmente en cálculos de gradientes y orientaciones de diferentes pixeles que componen la imagen. Entre estos métodos analíticos para la obtención de descriptores globales encontramos: PCA, la firma de Fourier, la transformada de Radon, HOG y GIST.

En primer lugar, el descriptor PCA (Principal Components Analysis) fue una de las primeras alternativas robustas. La idea de este descriptor es que dados unos datos de interés en un espacio multidimensional se proyecten a un espacio de menor dimensión conservando tanta información como sea posible.

En segundo lugar, la firma de Fourier se obtiene al aplicar a la imagen la transformada de Fourier en cada una de sus filas extrayendo así la información más relevante. Por un lado, si trabajamos con imágenes panorámicas se ha demostrado que es viable el uso de esta transformada, pero también puede ser utilizada en imágenes omnidireccionales aplicando la transformada de Fourier esférica. En ambos casos el resultado es la compresión de la información original.

En tercer lugar, *Histograms of Oriented Gradients* (HOG) se ha utilizado tradicionalmente como un método de descripción en el campo de la detección de objetos. La idea fundamental de los descriptores de Histograma de Orientación del Gradiente es la descripción de la apariencia de un objeto y su forma dentro de la imagen mediante la distribución de la intensidad de gradientes o la dirección de bordes. Este método consiste en la división de la imagen en pequeñas celdas interconectadas y el cálculo de los histogramas de orientaciones del gradiente de los respectivos píxeles de cada celda. Tras

esto, se combinan los histogramas en un solo vector obteniendo el descriptor HOG.

Otro posible descriptor holístico es la transformada de Radon que se suele utilizar en algunas tareas de visión por computador como la descripción de formas y en segmentación. Esta transformada consiste matemáticamente en la integral de una función bidimensional sobre líneas rectas, que también puede ser invertida para reconstruir imágenes a partir de sus proyecciones integrales de línea.

En último lugar, el descriptor GIST intenta copiar el sistema de percepción humana con el fin de describir escenas por reconocimiento de regiones con un color o textura prominente con respecto a sus vecinos. Este descriptor fue presentado por Oliva y Torralba con el nombre de *holistic representation of the spatial envelope*. Estos autores muestran que haciendo uso de este método se crea un espacio multidimensional en el que las escenas de la misma categoría semántica son proyectadas en puntos próximos.

El método de descripción GIST, matemáticamente intenta realizar una codificación de la información usando la transformada bidimensional de Fourier en varias regiones de la escena. Las regiones anteriores se distribuyen en una cuadrícula de forma regular y se les aplica el método PCA para conseguir un único vector de descripción con dimensión disminuida.

En este proyecto, GIST es uno de los descriptores con los que se han realizado los experimentos y se construye con la información de intensidad conseguida tras convolucionar la imagen con un filtro de Gabor con diferentes orientaciones y niveles de resolución. Más tarde, estos datos son guardados como la media del valor de intensidad, y al igual que HOG, se divide la imagen en celdas horizontales y se calcula la intensidad media de cada celda.

3.2.2. Descriptores de apariencia global basados en técnicas de *deep learning*

El *deep learning* o aprendizaje profundo es un subconjunto de *machine learning* muy utilizado en robótica móvil, puesto que hace uso de las redes neuronales profundas para la resolución de problemas. El método de descripción mediante aprendizaje profundo se basa la obtención de un descriptor holístico extraído de una capa intermedia de una red neuronal convolucional previamente entrenada.

Las CNNs (Convolutional Neural Networks), son un tipo especifico de modelo concebido para aceptar datos de entrada bidimensionales como pueden ser las imágenes. Estos modelos reciben su nombre debido a la operación matemática lineal de convolución que siempre esta presente en al menos una de las capas de la red neuronal. La operación convolucional más usada en aprendizaje profundo es la convolución bidimensional de una imagen I de dos dimensiones con un kernel K bidimensional, como se expone en la siguiente ecuación:

$$C(i,j) = (I * K) = \sum_m \sum_n I(m,n) * K(i-m, j-n)$$

El resultado de la operación de convolución normalmente se pasa a través de una función de activación no lineal (ReLu), y después se vuelve a modificar mediante una función de *pooling* que reemplaza la salida en un cierto lugar con un valor obtenido por datos de salida cercanos. Esta función de *pooling* ayuda a que la representación aprendida sea invariante a pequeñas translaciones de la imagen de entrada y realiza un *subsampling* de los datos de entrada. La función *pooling* más común es *max pooling* que reemplaza la

salida con el máximo valor de activación dentro de una región cercana rectangular.

Las capas de convolución y de *pooling* se encuentran apiladas para conseguir el aprendizaje de características de forma jerárquica, es decir, cuando se aprende de las imágenes las capas más cercanas a la entrada aprenden representaciones de características de bajo nivel como esquinas y bordes, mientras que aquellas más próximas a la salida aprenden representaciones de más alto nivel como contornos y partes de objetos. Una vez que las características de interés han sido aprendidas, sus activaciones son usadas en las capas finales, que suelen estar constituidas por neuronas totalmente interconectadas, para conseguir la clasificación de la entrada.

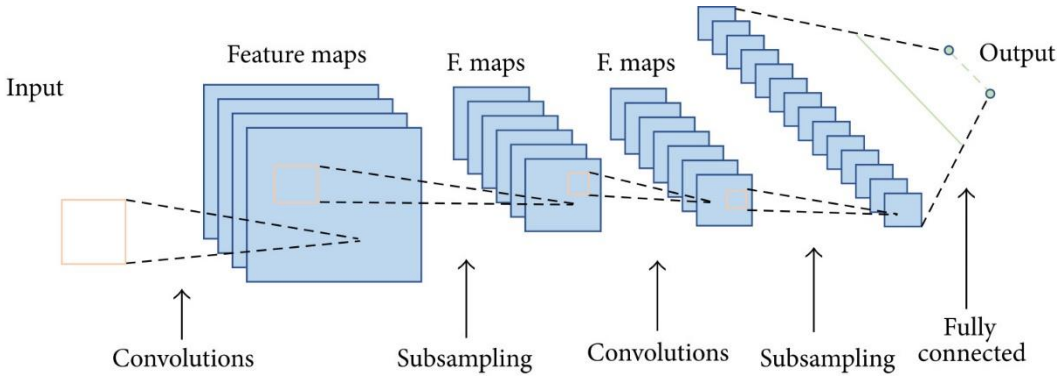


Figura 4. Red neuronal convolucional genérica.

En este proyecto se utilizó la arquitectura de red neuronal convolucional ALEXNET, que es una red neuronal pre-entrenada y cuyo objetivo es el de solucionar tareas de clasificación. Esta tarea es resuelta mediante el entrenamiento de la CNN con un conjunto de imágenes etiquetadas, para posteriormente introducir una imagen test, de la que se desconoce la etiqueta, como entrada. Las salidas obtenidas serán las etiquetas más probables de esa imagen test.

Sin embargo, en los últimos años se han hecho estudios que demuestran la posibilidad de utilizar la CNN ALEXNET en tareas de localización introduciéndole una imagen para obtener de sus capas intermedias un vector que se puede usar como descriptor holístico de la misma.

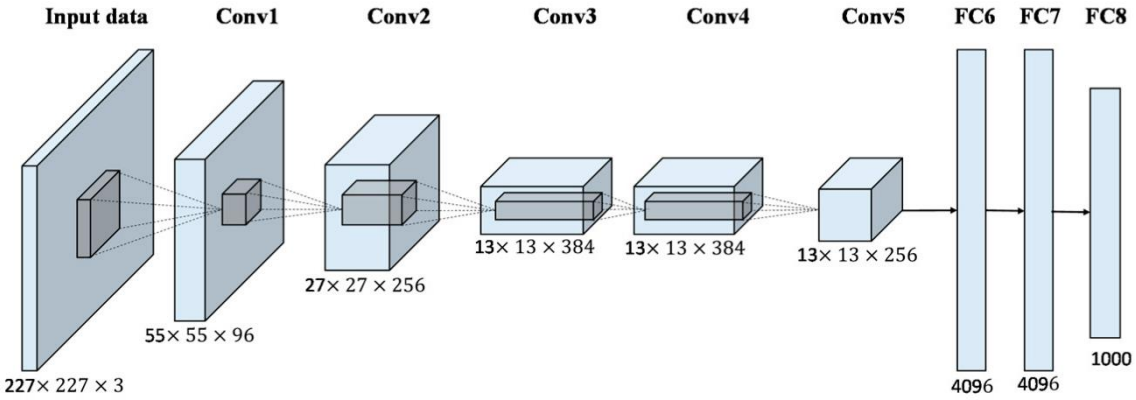


Figura 5. Arquitectura de la red neuronal convolucional pre-entrenada ALEXNET.

En la figura 5 se observa la CNN ALEXNET en la que se puede apreciar, que las primeras capas (conv1-conv5) son capas convolucionales. Por otro lado, tenemos las capas fc6-fc8 denominadas *fully connected*. En este trabajo se estudiará la viabilidad de los descriptores obtenidos de la capa fc6 tras la introducción de las imágenes de la base de datos elegida.

Por último, los descriptores holísticos basados en herramientas de *deep learning* como las CNNs, son una solución muy interesante en tareas de localización, pero debemos tener en consideración que estos descriptores son muy dependientes de las condiciones de iluminación. Por ello, en este proyecto también se tuvo en cuenta esta consideración y se eligió una base de datos que contiene imágenes con diferentes condiciones lumínicas, con el fin de tener un mayor conocimiento sobre como estos cambios afectan al cálculo de

la localización del robot, tras entrenar y probar nuestra red neuronal en diferentes condiciones de luminosidad.

Capítulo 4

4. *Machine learning*

El aprendizaje automático o *machine learning* es una de las ramas de la inteligencia artificial que se centra en conseguir que las máquinas aprendan a reconocer patrones siendo previamente entrenadas y así conseguir disminuir la necesidad de programación externa.

Teniendo en cuenta lo anterior, no es extraño el aumento exponencial del uso de *machine learning* en tareas de inteligencia artificial, ya que proporciona una mayor automatización en entornos laborales disminuyendo la necesidad de personal programador en ciertos ámbitos. Otra ventaja de estas técnicas es el tratamiento de un mayor número de datos de forma autónoma, lo cual es muy interesante en una sociedad en la que el número de datos aumenta en gran cantidad cada año. Existen tres grandes grupos de algoritmos de *machine learning*: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje de refuerzo.

4.1. Aprendizaje supervisado

El aprendizaje supervisado hace uso de bases de datos etiquetadas para entrenar algoritmos que clasifiquen datos o puedan predecir las salidas de forma precisa. Este aprendizaje utiliza una base de datos de entrenamiento que incluye las entradas y salidas correctas lo que permite al modelo aprender con el tiempo. Estos algoritmos miden su precisión por medio de la función de pérdida ajustando el error hasta que este es suficientemente pequeño. Además, estos pueden ser usados para solventar muchos tipos de problemas como, por ejemplo, la clasificación de spam en mensajes electrónicos. Dentro de este tipo de aprendizaje podemos encontrar diversos tipos de algoritmos: el clasificador de Naïve Bayes, regresión logística, regresión lineal, SVM (support vector machines), y redes neuronales artificiales (RNA) entre otros.

La regresión lineal es usada para identificar la relación entre una variable dependiente y una o más variables independientes y es típicamente utilizada para realizar predicciones sobre futuros resultados. Cuando solo se tiene una variable independiente y una dependiente es conocido como regresión lineal simple, mientras que, si tiene más de una variable independiente se denomina regresión lineal múltiple. Para cada tipo de regresión lineal, se dibuja una línea de ajuste que es calculada por el método de mínimos cuadrados.

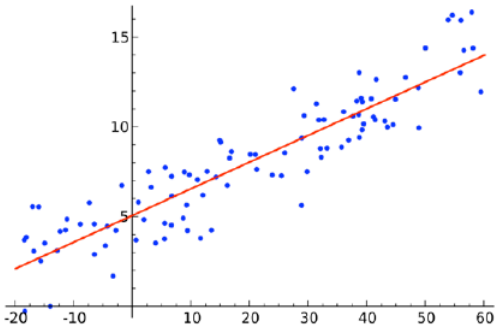


Figura 6. Ejemplo de regresión lineal.

Mientras que la regresión lineal se usa cuando las variables dependientes son continuas, la regresión logística es utilizada cuando estas variables son categóricas, es decir, que tienen salidas binarias. Mientras que los dos algoritmos de regresión intentan comprender la relación entre los datos de entrada, la regresión logística solo es utilizada para resolver problemas de clasificación binaria.

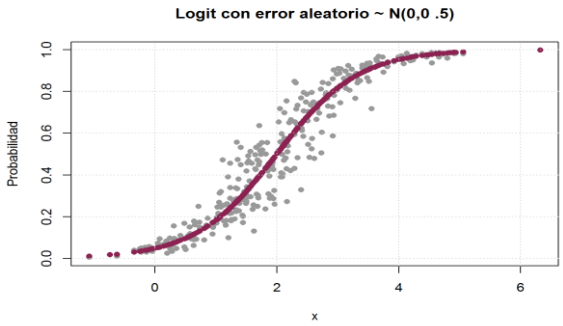


Figura 7. Ejemplo de regresión logística.

En tercer lugar, el clasificador de Naïve Bayes adopta el principio de clase condicional independiente del teorema de Bayes. Esto significa que la presencia de una característica no afecta a la presencia de otra en la probabilidad de un cierto resultado, y cada predictor tiene el mismo efecto en ese resultado. Esta técnica es usada principalmente de clasificación de texto y sistemas de recomendación.

Por otro lado, el SVM fue introducido por Cortes y Vapnik es un algoritmo que puede ser utilizado tanto para tareas de clasificación como de regresión. El algoritmo marca cada dato de entrada en un espacio n-dimensional, donde n es el número de características, con el valor de cada característica situado en una coordenada particular. Tras esto, se puede realizar la tarea de clasificación encontrando el hiper-plano o hiper-planos que mejor diferencia las categorías.

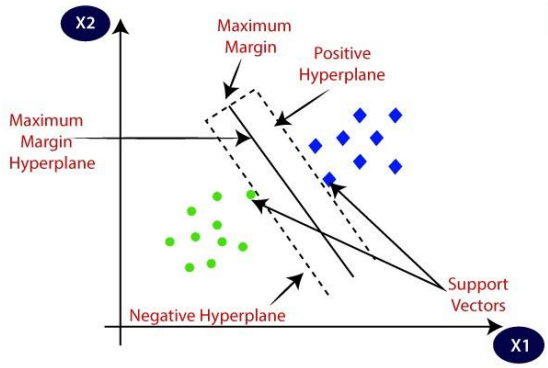


Figura 8. Ejemplo de algoritmo SVM.

En último lugar, las redes neuronales artificiales (RNA) son modelos computacionales que procesan la información imitando el funcionamiento de las neuronas biológicas. Estas redes están compuestas por nodos, también denominados neuronas, que reciben y transmiten información. El objetivo de las redes neuronales artificiales es ayudar a que los sistemas informáticos funcionen como un cerebro humano, teniendo su misma capacidad de aprendizaje y pensamiento.

La estructura de una RNA consiste en una capa de entrada, una capa oculta y una capa de salida, como se puede apreciar en la figura 9, en la que n es el número de neuronas que componen la capa de entrada y m el número de neuronas que componen la capa oculta. De estas redes neuronales surgen las técnicas de *deep learning* al aumentar el número de capas dispuestas entre la entrada y la salida.

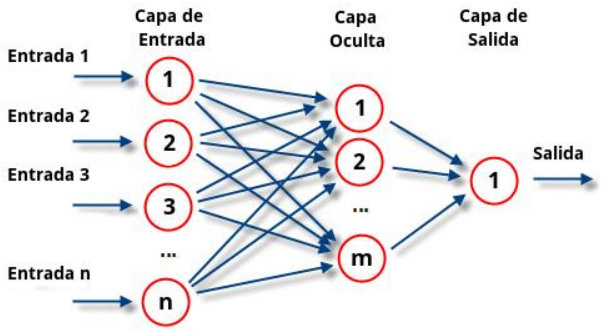


Figura 9. Estructura de una red neuronal artificial.

También existen dentro del aprendizaje supervisado técnicas que se basan en el uso de *deep learning* como, por ejemplo: *Feedforward Neural Networks* o redes neuronales prealimentadas, una variación popular de esta última como son las CNNs, que fueron ampliamente explicadas en la subsección 3.2.2 del capítulo anterior, y *Recurrent Neural Networks* (RNNs) o redes neuronales recurrentes.

Las redes neuronales prealimentadas, también conocidas como perceptrones multicapa (MLPs), son los modelos de aprendizaje supervisado más comunes. Su objetivo consiste en trabajar como aproximador de funciones y tienen la misma estructura que las RNA a excepción del número de capas ocultas que puede ser mayor. Durante el proceso de aprendizaje los pesos de cada nodo son actualizados usando el método de *backpropagation* para conseguir optimizar la función de pérdida.

En contraste a las redes neuronales prealimentadas, las redes neuronales recurrentes (RNNs) son modelos en los cuales la salida es una función no solo de las entradas actuales si no, también de las anteriores salidas. Esto significa que Las RNNs tiene memoria de las salidas previas y, por tanto, pueden codificar la información presente en la propia secuencia a diferencia de las MLPs. Como consecuencia este tipo de modelos pueden ser muy útiles para aprender de datos secuenciales.

Por último, estos métodos basados en aprendizaje supervisado suelen usarse para reconocimiento de imágenes y objetos, análisis predictivo y detección de spam entre otros. Aunque presentan ciertas desventajas como el consumo excesivo de tiempo, las bases de datos son mas propicias al error humano, y a diferencia del aprendizaje no supervisado, estos métodos no pueden agrupar o clasificar datos por si solos.

4.2. Aprendizaje no supervisado

El aprendizaje no supervisado apunta hacia el desarrollo de modelos que son capaces de extraer representaciones significativas y de alto nivel de datos multidimensionales sin etiquetar. El algoritmo de aprendizaje automático estudia los datos para identificar patrones y determinar correlaciones mediante el análisis de los datos disponibles sin intervención de ningún operador humano que proporcione instrucciones. Esta funcionalidad está inspirada en el córtex visual del cerebro que requiere de una cantidad muy pequeña de datos etiquetados.

El *clustering* es el método de aprendizaje no supervisado más usado y consiste en agrupar la información aportada en vectores en base a la similitud entre los propios vectores de información.

El aprendizaje no supervisado tiene muchas aplicaciones como, por ejemplo: reconocimiento de objetos, detección de imagen, clasificación y

segmentación para diagnosis de pacientes y detección de anomalías. Estos métodos, al igual que los de aprendizaje supervisado, presentan algunas desventajas como la alta complejidad computacional al tener grandes volúmenes de datos, tiempos de entrenamientos más largos y un mayor riesgo de resultados imprecisos.

4.3. Aprendizaje de refuerzo

El aprendizaje de refuerzo se encuentra en un punto intermedio entre el aprendizaje supervisado y el no supervisado. En estos métodos un agente es definido para interactuar con el entorno buscando realizar la mejor acción para cada estado en cualquier instante de tiempo.

Capítulo 5

5. Material y métodos

Para la realización de estos experimentos era primordial tener una buena base de datos, en este proyecto se utilizó la base de datos de imágenes omnidireccionales de la universidad de Bielefeld en Alemania. Estas imágenes se tomaron en el laboratorio de robótica del grupo de ingeniería de computadores con un sistema catadióptrico, es decir, con una cámara montada sobre el robot ActivMedia Pioneer 3-DX apuntando hacia arriba a un espejo convexo hiperbólico, como se puede apreciar en la figura 10. La cámara es una ImagingSource DFK 4303 y capturo las imágenes a la resolución más alta posible (752 x 564).

(a) (b)

Figura 10. (a) Robot ActivMedia Pioneer 3-DX con una cámara y un espejo hiperbólico. (b) Imagen ampliada del sistema catadióptrico.

La zona de captura de las imágenes tiene unas dimensiones de 2,7 x 4,8 m, que cubren prácticamente todo el espacio libre del suelo. Para la captura de la base de datos se posicionó el robot a lo largo de una línea de 10 posiciones de inicio separadas 30 cm. El robot se va posicionando manualmente en cada una de las posiciones de inicio y se activa por conexión ethernet. Al activarse captura una imagen de la posición inicial y se mueve 30 cm hacia delante para tomar otra instantánea, y así sucesivamente hasta llegar

al final de la sala. Después se coloca en la siguiente posición inicial y se repite el proceso.

En esta base de datos fueron capturadas 170 imágenes en cada una de las condiciones, en la misma habitación como se expone a continuación:

- *Original*: estas imágenes fueron tomadas con el estado de la habitación por defecto, con los dos tubos de luz fluorescentes del techo encendidos y las cortinas y la puerta cerradas.
- *Chairs*: a original se le añadieron tres sillas de oficina y se capturaron las imágenes.
- *Arboreal*: una planta de tres metros de altura fue añadida a la disposición de *original* para la captura de las imágenes.
- *Screen*: se añade una pantalla de proyección de una altura de dos metros y medio a la disposición de *original* y se toman las imágenes.
- *Day*: estas imágenes fueron capturadas con las cortinas abiertas a plena luz del día.
- *Twilight*: las imágenes fueron capturadas con las cortinas y la puerta abiertas en un periodo de aproximadamente cuarenta minutos durante la puesta de sol.

- *Doorlit*: a partir de la disposición de original, se apaga el tubo de luz fluorescente cercano a la ventana.

- *Winlit*: a partir de la disposición de original, se apaga el tubo de luz fluorescente cercano a la puerta.

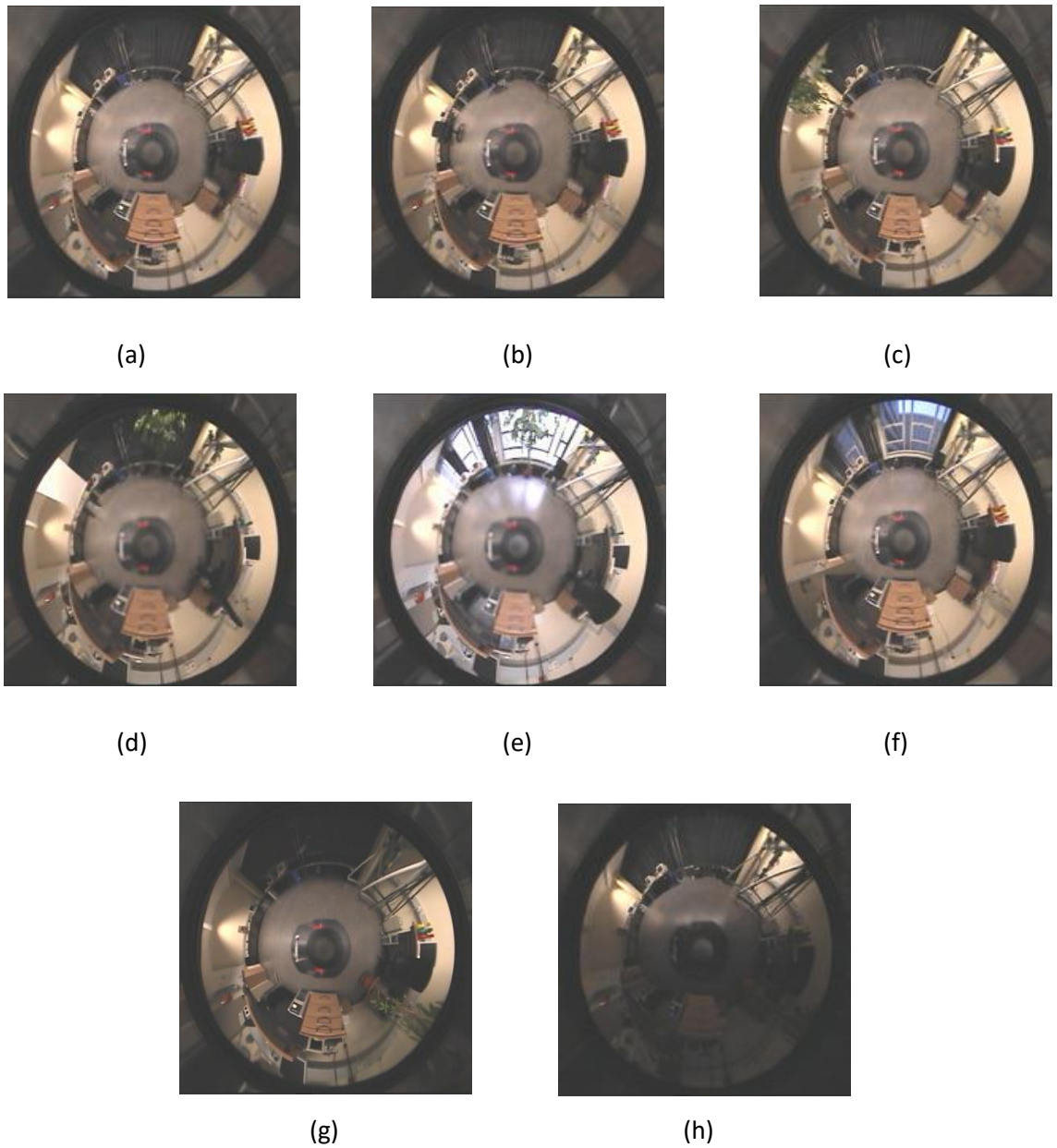


Figura 11. Base de datos de la universidad de Bielefeld en Alemania. (a) Original. (b) Chairs. (c) Arboreal. (d) Screen. (e) Day. (f) Twilight. (g) Doorlit. (h) Winlit.

Por tanto, se ha elegido esta base de datos ya que, es ideal para este proyecto debido a la robustez que podrá presentar frente a cambios de iluminación y oclusiones mediante objetos, que podrían ser condiciones de trabajo reales del robot.

Una vez tenemos nuestra base de datos debemos conocer también el *ground truth* o posición real (x,y) en la que fue tomada cada una de las imágenes. Sabiendo el punto inicial del robot y que la rejilla de captura de imágenes tiene una resolución de 30 cm, podemos calcular el *ground truth*. Teniendo ya la base de datos de imágenes y la posición de cada una de estas, estamos preparados para resolver las tareas de *mapping* y localización, que se llevarán a cabo a través del programa informático Matlab.

5.1. Tarea de *mapping*

En este trabajo la tarea de *mapping* consiste en la representación mediante descriptores extraídos de imágenes omnidireccionales del entorno del robot móvil, para más tarde ser usados en la tarea de localización que es el objetivo final del proyecto.

Para la realización de esta tarea se han elegido dos métodos de creación de descriptores holísticos, un método analítico como es GIST y otro basado en *deep learning* como es el descriptor basado en la CNN ALEXNET.

El descriptor basado en ALEXNET explicado en la subsección 3.2.2, se obtiene mediante la introducción de la imagen como entrada en la red neuronal convolucional ALEXNET, que ha sido previamente entrenada para tareas de clasificación. Una vez hecho esto, se extraerá la capa totalmente conectada fc6 de la red neuronal convolucional mediante Matlab, que contiene el descriptor de nuestra instantánea. A diferencia de GIST, en este método si que podemos hacer uso de las imágenes RGB, es decir, imágenes que contienen los tres canales de color, ya que, esta CNN ha sido diseñada para trabajar con ellas. Hay que tener en cuenta, que la entrada de la CNN ALEXNET debe ser de tamaño 227 x 227 x 3, y nuestras imágenes omnidireccionales son de 752 x 564 x 3, por tanto, se requiere de una normalización. Esta normalización se realizará dentro de Matlab mediante la función *imresize*.

También haremos uso del descriptor GIST previamente explicado en la subsección 3.2.1, que requiere de imágenes en escala de grises para su cálculo. Por tanto, se procesará la base de datos de imágenes para convertirla del espacio RGB a escala de grises usando el programa informático Matlab.

5.2. Tarea de localización

Una vez se han conseguido los descriptores holísticos de las imágenes omnidireccionales de nuestra base de datos podemos comenzar la resolución de la tarea de localización. Esta tarea consiste en la introducción de los descriptores de las imágenes de entrenamiento, así como las posiciones en las que estas fueron tomadas, a la red neuronal con el fin de realizar el entrenamiento de esta. Tras esto, se introducen los descriptores de las imágenes de prueba, de las que nosotros conocemos las posiciones, pero la red neuronal no, y esta trata de estimar ese valor de posición.

La manera de medir la efectividad de los métodos de descripción propuestos será a través de dos parámetros: el coste computacional, es decir, el tiempo que tarda la red neuronal en entrenarse con el descriptor elegido, más el tiempo que tarda en obtener la salida dada una imagen de prueba y el error entre la salida de la red y la posición real de la imagen testeo introducida. La primera se llevará a cabo con la función tic toc de Matlab que mide el tiempo de procesamiento de las líneas de código que se encuentran entre los anteriores comandos. Mientras que, el error de posición se calculará utilizando la distancia euclídea entre el resultado estimado y la posición real de la forma en que se expone en la siguiente ecuación:

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Dados dos puntos P_1 y P_2 de coordenadas cartesianas (x_1, y_1) y (x_2, y_2), que corresponden al punto estimado de la red neuronal y la posición real del robot de un descriptor de una imagen, se utiliza la ecuación anterior para obtener la distancia entre ambos puntos que corresponde al error de posición. Después se realizará una media de los errores de posición de todos los descriptores de las imágenes para obtener nuestro valor final de error.

En este proyecto se hará un estudio de los diferentes parámetros y criterios de parada del entrenamiento de la red neuronal que nos proporciona Matlab a la hora de resolver la tarea de localización. Por consiguiente, es interesante mencionar cada uno de estos parámetros y criterios que serán utilizados para conseguir los resultados del próximo capítulo.

En primer lugar, se debe mencionar que la red neuronal de ajuste que se va a utilizar en Matlab se denomina fitnet y permite la programación del número de capas ocultas que queremos que tenga nuestra red y el número de neuronas en cada capa. También se puede elegir la función de entrenamiento para nuestra red neuronal, así como el porcentaje de nuestra base de datos de entrenamiento que se usará para entrenar, el porcentaje que se usará para testear y el porcentaje para la validación. Otras variables que se pueden elegir en la función de entrenamiento *train* de Matlab son: *max_fail* o *validation*, *goal* o *performance* y el parámetro *epochs*.

Para cada uno de los siguientes experimentos se estudiarán las mejores configuraciones de los parámetros y criterios de parada anteriormente mencionados.

Teniendo en cuenta que la base de datos de la que se dispone consta de 8 grupos de imágenes tomadas en condiciones distintas, debemos elegir cuales formaran parte de las imágenes de testeo y cuáles de las de entrenamiento, por tanto, se realizará un estudio de dos configuraciones de

prueba utilizando el descriptor basado en ALEXNET. Por un lado, se utilizarán las imágenes de *day* para testeo y se introducirán las demás, describiéndolas previamente, a la red neuronal para su entrenamiento. Y, por otro lado, se usarán las imágenes de *doorlit* para la realización de las pruebas y el resto será utilizado para el entrenamiento.

Tras la elección de la mejor configuración de las imágenes de testeo y entrenamiento, esta se usará para los demás experimentos, siendo uno de estos experimentos la localización con el descriptor GIST. Después, teniendo tanto los resultados de GIST como de ALEXNET, se realizará la comparativa entre ambos para determinar cuál de los dos algoritmos de descripción es el más adecuado.

Por último, realizaremos un estudio, con la mejor configuración de imágenes de testeo y el método de calculo de descriptores más apropiado, sobre la forma de calcular la localización, es decir, de extraer la posición del robot. Se propone entonces, la introducción por separado de las coordenadas (x,y) de la posición real del robot, realizando así dos entrenamientos independientes. El cálculo de la localización se realiza de la misma manera, pero solo se obtiene una coordenada de la posición en cada entrenamiento. Tras esto, se juntan las dos coordenadas en una matriz y se realiza la distancia euclídea para conseguir el error de posición. El tiempo de computo será la suma de cada entrenamiento más su tiempo de cálculo de la localización al introducir los descriptores de testeo.

Tras la resolución de la tarea de localización mediante los anteriores métodos, y sabiendo que las redes neuronales mejoran su eficacia cuanto mayor es la base de datos de la que disponen para su entrenamiento, se planteo el estudio de una técnica denominada *data augmentation*.

La técnica de *data augmentation* consiste en aplicar efectos a nuestra base de datos para aumentar el número de imágenes de las que se disponen intentando así conseguir un mejor entrenamiento de la red neuronal y hacerla más robusta frente a posibles errores en la captura de las imágenes del entorno del robot.

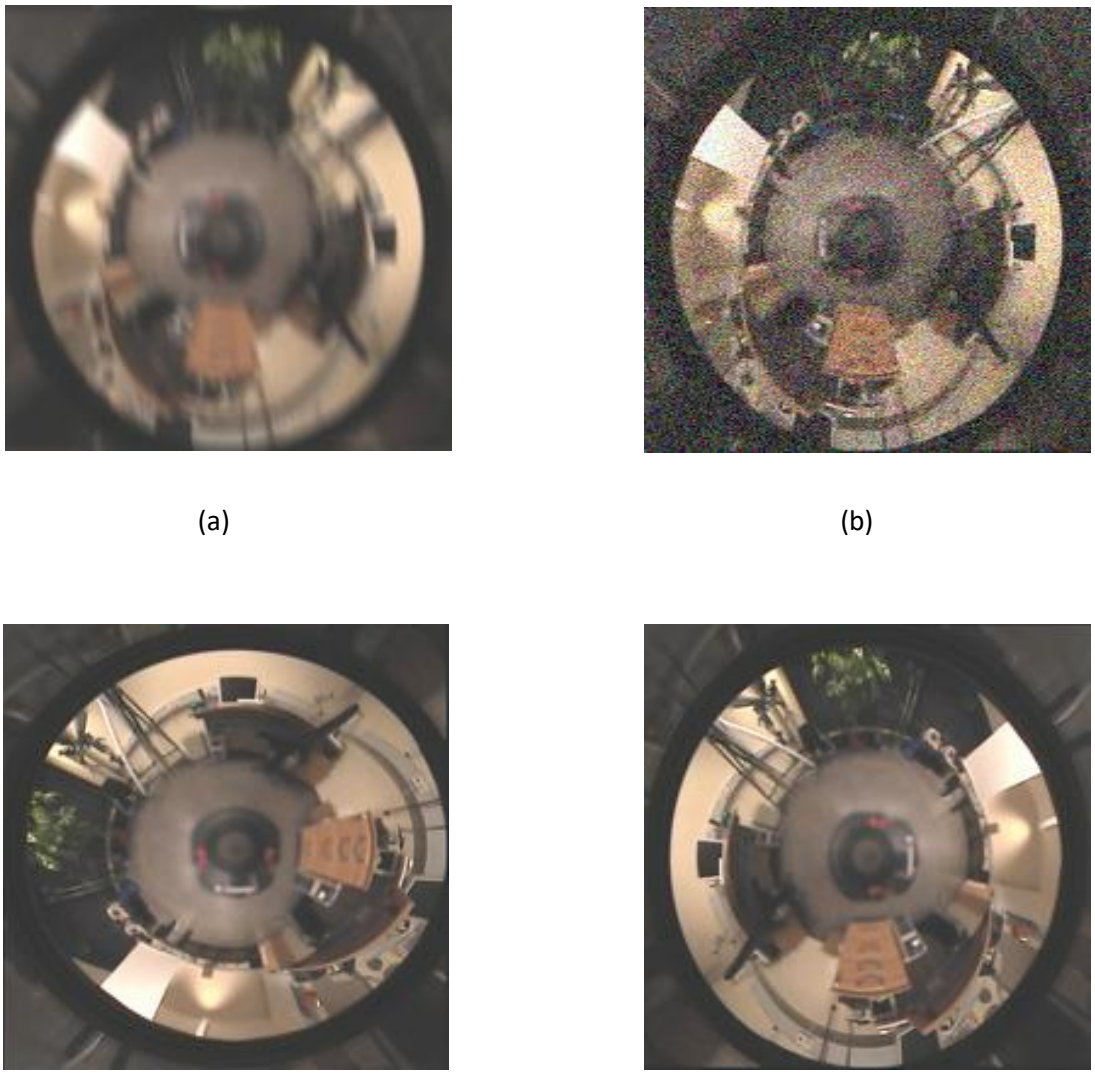
Los siguientes efectos se han aplicado a la base de datos utilizando Matlab, excepto a las imágenes de prueba correspondientes en cada experimento:

- *Blur*: este efecto también conocido como suavizado, actúa como un filtro de paso bajo, ya que deja pasar solo las bajas frecuencias. Las frecuencias en imágenes son los cambios de valores de los pixeles. El suavizado suele utilizarse para quitar ruido de las imágenes intentando dejar el resto de la imagen intacta.

- Ruido: es una variación del brillo o la información de color de una imagen que simula un fallo en la captura de esta. Existen varios tipos de ruido, en este proyecto se utilizó el ruido gaussiano.

- Translación: este efecto consiste en mover la imagen a lo largo del eje x o y, aunque dado que nuestra base de datos cuenta con imágenes omnidireccionales, una translación equivale a una rotación.

- Reflexión: para añadir este efecto a las imágenes de la base de datos se utiliza la función de Matlab *flip* que refleja la imagen alrededor de su eje horizontal, vertical o de ambos.



(a) (b)

(c) (d)

Figura 12. Efectos realizados a la base de datos de la universidad de Bielefeld en Alemania. (a) Blur. (b) Ruido. (c) Translación. (d) Reflexión.

Al añadir sobre nuestra base de datos cuatro efectos tendremos una cantidad de combinaciones igual a 15, por lo que nuestra base de datos aumentará 15 veces, teniendo previamente 170 imágenes por cada condición de captura, que se convierten en 2720 consiguiendo un total de 16320 imágenes. Existen otros efectos como darle brillo a la imagen, oscurecerla o colocarle oclusiones, pero no se han creído necesarios puesto que la base de datos ya cuenta con cambios de iluminación y oclusiones.

Cabe destacar, que esta técnica no pudo ser realizada con el algoritmo GIST debido a un extremo coste computacional, que no podía ser resuelto con el material del que se disponía. Por tanto, los experimentos con *data augmentation* solo serán realizados con el descriptor basado en la red neuronal convolucional ALEXNET. Estos experimentos con *data augmentation* serán los mismos que los mencionados anteriormente realizándose de la misma forma, pero con una base de datos mayor. Deberemos tener en cuenta que hay que cambiar el *ground truth* para que concuerde con los nuevos datos obtenidos y poder introducirlo, junto a los descriptores de la base de datos con *data augmentation,* de una forma correcta en la red neuronal para conseguir la localización del robot en su entorno.

Capítulo 6

6. Experimentos

Cabe destacar que en estos experimentos no se ha tratado el criterio de parada de *performance*, ya que, en estudios anteriores con una base de datos similar siempre se obtuvo que los mejores resultados se obtenían cuando performance era igual a 1^{-6}. Por tanto, este criterio se dejará con ese valor y además será el criterio de parada a utilizar en los experimentos en los cuales no estudiemos otros criterios de parada.

El procedimiento para todos los descriptores es el mismo y se detalla a continuación:

- Primero, teniendo por defecto el criterio de parada de entrenamiento de la red neuronal anteriormente mencionado, estudiaremos cual es la mejor función para entrenar nuestra red neuronal de entre cuatro posibles. Matlab posee muchas funciones para realizar esta tarea, pero la mayoría de ellas tienen un coste computacional muy alto o directamente insalvable. Por tanto, se eligieron las funciones cgp (Polak-Ribiére Conjugate Gradient), cgf (Fletcher-Powell Conjugate Gradient), cgb (Conjugate Gradient with Powell/Beale Restarts) y scg (Scaled Conjugate Gradient), las cuales en un estudio preliminar dieron resultados aceptables. Para este experimento se hace uso también del número de neuronas y capas ocultas que tienen que formar la red neuronal y de la proporción de datos de entrenamiento. Estas dos últimas variables se dejarán por defecto en 2 cada de 10 neuronas cada una y en una disposición de datos de entrenamiento de 85% para entrenamiento, 10% para validación y 5% para test.

- En segundo lugar, calcularemos, utilizando la mejor función de entrenamiento obtenida del primer experimento, el número óptimo de capas ocultas y de neuronas que debe contener cada capa de nuestra red neuronal. Este experimento se realiza dejando por defecto el criterio de parada, y con la misma disposición de datos de entrenamiento del experimento anterior. Se va entrenando la red neuronal variando el número de capas ocultas y de neuronas, estas pueden oscilar entre 2 y 5 capas ocultas en las que hay combinaciones de neuronas que varían de 5 en 5 entre los valores de 10 neuronas mínimo y 20 neuronas máximo. También hay que tener en cuenta que, en este experimento se ha realizado una preselección de los mejores casos entre los que tienen el mismo número de capas ocultas y después se han comparado estos entre sí. Esta preselección ha sido necesaria debido a la gran cantidad de datos que se obtuvieron en esos experimentos, los cuales no se podían mostrar de forma óptima en una sola gráfica

- Después se hace un estudio de la proporción de datos de entrenamiento entrenando la red neuronal y calculando el error de localización para disposiciones como 85% para entrenamiento 10% para validación y 5% para test entre otras. La función de entrenamiento será la calculada previamente y el número de capas y neuronas se mantiene por defecto al igual que en el experimento anterior. Una vez calculadas se comparan mediante una gráfica y se elige la mejor para ser utilizada en los siguientes experimentos.

- Más tarde, con la configuración de parámetros de entrenamiento anteriores se analizan diferentes criterios de parada al que utilizamos por defecto. Uno de estos es el parámetro *epochs,* con el cual dándole valores entre 100 y 10000 calculamos el error de localización para este criterio de parada.

- A continuación, se realiza el mismo experimento anterior, pero esta vez con el parámetro *validation* al que se le dan valores entre 10 y 10000. Este parámetro consiste en el número de *epochs* consecutivas en las que aumenta el error de validación, el cuál es calculado tras cada *epoch*.

Cabe destacar, que todos los experimentos de este capítulo son reflejados con sus resultados en gráficas, en las cuales se miden las variables tiempo y error. El error corresponde al error de localización, es decir, la diferencia entre las coordenadas estimadas por la red neuronal de regresión y la posición real del robot. Mientras que el tiempo corresponde al tiempo que tarda en entrenarse la red neuronal más el tiempo que tarda esta red en estimar las coordenadas de la posición del robot, sumándole el tiempo de cálculo del error de localización.

6.1. Experimentos con el descriptor basado en la CNN ALEXNET usando *day* como imágenes de prueba

6.1.1. Resultados sin *data augmentation*

Todos los resultados conseguidos en los experimentos de esta sección se realizan sin el uso de la técnica de *data augmentation* en la base de datos de entrenamiento de la red neuronal de regresión.

6.1.1.1. Algoritmos de entrenamiento de la red neuronal

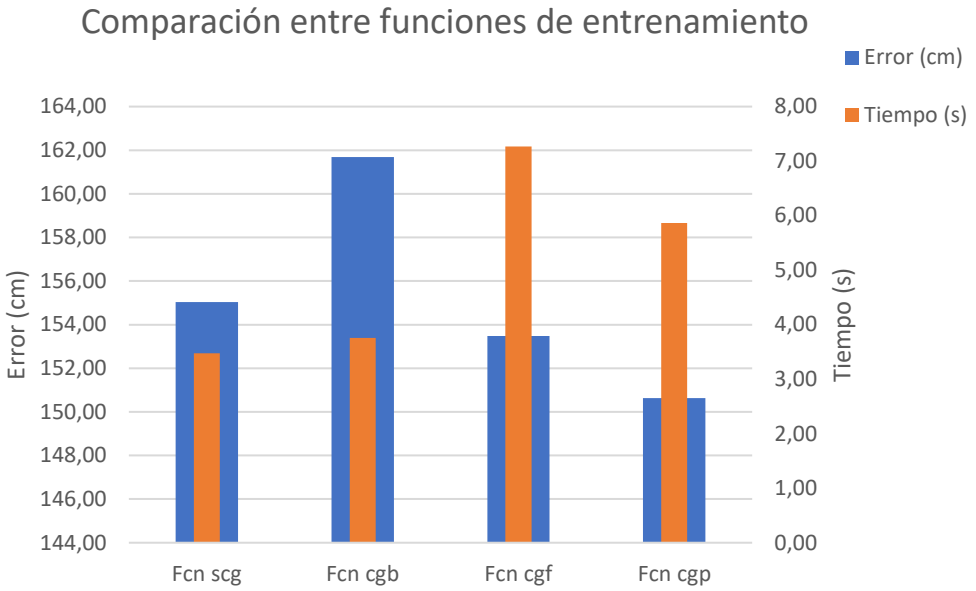


Figura 13. Comparativa entre funciones de entrenamiento con el descriptor basado en la CNN ALEXNET, usando *day* como imágenes de testeo y sin *data augmentation*.

En la figura 13 podemos observar que, aunque la función scg sea la que ha obtenido el menor tiempo de cómputo, se ha elegido la función cgp para el entrenamiento de la red, ya que su error de posición es mucho más bajo que el resto y la diferencia de tiempos no supone un problema. Por tanto, se ha utilizado esta función para los próximos experimentos de esta sección.

6.1.1.2. Número de capas ocultas y de neuronas

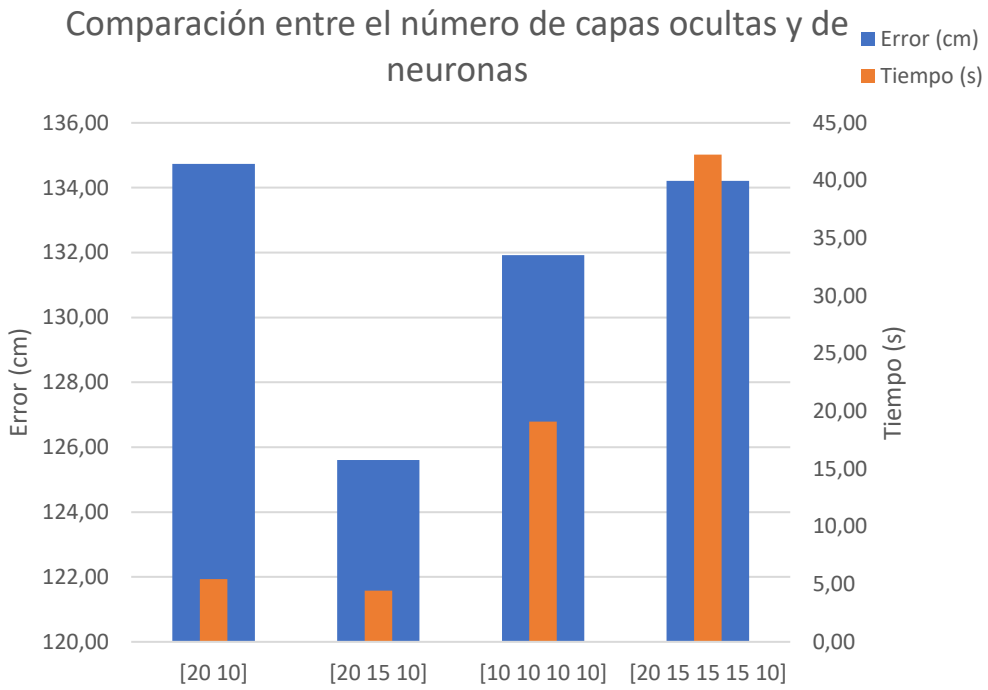


Figura 14. Comparativa entre número de capas ocultas y de neuronas con el descriptor basado en la CNN ALEXNET, usando *day* como imágenes de test y sin *data augmentation*.

Para el experimento de capas ocultas y neuronas usando el descriptor ALEXNET con *day* como imágenes de prueba, podemos observar que el número de capas ocultas, exceptuando las redes que tienen 2 y 3 capas en las cuales la diferencia de tiempo es mínima, afecta en gran manera al coste computacional de la red neuronal, cuantas más capas ocultas le pongamos a nuestra red más tiempo le llevará el calculo del error de localización. También se puede observar que el error mínimo de localización se alcanza con 3 capas ocultas y 20, 15 y 10 neuronas respectivamente.

6.1.1.3. Proporción de datos de entrenamiento

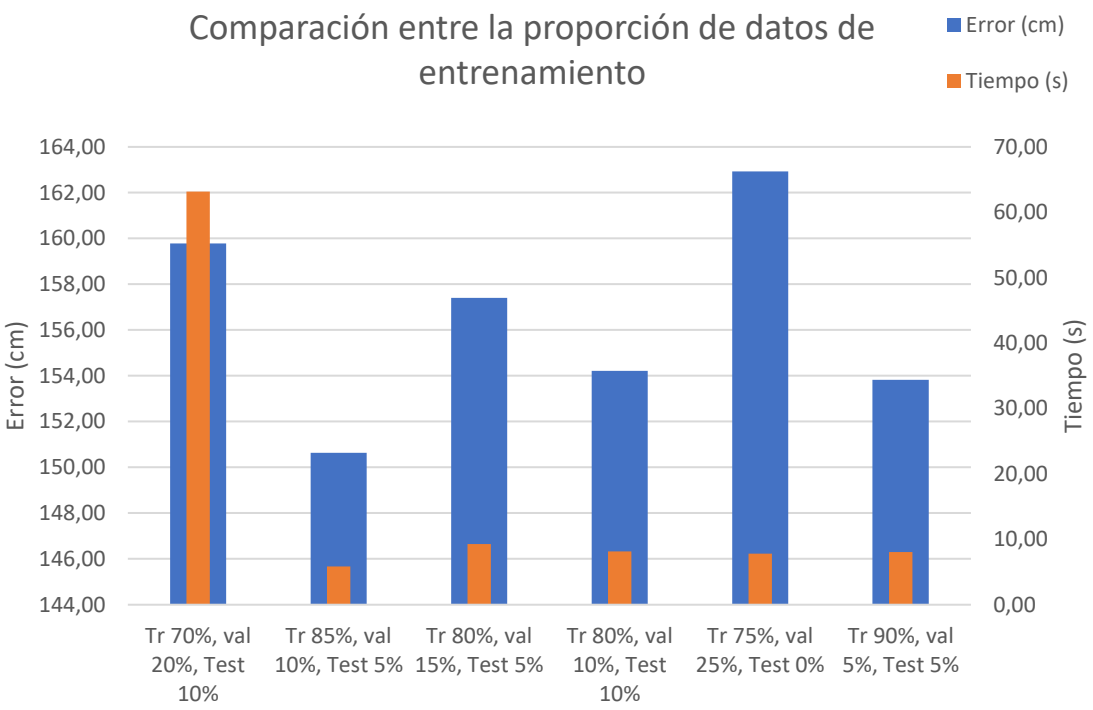


Figura 15. Comparativa entre proporción de datos de entrenamiento con el descriptor basado en la CNN ALEXNET, usando *day* como imágenes de test y sin *data augmentation*.

En los resultados de este experimento, con el descriptor ALEXNET y day como imágenes de prueba, se aprecia que las proporciones de datos de entrenamiento con un menor porcentaje de imágenes usadas para entrenar tienen un mayor tiempo de cómputo y error de localización que el resto. El error y tiempo mínimo lo consigue la proporción de datos de entrenamiento que había sido elegida como proporción por defecto, es decir, 85% para entrenamiento, 10% para validación y 5% para test.

6.1.1.4. Parámetro *epochs*

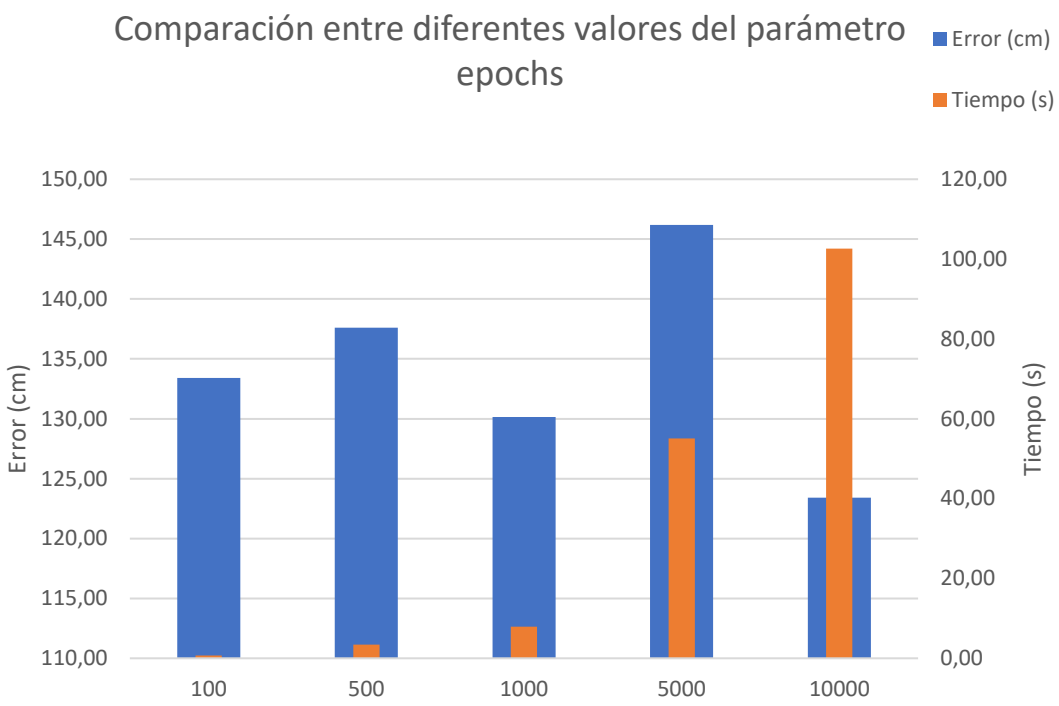


Figura 16. Comparativa entre diferentes valores del parámetro *epochs* con el descriptor basado en la CNN ALEXNET, usando *day* como imágenes de testeo y sin *data augmentation*.

En la figura 16 se puede observar que a medida que aumenta el número de *epochs* también lo hace el tiempo de cálculo. Aunque se obtuvo en 10000 *epochs* el mejor resultado en error de todos los experimentos hasta el momento, también se consiguió el peor resultado en tiempo y así es justificado el desuso de esta configuración, ya que, conseguimos disminuir el error de localización 2 cm respecto al mejor valor del experimento de capas ocultas y neuronas, en el cual se utiliza el criterio de parada por defecto, pero aumentamos el tiempo de entrenamiento casi 25 veces.

6.1.1.5. Parámetro *validation*

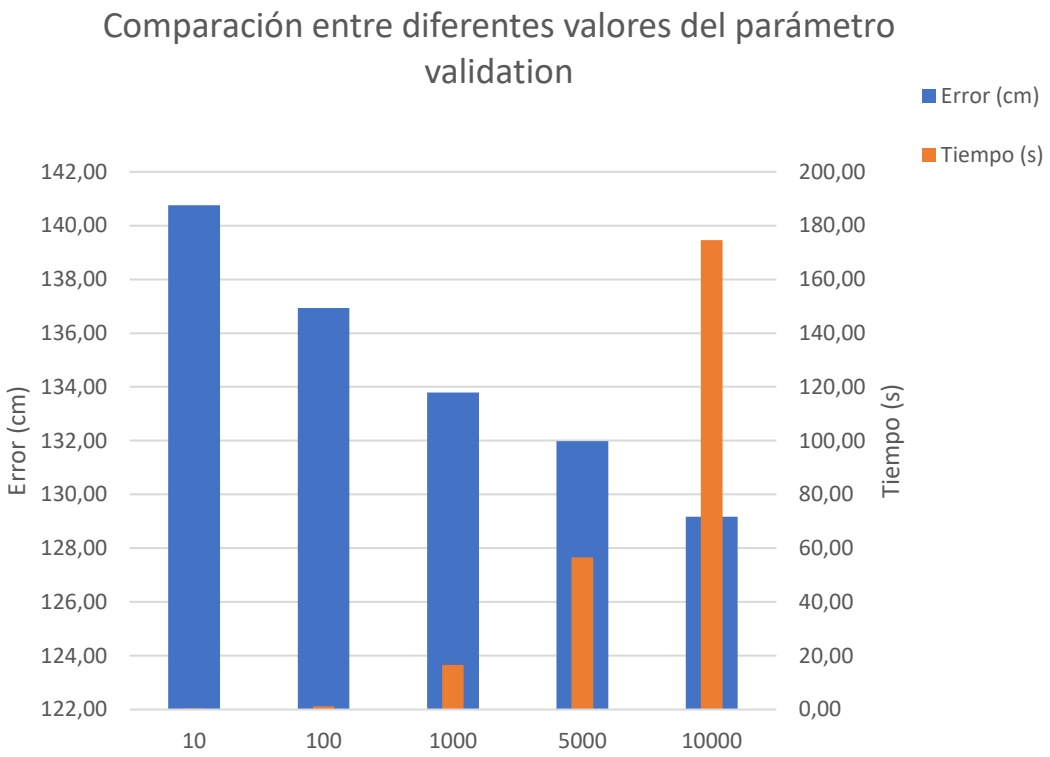


Figura 17. Comparativa entre diferentes valores del parámetro *validation* con el descriptor basado en la CNN ALEXNET, usando *day* como imágenes de test y sin *data augmentation*.

En el experimento del parámetro *validation* con el descriptor ALEXNET y con *day* como imágenes de prueba, se puede apreciar que el error de localización disminuye de forma progresiva cuanto mayor es el valor de *validation,* pero también se incrementa de la misma manera el tiempo de cómputo. Este criterio de parada, aún con su mejor configuración no consigue mejorar ni en error ni por supuesto en tiempo al mejor caso de los anteriores experimentos realizados con otros criterios de parada.

6.1.2. Resultados con *data augmentation*

6.1.2.1. Algoritmos de entrenamiento de la red neuronal

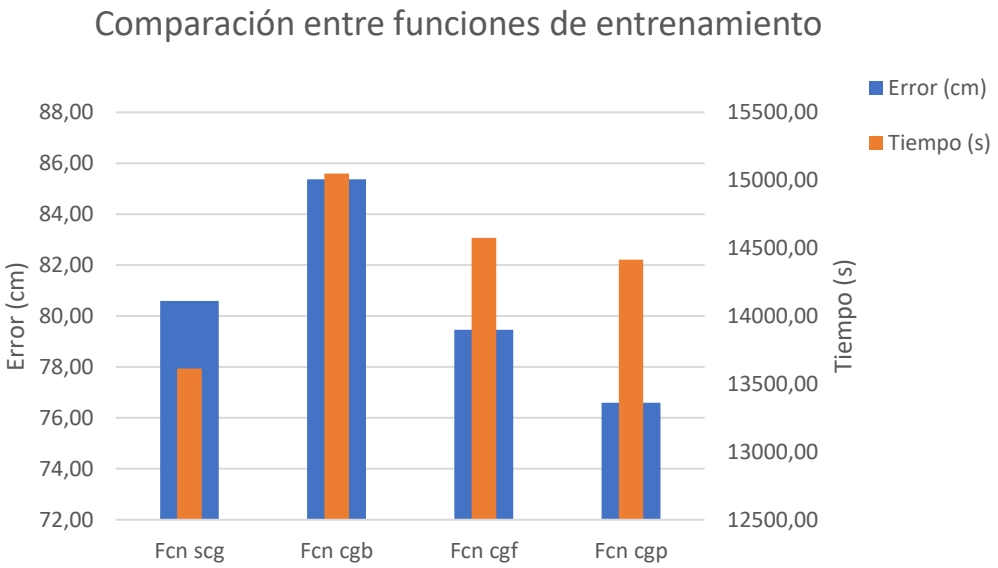


Figura 18. Comparativa entre funciones de entrenamiento con el descriptor ALEXNET, usando *day* como imágenes de prueba y con *data augmentation*.

En el experimento de la elección del algoritmo de entrenamiento con el descriptor ALEXNET, con day como imágenes de prueba y usando la técnica de *data augmentation*, podemos observar que las mejores funciones son la scg y cgp como ocurría también en el mismo experimento sin *data augmentation*. Aunque la función scg consigue ser 1000 segundos más rápida, se ha elegido la función cgp para los próximos experimentos debido a que consigue un error de localización menor y la diferencia de tiempo es asumible.

6.1.2.2. Número de capas ocultas y de neuronas

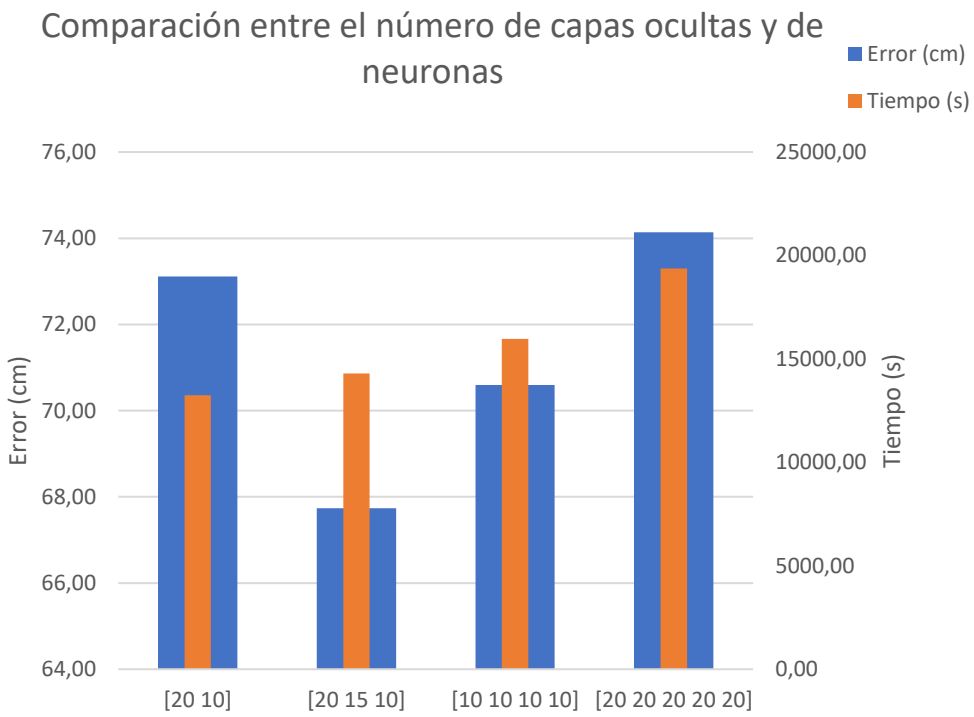


Figura 19. Comparativa entre número de capas ocultas y de neuronas utilizando el descriptor ALEXNET, con *day* como imágenes de test y con *data augmentation*.

En el experimento de variación del número de capas ocultas y neuronas utilizando el descriptor ALEXNET con *day* como imágenes de prueba y con *data augmentation*, se puede apreciar que el error disminuye drásticamente en redes neuronales con 3 capas ocultas y aumenta progresivamente al añadirle más capas. También podemos ver que el tiempo de cálculo del entrenamiento y la localización aumenta conforme más capas ocultas tenga la red neuronal en cuestión. Por tanto, la mejor disposición de capas ocultas y neuronas es 3 capas de 20, 15 y 10 neuronas respectivamente.

6.1.2.3. Proporción de datos de entrenamiento

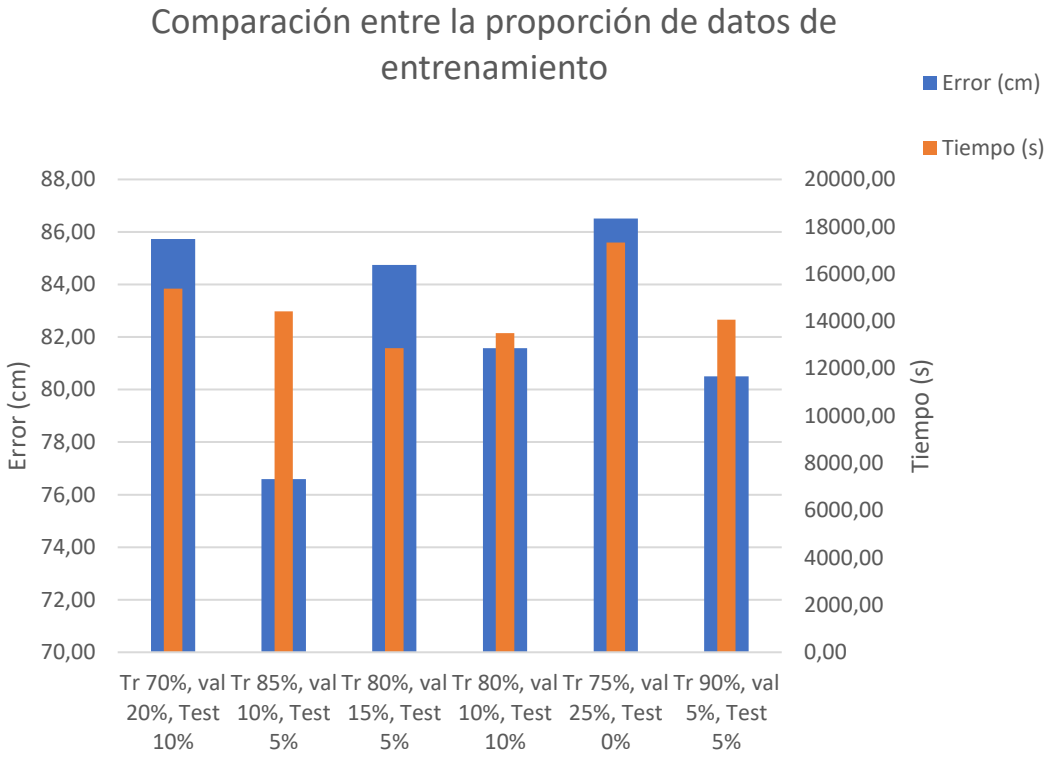


Figura 20. Comparativa entre proporción de datos de entrenamiento con el descriptor ALEXNET, usando *day* como imágenes de prueba y con *data augmentation*.

Para el caso de proporción de datos de entrenamiento usando el descriptor ALEXNET, con *day* como imágenes test y utilizando la técnica de *data augmentation,* se tiene que, los tiempos de cálculo *en* proporciones de entrenamiento donde hay un 80% o más imágenes que se utilizan para entrenamiento, son menores que los demás. En cuanto al error de localización observamos que, la proporción elegida por defecto de 85% para entrenamiento, 10% para validación y 5% para test, es la mejor con una diferencia de 4 cm con la segunda mejor proporción. Esta proporción también obtuvo el mejor resultado en el experimento sin *data augmentation*.

6.1.2.4. Parámetro *epochs*

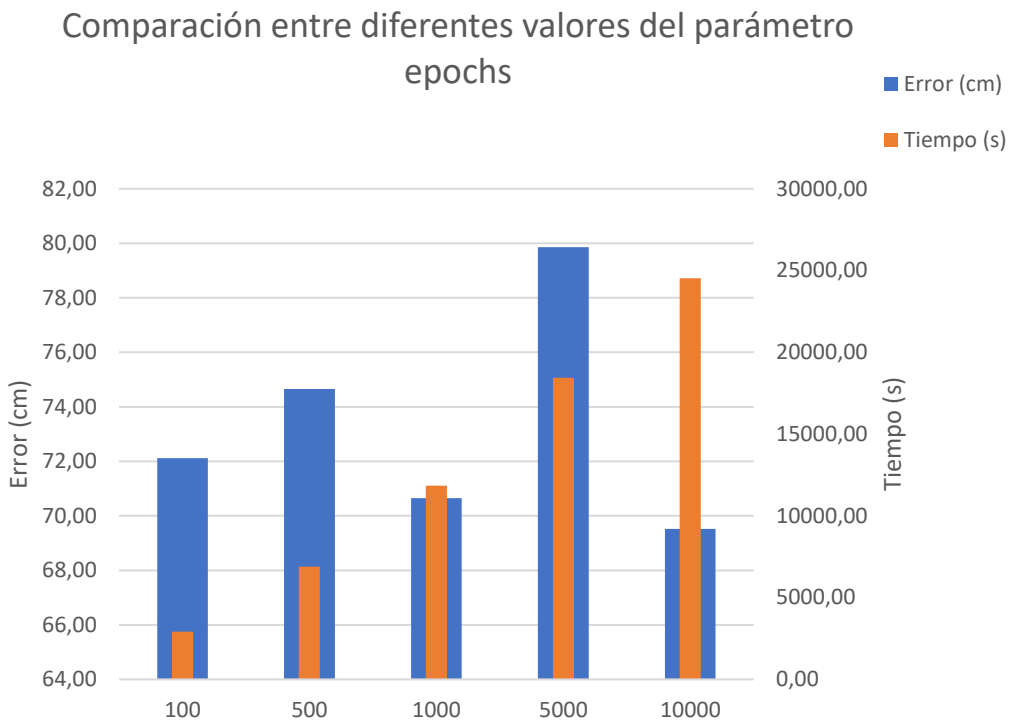


Figura 21. Comparativa entre diferentes valores del parámetro *epochs* utilizando el descriptor ALEXNET, con *day* como imágenes de prueba y con *data augmentation.*

En este experimento de variación del parámetro *epochs*, cuyos resultados se encuentran en la figura 21, se puede observar que no existe una tendencia clara en el error de localización, siendo este mínimo cuando el parámetro *epochs* alcanza el valor de 10000.

6.1.2.5. Parámetro *validation*

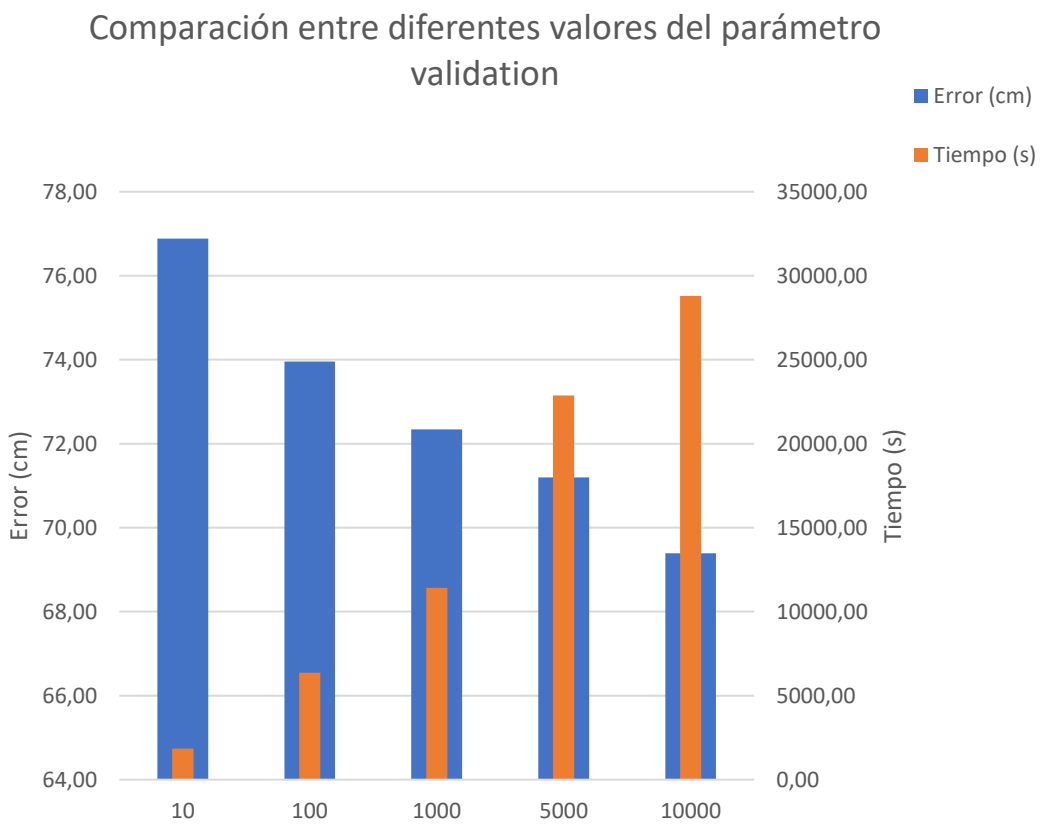


Figura 22. Comparativa entre diferentes valores del parámetro *validation* utilizando el descriptor ALEXNET, con *day* como imágenes de prueba y con data *augmentation.*

Para el caso de variación del parámetro *validation* cuando se utiliza el descriptor holístico ALEXNET, *day* como imágenes de test y con la técnica de *data augmentation* tenemos que, el error de localización disminuye de una forma constante a medida que se aumentan el número de validaciones llegando a su mínimo en 10000. Por otro lado, se puede ver en la figura 22 que el coste computacional aumenta a medida que se incrementan el número de validaciones.

6.1.3. Comparaciones entre los resultados con *data augmentation* y sin *data augmentation*

En este apartado se analizan los resultados de las mejores configuraciones del descriptor basado en la CNN ALEXNET usando *day* como imágenes de testeo con y sin *data augmentation.*

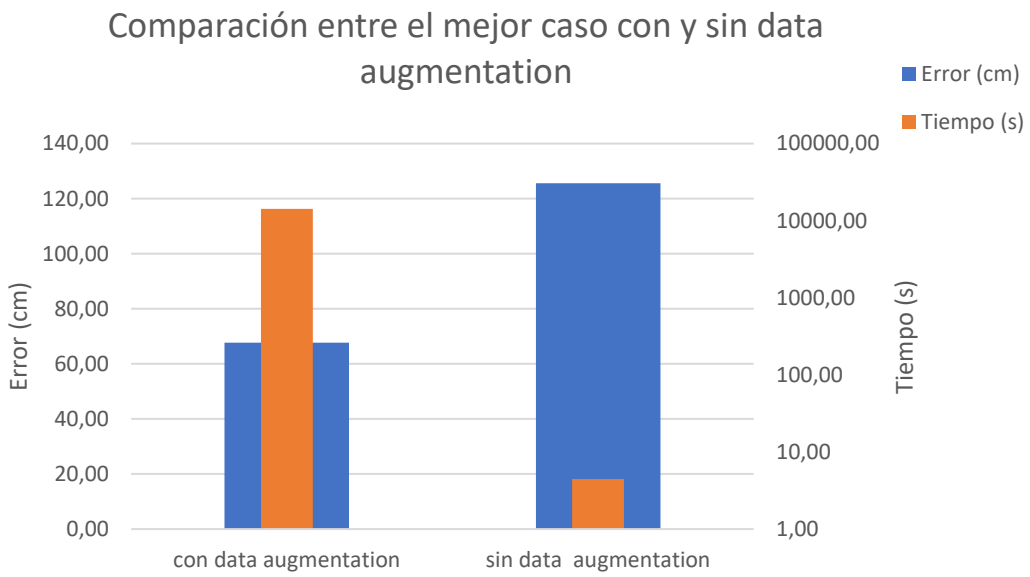


Figura 23. Comparativa entre las mejores configuraciones utilizando el descriptor ALEXNET, con *day* como imágenes de prueba, con y sin *data augmentation*.

La mejor configuración obtenida con el uso del descriptor ALEXNET, *day* como imágenes de test y sin *data augmentation* es la siguiente: se ha elegido el criterio de parada por defecto, es decir, el parámetro performance, la función de entrenamiento cgp, una disposición de 3 capas ocultas de 20, 15 y 10 respectivamente y una proporción de datos de entrenamiento de 85% para entrenamiento, 10% para validación y 5% para test.

La mejor configuración del mismo experimento con *data augmentation* ha sido igual a la configuración sin *data augmentation*. Por tanto, la comparación de la figura 23 se realiza con la misma configuración con y sin *data augmentation*. En esa figura se puede observar que al usar la técnica de

aumento de datos el coste computacional es mucho mayor que sin el uso de esta. Sin embargo, el error de localización disminuye hasta ser prácticamente la mitad del error sin usar aumento de datos.

6.2. Experimentos con el descriptor basado en la CNN ALEXNET usando *doorlit* como imágenes de prueba

6.2.1. Resultados sin *data augmentation*

6.2.1.1. Algoritmos de entrenamiento de la red neuronal

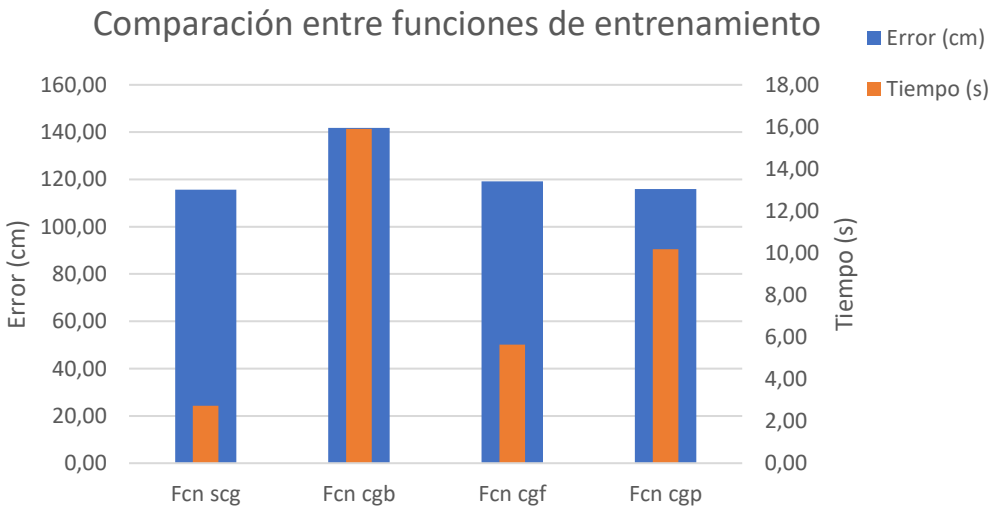


Figura 24. Comparativa entre funciones de entrenamiento con el descriptor ALEXNET, con *doorlit* como imágenes de prueba y sin *data augmenta*tion.

En este experimento se prueban diferentes funciones de entrenamiento para la red neuronal, teniendo ALEXNET como descriptor, con *doorlit* como imágenes de prueba y sin aumento de datos. Como se puede observar en la figura 24, tanto las funciones scg, cgf y cgp están muy parejas en cuanto a error de localización se refiere. Sin embargo, se eligió la función scg, la cual además de tener un error ligeramente más pequeño también es la función de entrenamiento que consigue un coste computacional menor al resto.

6.2.1.2. Número de capas ocultas y de neuronas

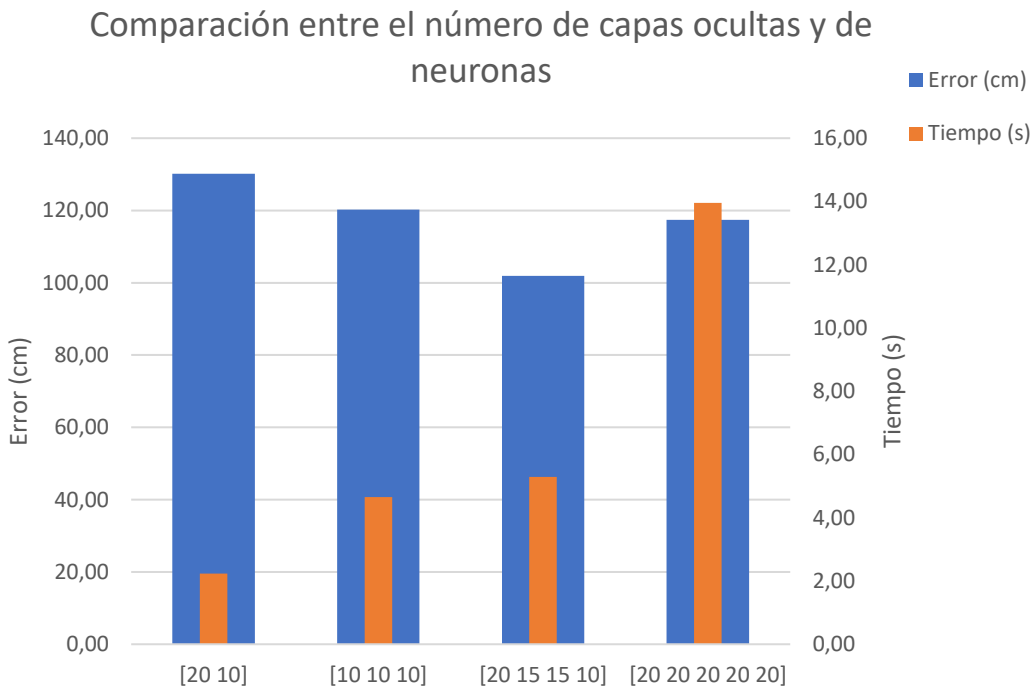


Figura 25. Comparativa entre el número de capas ocultas y neuronas utilizando el descriptor ALEXNET, usando *doorlit* como imágenes de test y sin *data augmentation*.

En el caso de la elección de número de capas ocultas y neuronas con el descriptor ALEXNET, con *doorlit* como imágenes de prueba y sin *data augmentation*, se puede observar que a medida que aumenta el número de capas ocultas y neuronas se incrementa el tiempo de cómputo. También se aprecia que el error disminuye cuantas más capas ocultas se tienen en la red neuronal hasta llegar a 4 capas, a partir de este número de capas este error vuelve a incrementarse. Por tanto, se elegirá la disposición de 4 capas con 20, 15, 15 y 10 neuronas respectivamente debida a la gran diferencia de error y un tiempo de cálculo razonable.

6.2.1.3. Comparación entre proporción de datos de entrenamiento

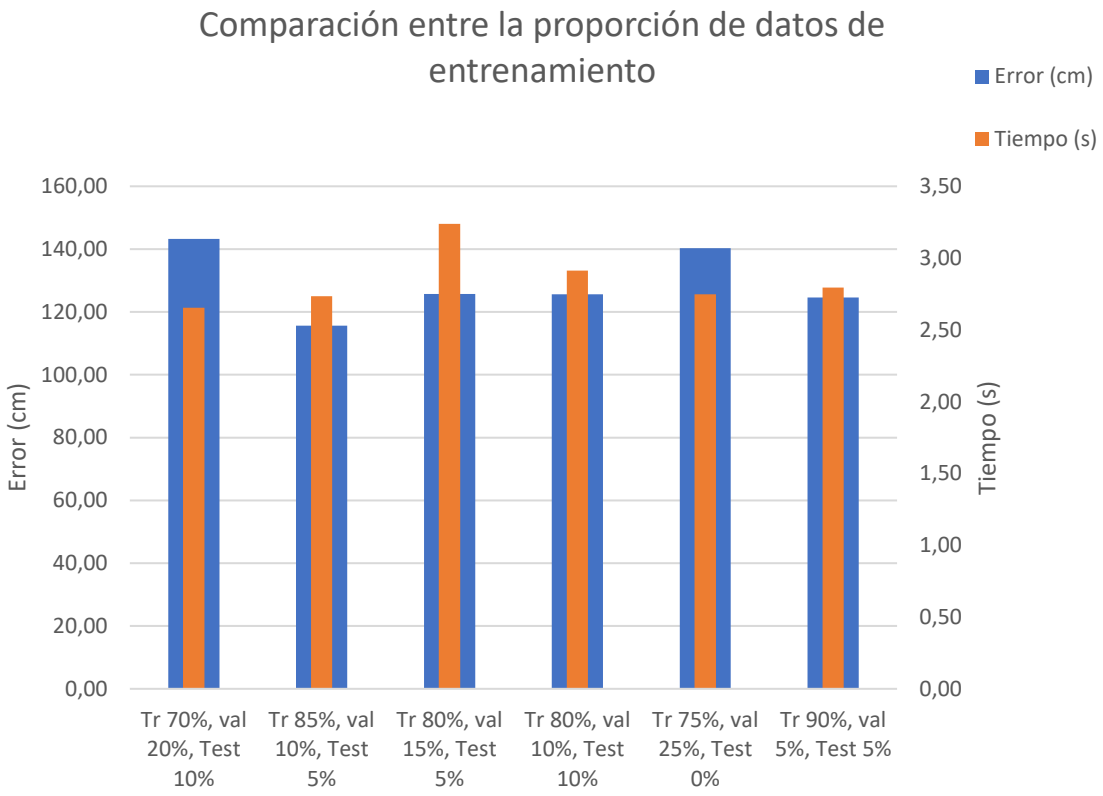


Figura 26. Comparativa entre proporciones de datos de entrenamiento con el descriptor ALEXNET, usando *doorlit* como imágenes de prueba y sin *data augmentation*.

En este experimento de proporciones de datos de entrenamiento utilizando el descriptor ALEXNET, con *doorlit* como imágenes de prueba y sin aumento de datos, se puede observar que el coste computacional es similar en todas las proporciones de datos siendo en la proporción 85% para entrenamiento, 10% para validación y 5% para test ligeramente inferior. Además, esa configuración consigue el error de localización más pequeño y, por tanto, será la proporción elegida.

6.2.1.4. Parámetro *epochs*

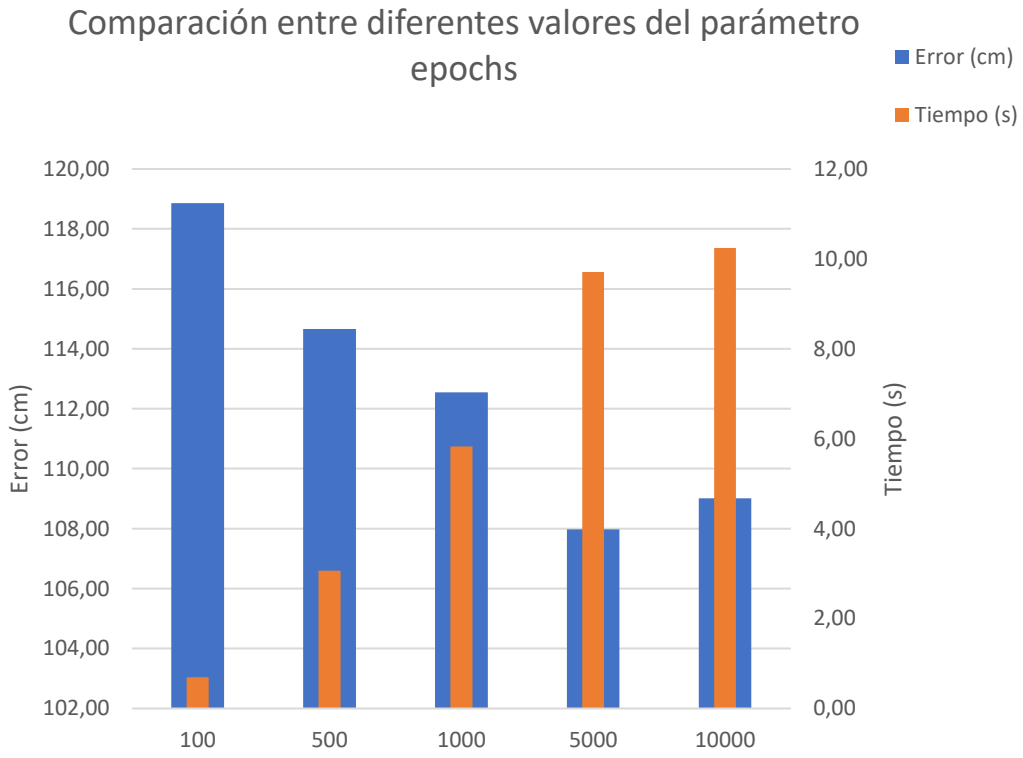


Figura 27. Comparativa entre diferentes valores del parámetro *epochs* utilizando el descriptor ALEXNET, con *doorlit* como imágenes de test y sin *data augmentation*.

En el caso de la variación del parámetro *epochs* cuando se utiliza el descriptor ALEXNET, *doorlit* como imágenes de prueba y no se usa aumento de datos, se puede apreciar que, como en todos los experimentos realizados hasta el momento con este parámetro, el coste computacional aumenta de forma progresiva a medida que se incrementan el número de *epochs* hasta llegar a su máximo en 10000 *epochs.* En cuanto al error de localización, disminuye conforme aumenta el número de epochs hasta llegar a un mínimo en 5000 epochs, y a partir de este valor vuelve a incrementarse el error de posición.

6.2.1.5. Parámetro *validation*

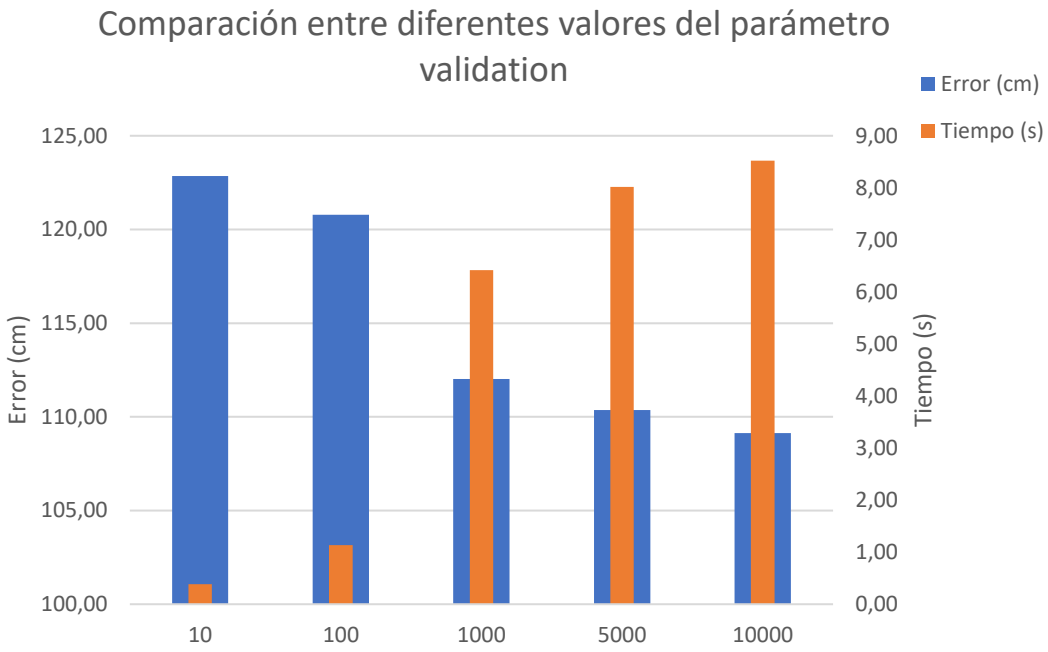


Figura 28. Comparativa entre diferentes valores del parámetro *validation* utilizando el descriptor ALEXNET, con *doorlit* como imágenes de test y sin *data augmentation*.

En el experimento del parámetro *validation* con el descriptor ALEXNET, con *doorlit* como imágenes de prueba y sin utilizar aumento de datos, podemos observar que el error de localización disminuye a medida que aumenta el número de validaciones, llegando al valor mínimo en 10000 validaciones. Por otro lado, el tiempo de cómputo del entrenamiento y el cálculo del error de localización aumenta cuanto mayor es el valor del parámetro validation.

6.2.2. Resultados con *data augmentation*

6.2.2.1. Algoritmos de entrenamiento de la red neuronal

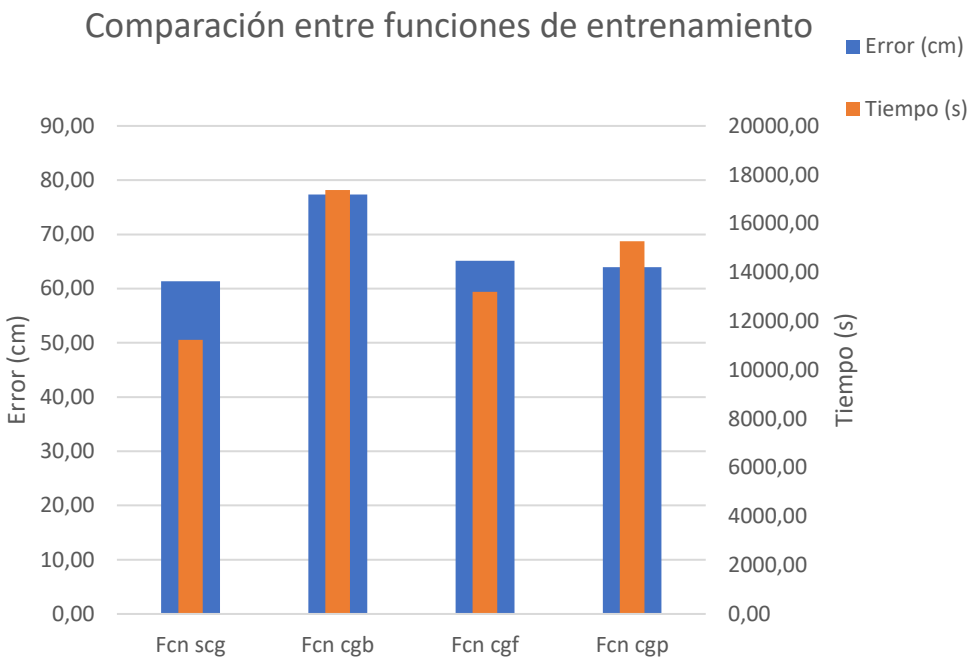


Figura 29. Comparativa entre funciones de entrenamiento de usando el descriptor ALEXNET, con *doorlit* como imágenes de prueba y con *data augmentation*.

En este experimento se prueban diferentes funciones de entrenamiento para la red neuronal, teniendo ALEXNET como descriptor, con *doorlit* como imágenes de prueba y con aumento de datos. En la figura 29, podemos observar que la mejor función de entrenamiento de nuestra red neuronal es la scg, al igual que ocurría en el mismo experimento sin la utilización de *data augmentation.* La función scg mejora al resto tanto en error de localización como en tiempo de cómputo.

6.2.2.2. Número de capas ocultas y de neuronas

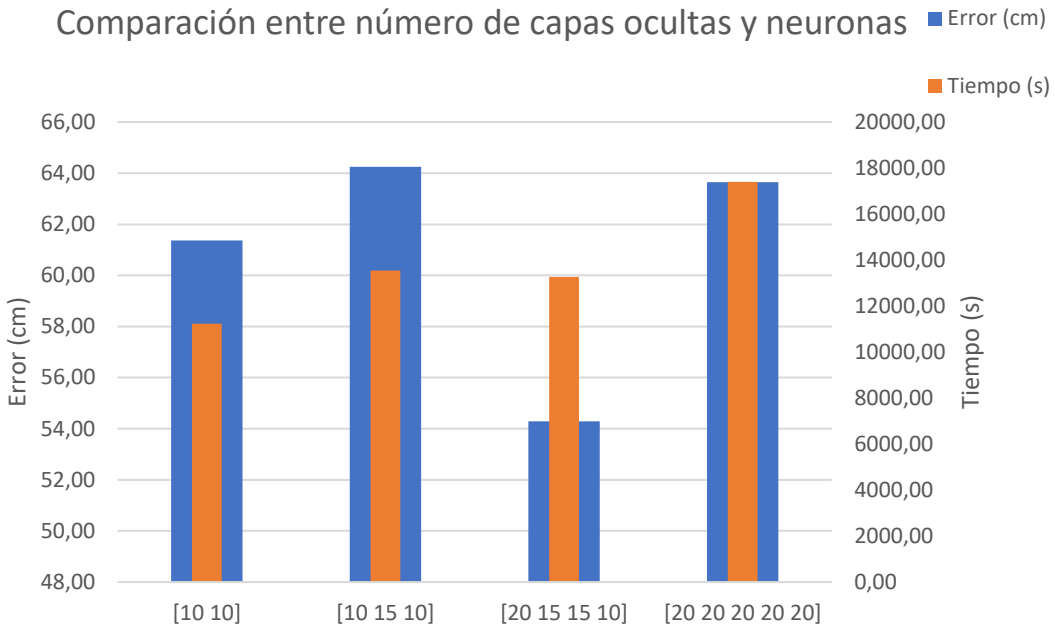


Figura 30. Comparativa entre el número de capas ocultas y de neuronas utilizando el descriptor ALEXNET, con *doorlit* como imágenes de test y con *data augmentation*.

En el caso de la elección de número de capas ocultas y neuronas con el descriptor ALEXNET, con *doorlit* como imágenes de prueba y con *data augmentation*, se puede apreciar que, en general el coste computacional aumenta cuanto mayor es el número de capas de nuestra red neuronal, a excepción de la disposición de 4 capas ocultas y 20, 15, 15, 10 neuronas respectivamente que consigue un tiempo ligeramente inferior a las redes neuronales con 3 capas ocultas. Por tanto, se eligió la disposición de 4 capas para la realización de los experimentos correspondientes.

6.2.2.3. Proporción de datos de entrenamiento

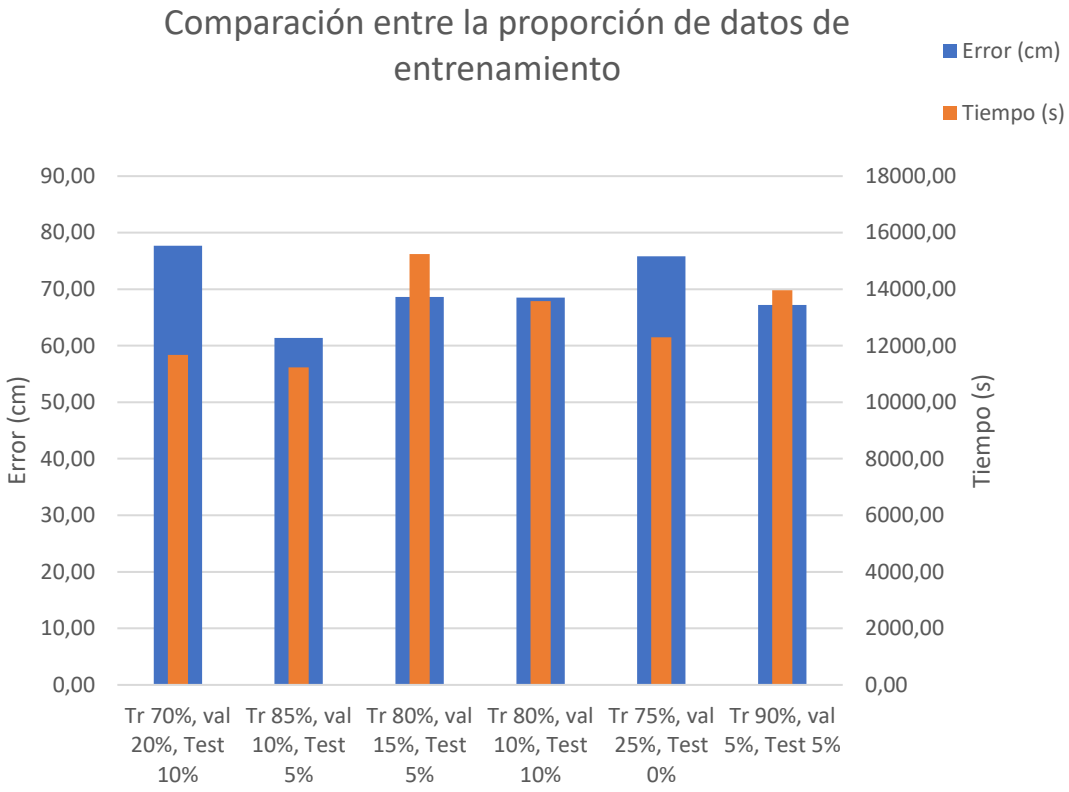


Figura 31. Comparativa entre proporciones de datos de entrenamiento utilizando el descriptor ALEXNET, con *doorlit* como imágenes de entrada y con *data augmentation*.

En este experimento de proporciones de datos de entrenamiento utilizando el descriptor ALEXNET, con *doorlit* como imágenes de prueba y utilizando la técnica de aumento de datos, se observa que, tanto el tiempo mínimo de cómputo como el error mínimo de localización se consigue con la proporción de datos de 85% para entrenamiento, 10% para validación y 5%. Cabe destacar, que este resultado es igual que el del mismo experimento sin la utilización del aumento de datos.

6.2.2.4. Parámetro *epochs*

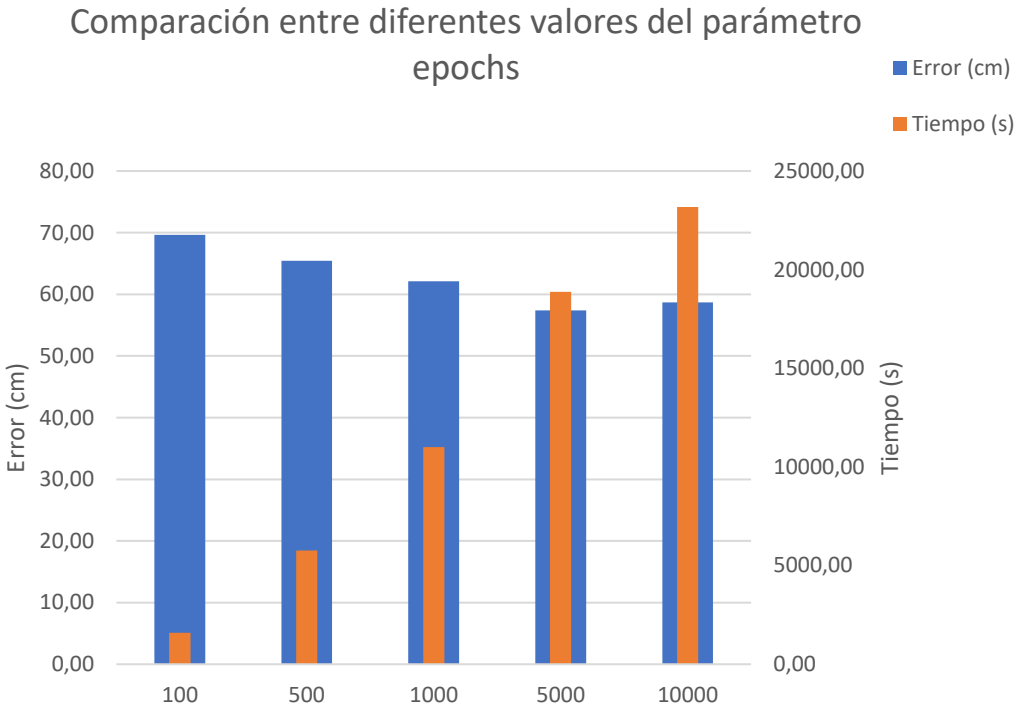


Figura 32. Comparativa entre distintos valores del parámetro *epochs* utilizando el descriptor ALEXNET, con *doorlit* como imágenes de prueba y con *data augmentation*.

En el caso de la variación del parámetro *epochs* cuando se utiliza el descriptor ALEXNET, *doorlit* como imágenes de prueba y no se usa aumento de datos, se puede apreciar que, al aumentar el número de *epochs* el tiempo de cómputo incrementa, mientras que el error de localización decrece hasta llegar a los 5000 *epochs*, a partir de valores por encima del anterior el error ya no se ve disminuido.

6.2.2.5. Parámetro *validation*

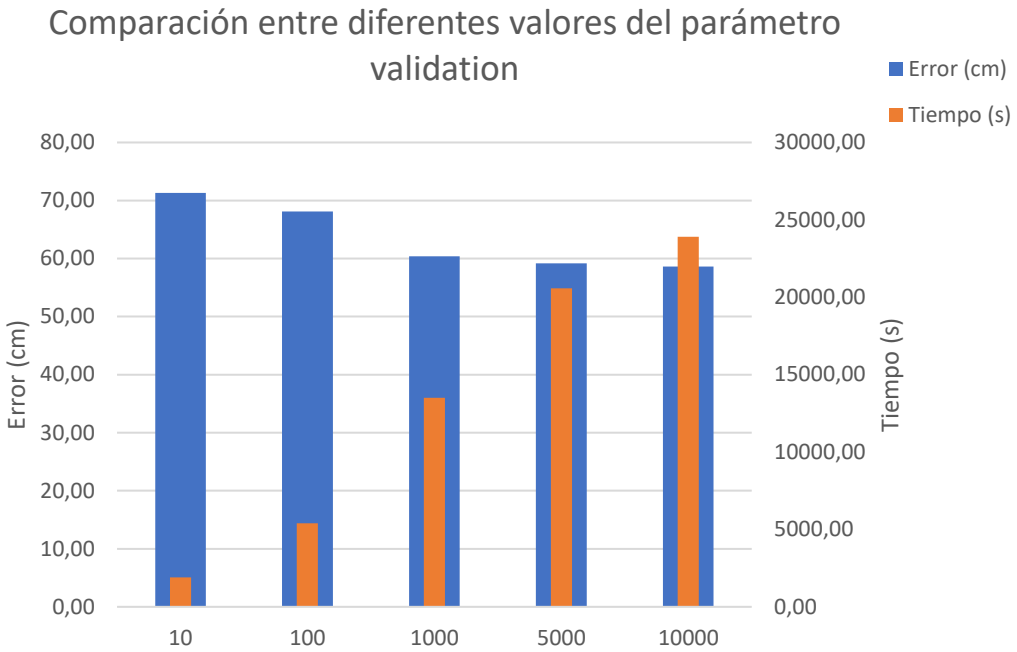


Figura 33. Comparativa entre distintos valores del parámetro *validation* utilizando el descriptor ALEXNET, con *doorlit* como imágenes de prueba y con *data augmentation*.

En el experimento del parámetro *validation* con el descriptor ALEXNET, con *doorlit* como imágenes de prueba y utilizando aumento de datos, podemos observar que el error de localización decrece progresivamente hasta alcanzar su mínimo cuando las validaciones son igual a 10000. Por otro lado, el tiempo de cómputo aumenta cuanto más se incrementa el valor del parámetro validation.

6.2.3. Comparaciones entre los resultados con *data augmentation* y sin *data augmentation*

En esta sección se analizan los resultados de las mejores configuraciones del descriptor basado en la CNN ALEXNET usando *doorlit* como imágenes de testeo con y sin *data augmentation.*

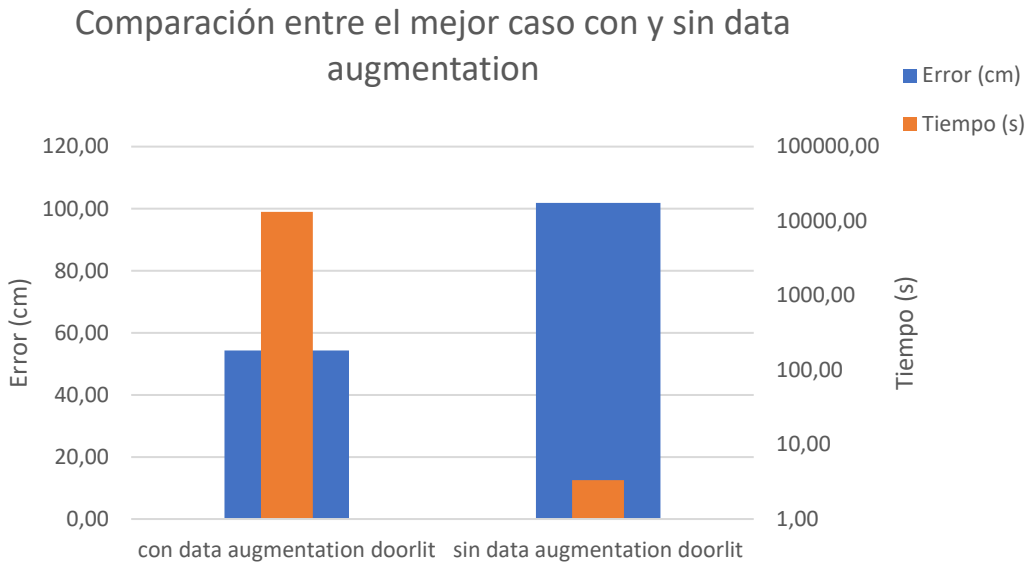


Figura 34. Comparativa entre el mejor caso con y sin *data augmentation* utilizando el descriptor ALEXNET y con *doorlit* como imágenes de test.

La mejor configuración que se obtiene utilizando el descriptor ALEXNET, con *doorlit* como imágenes de test y sin *data augmentation* es la siguiente: se ha elegido el parámetro performance, que es el criterio de parada por defecto, la función de entrenamiento scg, una disposición de 4 capas ocultas de 20, 15, 15 y 10 respectivamente y una proporción de datos de entrenamiento de 85% para entrenamiento, 10% para validación y 5% para test.

La configuración óptima del mismo experimento con *data augmentation* ha sido la misma que la configuración sin *data augmentation*. Por tanto, esta comparación se realiza con la misma configuración con y sin *data augmentation*. En la figura 34 se puede observar, que al usar la técnica de aumento de datos el tiempo de cómputo aumenta en gran cantidad respecto a

los resultados sin el uso de esta. Sin embargo, el error de localización disminuye drásticamente, obteniendo el error de ninguna configuración hasta el momento. Estos resultados son muy similares a los obtenidos en la subsección 6.1.3, por lo que se puede deducir que en líneas generales el *data augmentation* reduce en gran manera el error de localización a costa de un coste computacional muy elevado.

6.3. Comparación entre diferentes configuraciones de imágenes de prueba y de entrenamiento

En este experimento se pretende conocer cómo afectan los cambios de iluminación de la sala a la efectividad de nuestra red neuronal. Esto se realiza mediante la comparación entre dos conjuntos de imágenes de test. Para ello, se recogieron los datos de los mejores casos tanto del entrenamiento de la red neuronal con el descriptor basado en la CNN ALEXNET usando *day* como imágenes de testeo, como del entrenamiento del descriptor basado en ALEXNET utilizando *doorlit* como imágenes de prueba con y sin *data augmentation*, y se comparan entre sí.

Las mejores configuraciones que se obtienen utilizando el descriptor ALEXNET, con *doorlit* como imágenes de test, con y sin *data augmentation* están descritas en la subsección anterior 6.2.3. Mientras que, las configuraciones óptimas que se obtienen utilizando el descriptor ALEXNET, con *day* como imágenes de test, con y sin *data augmentation* se describen en la subsección 6.1.3.

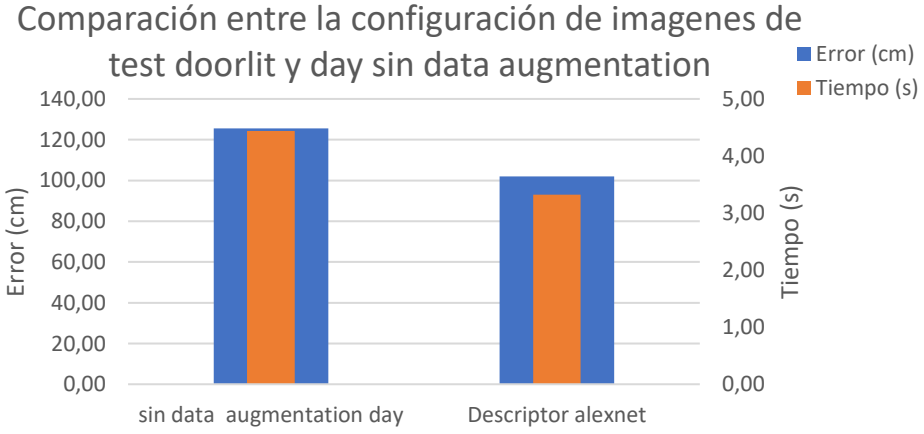


Figura 35. Comparativa entre day y doorlit como imágenes de test utilizando el descriptor ALEXNET sin *data augmentation*.

En la figura 35 se puede apreciar que las redes neuronales entrenadas con todos los grupos de imágenes y probadas con *doorlit* sin utilizar *data augmentation*, consiguen tanto un menor coste computacional como un error de localización notablemente más pequeño respecto a los experimentos realizados con *day*.

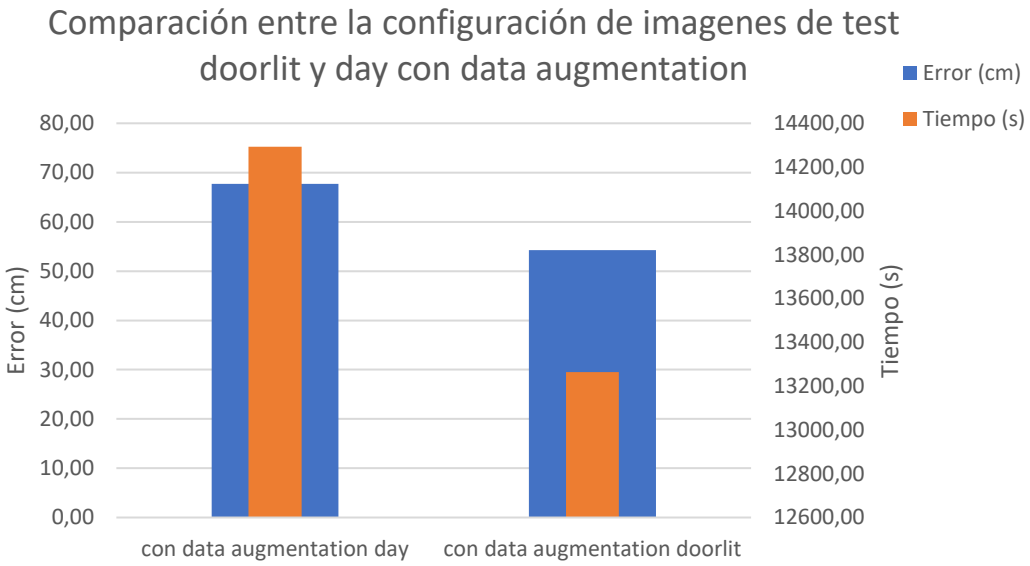


Figura 36. Comparativa entre *day* y *doorlit* como imágenes de test utilizando el descriptor ALEXNET con *data augmentation*.

En el caso de las redes neuronales entrenadas con todos los grupos de imágenes y probadas con *doorlit* utilizando *data augmentation*, se observa lo mismo que en la comparación sin utilizar *data augmentation*, es decir, los experimentos con doorlit tienen un coste computacional más pequeño y un error de posición 13 cm menor.

En este experimento se puede llegar a la conclusión de que nuestra red neuronal de regresión se ve más afectada por imágenes con una mayor luminosidad como son las del conjunto de *day,* ya que la red testeada con ese

conjunto de imágenes consigue un coste computacional y un error de localización mayor tanto utilizando data augmentation como sin la utilización de esta técnica.

Con los resultados obtenidos en estos experimentos se decidió usar la configuración de imágenes con *doorlit* para test y el resto para entrenamiento en los siguientes experimentos debido a su mejora respecto a *day* tanto con *data augmentation* como sin utilizar dicha técnica.

6.4. Experimentos con el descriptor GIST usando *doorlit* como imágenes de prueba

6.4.1. Algoritmos de entrenamiento de la red neuronal

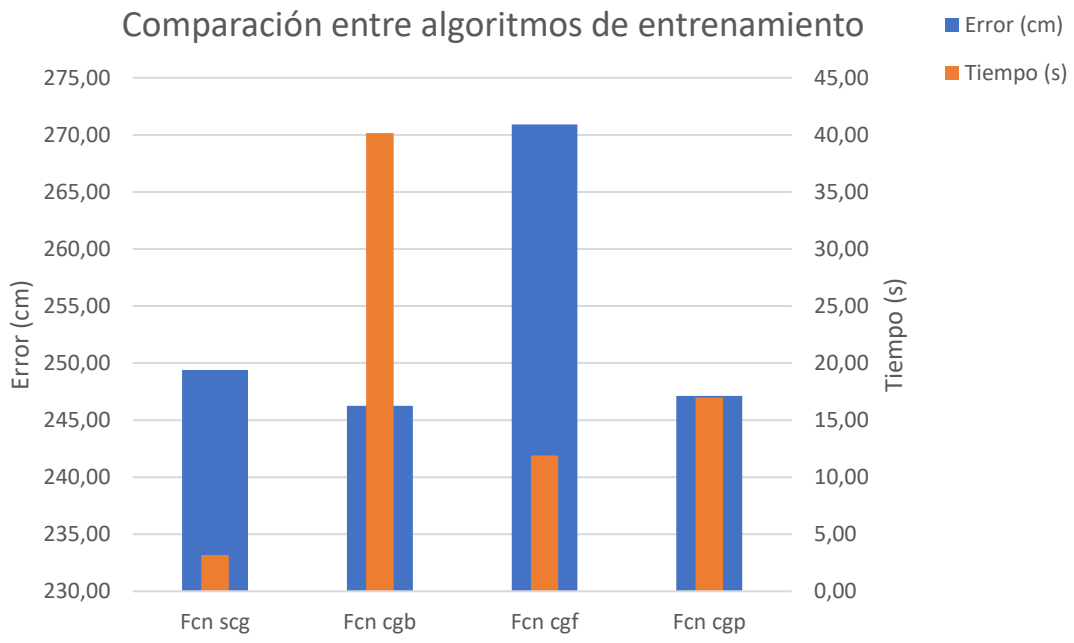


Figura 37. Comparativa entre funciones de entrenamiento de la red neuronal usando el descriptor GIST y con *doorlit* como imágenes de prueba.

En la figura 37, en la cual se muestran los datos del experimento realizado con varias funciones de entrenamiento de la red neuronal utilizando el descriptor GIST y con *doorlit* como imágenes de test, se puede apreciar que,

aunque scg no es la función de entrenamiento con menor error, siendo superada por las funciones cgp y cgb, si es la más rápida y teniendo una diferencia de error tan pequeña nos compensa que la función elegida sea entre 5 y 12 veces más veloz.

6.4.2. Número de capas ocultas y neuronas

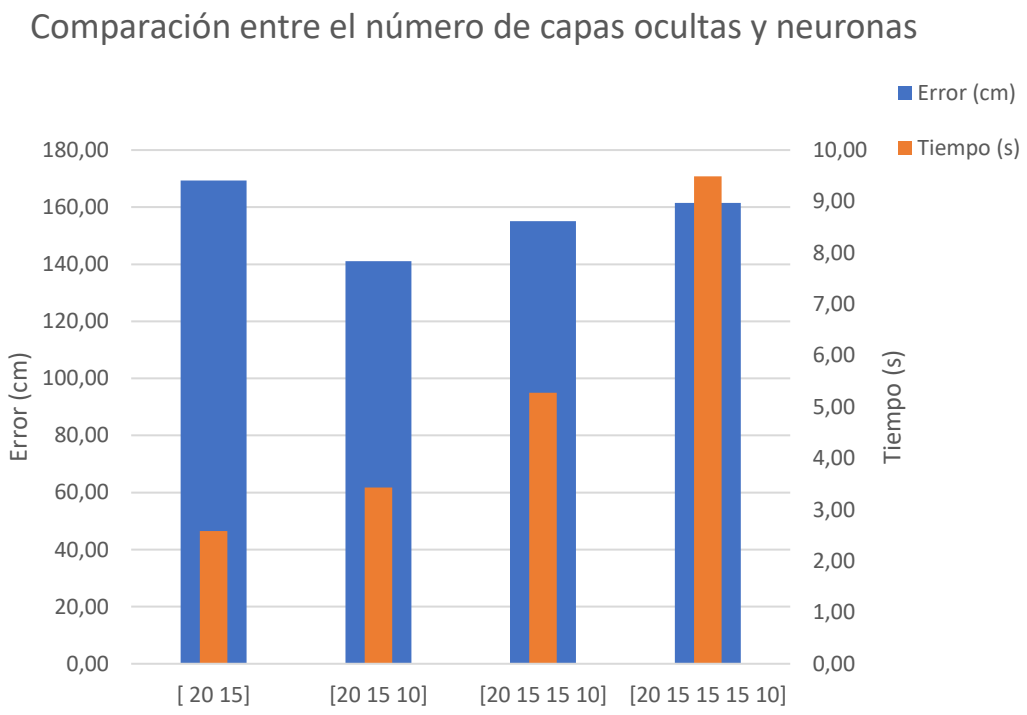


Figura 38. Comparativa entre el número de capas y de neuronas utilizando el descriptor GIST y usando doorlit como imágenes de prueba

En el caso de la elección de número de capas ocultas y neuronas con el descriptor GIST, se puede apreciar que, en general el coste computacional se incrementa notablemente a medida que aumenta el número de capas ocultas de nuestra red neuronal. En cuanto al error de localización, se observa en la figura 38 que el mínimo se alcanza con la disposición de 3 capas ocultas y 20, 15 y 10 neuronas respectivamente. Aunque existe otra disposición con un menor coste computacional la diferencia es ínfima respecto a las ventajas que se adquieren en el error de localización.

6.4.3. Proporción de datos de entrenamiento

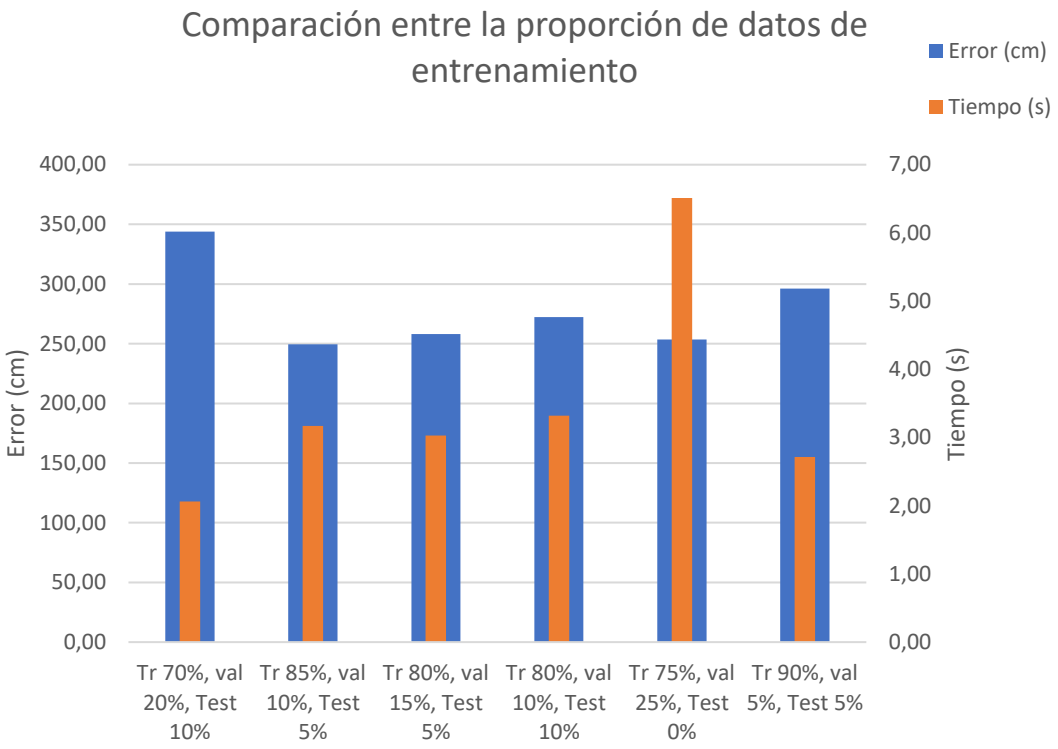


Figura 39. Comparativa entre distintas proporciones de datos de entrenamiento usando el descriptor GIST y con *doorlit* como imágenes de prueba.

En este experimento, cuyos resultados se muestran en la figura 39, de proporciones de datos de entrenamiento utilizando el descriptor GIST, se observa que las proporciones de datos de entrenamiento que poseen un menor porcentaje de datos para test tienen un coste computacional ligeramente más alto. En cuanto a la elección de la mejor proporción de datos, tenemos dos candidatos cuyos errores de localización son muy similares. Finalmente se ha escogido la proporción de datos de 85% para entrenamiento, 10% para validación y 5% para test debido a su menor coste computacional.

6.4.4. Parámetro *epochs*

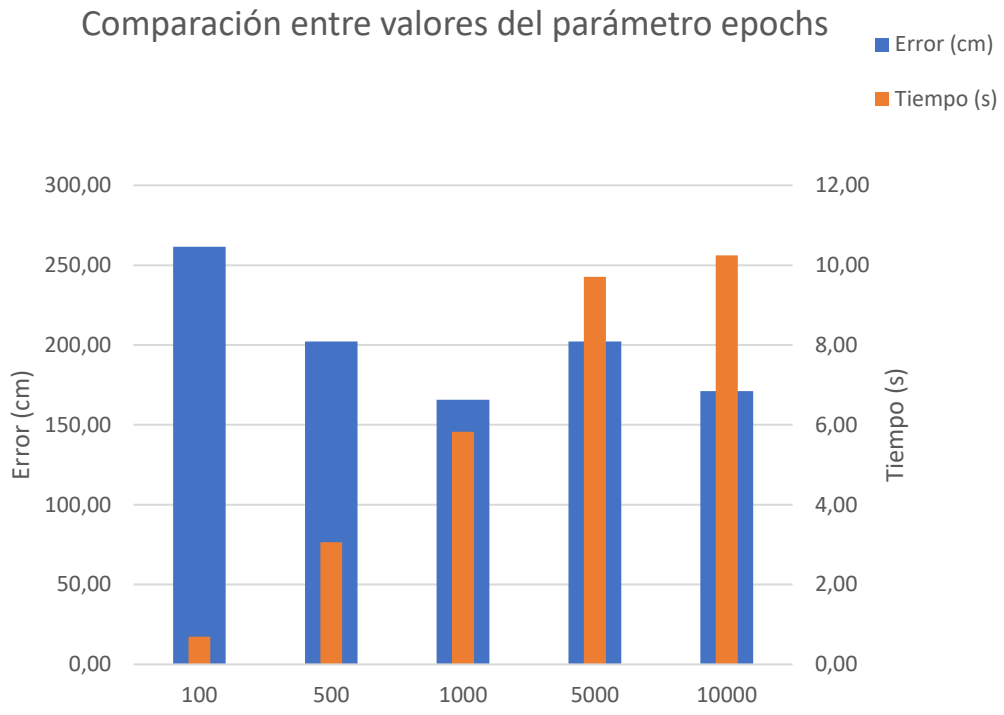


Figura 40. Comparativa entre diferentes valores del parámetro *epochs* usando el descriptor GIST y con *doorlit* como imágenes de prueba.

En esta gráfica podemos observar que el error disminuye cuanto más aumenta el número de epochs hasta llegar a 1000, después de esto se vuelve a incrementar hasta llegar a los 5000 y disminuye de nuevo. El valor mínimo de error se alcanza cuando los epochs son igual a 1000. En cuanto al coste computacional, podemos decir que este aumenta cuanto mayor es el número de epochs o iteraciones que se deben realizar.

6.4.5. Parámetro *validation*

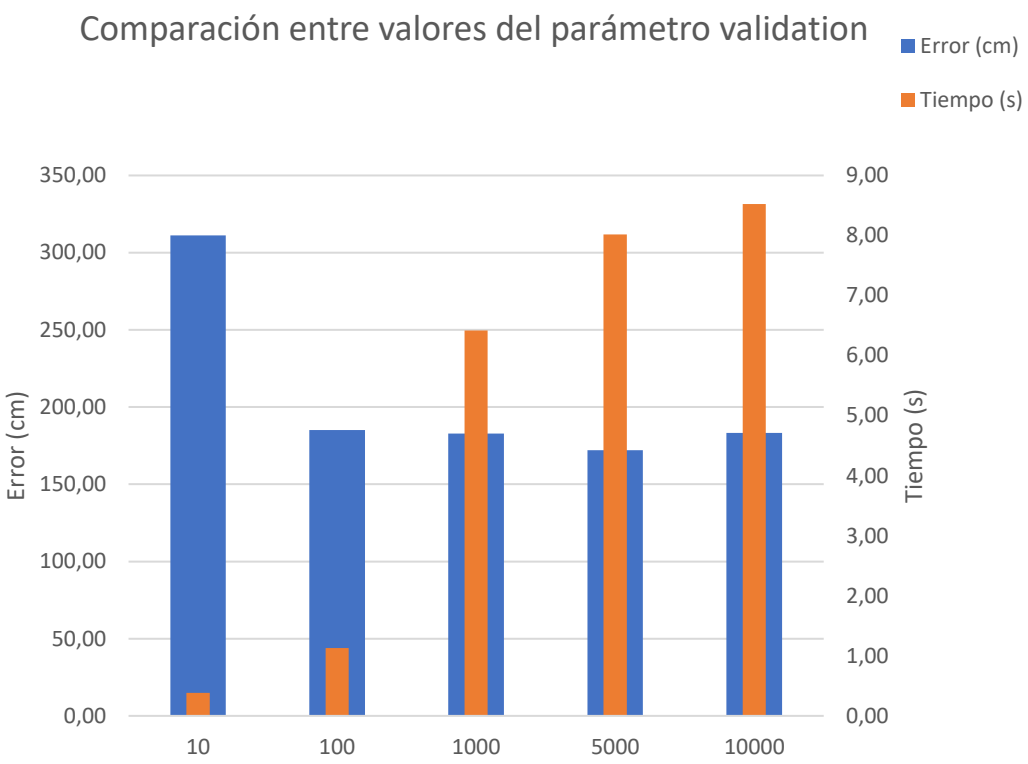


Figura 41. Comparativa entre distintos valores del parámetro *validation* usando el descriptor GIST y con *doorlit* como imágenes de prueba.

En este experimento se ha variado el parámetro *validation* utilizando el descriptor GIST y se han obtenido los resultados que se pueden ver en la figura 41. En esta figura se puede apreciar que, al aumentar el número de validaciones el tiempo de entrenamiento de la red neuronal y el tiempo de cálculo del error de localización se incrementa, llegando a su valor máximo cuando las validaciones son igual a 10000. En cuanto al error de localización podemos ver un decrecimiento de este conforme aumentan las validaciones hasta llegar al valor de 5000, a partir de este valor el error aumenta de nuevo.

6.5. Comparación del descriptor basado en ALEXNET con el descriptor GIST usando *doorlit* como imágenes de prueba y sin el uso de la técnica de aumento de datos

En este experimento se tomarán las mejores configuraciones tanto del descriptor basado en la CNN ALEXNET como del descriptor GIST sin *data augmentation* y se comparan entre sí.

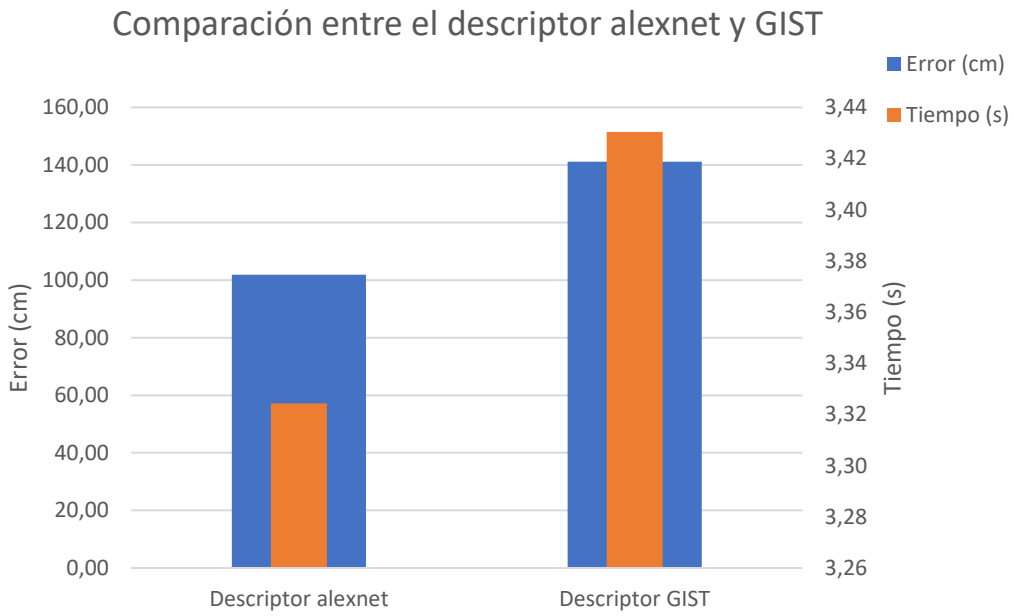


Figura 42. Comparativa entre el mejor caso del descriptor ALEXNET y del descriptor GIST.

La mejor configuración que se obtiene utilizando el descriptor ALEXNET, con *doorlit* como imágenes de test y sin *data augmentation* se describe en la subsección 6.2.3.

Mientras que, la configuración óptima que se obtiene utilizando el descriptor GIST, con *doorlit* como imágenes de test es la siguiente: se ha elegido el criterio de parada por defecto, es decir, el parámetro performance, la función de entrenamiento scg, una disposición de 3 capas ocultas de 20, 15

y 10 respectivamente y una proporción de datos de entrenamiento de 85% para entrenamiento, 10% para validación y 5% para test.

En la figura 42 podemos apreciar que comparando los resultados obtenidos entre la mejor configuración del descriptor GIST y del descriptor ALEXNET en las mismas condiciones, el descriptor ALEXNET consigue un tiempo de cómputo ligeramente menor y un error de localización considerablemente más reducido, habiendo una diferencia entre los dos de 40 cm. Por tanto, se puede decir que el descriptor ALEXNET mejora en todos los sentidos al descriptor GIST y, además es posible aplicarle la técnica de aumento de datos con lo que se reduce el error de localización prácticamente a la mitad, a costa de un gran empeoramiento en tiempo de cómputo.

6.6. Experimentos con el descriptor basado en la CNN ALEXNET usando *doorlit* como imágenes de prueba e introduciendo las coordenadas del *ground truth* por separado

Debido a que, en todos los casos, con el descriptor ALEXNET, en los que se ha aplicado el *data augmentation* ha mejorado considerablemente el error de localización a pesar de aumentar en consecuencia el coste computacional, se ha decidido utilizar solo el *data augmentation* en la realización de este experimento.

Además, se debe añadir que en esta prueba se realiza un entreno de dos redes neuronales independientes, en las que se introducen, a cada una de ellas, una coordenada de la posición real del robot. Por tanto, al testear las dos redes por separado, en cada una de ellas conseguiremos la estimación de una coordenada de la posición del robot. Tras esto, se juntan las dos coordenadas (x,y) y se realiza el cálculo del error de posición mediante el método de distancia euclídea.

6.6.1. Algoritmos de entrenamiento de la red neuronal

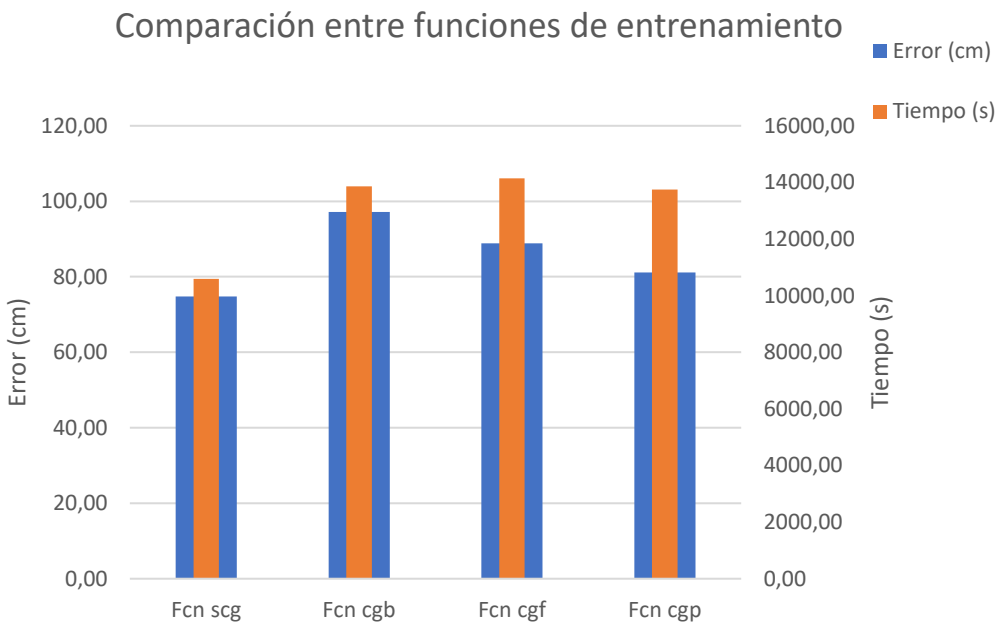


Figura 43. Comparativa entre funciones de entrenamiento utilizando el descriptor ALEXNET, usando *doorlit* como imágenes de prueba e introduciendo las coordenadas de la posición real del robot por separado.

En este experimento se prueban diferentes funciones de entrenamiento para la red neuronal, teniendo ALEXNET como descriptor e introduciendo las coordenadas de posición reales del robot por separado. En la figura 43, podemos observar que la mejor función de entrenamiento de nuestra red neuronal es la scg, función que mejora al resto tanto en error de localización como en tiempo de cómputo.

6.6.2. Número de capas ocultas y neuronas

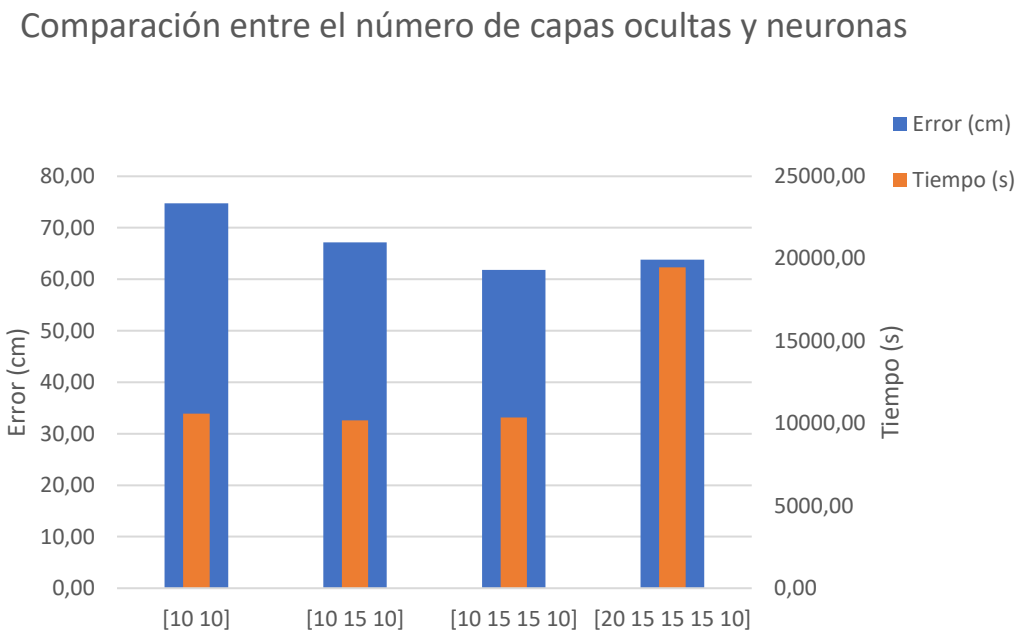


Figura 44. Comparativa entre número de capas ocultas y neuronas utilizando el descriptor ALEXNET, usando *doorlit* como imágenes de prueba e introduciendo las coordenadas de la posición real del robot por separado.

En el caso de la elección de número de capas ocultas y neuronas con el descriptor ALEXNET e introduciendo las coordenadas de posición reales del robot por separado, se puede apreciar que, el coste computacional aumenta casi el doble en redes neuronales que contienen 5 capas ocultas, en el resto todos los tiempos son similares. En la figura 44 podemos ver que la disposición de 4 capas ocultas y 10, 15, 15 y 10 neuronas respectivamente consigue un error de localización menor al resto, y, por tanto será la disposición elegida para los experimentos correspondientes.

6.6.3. Proporción de datos de entrenamiento

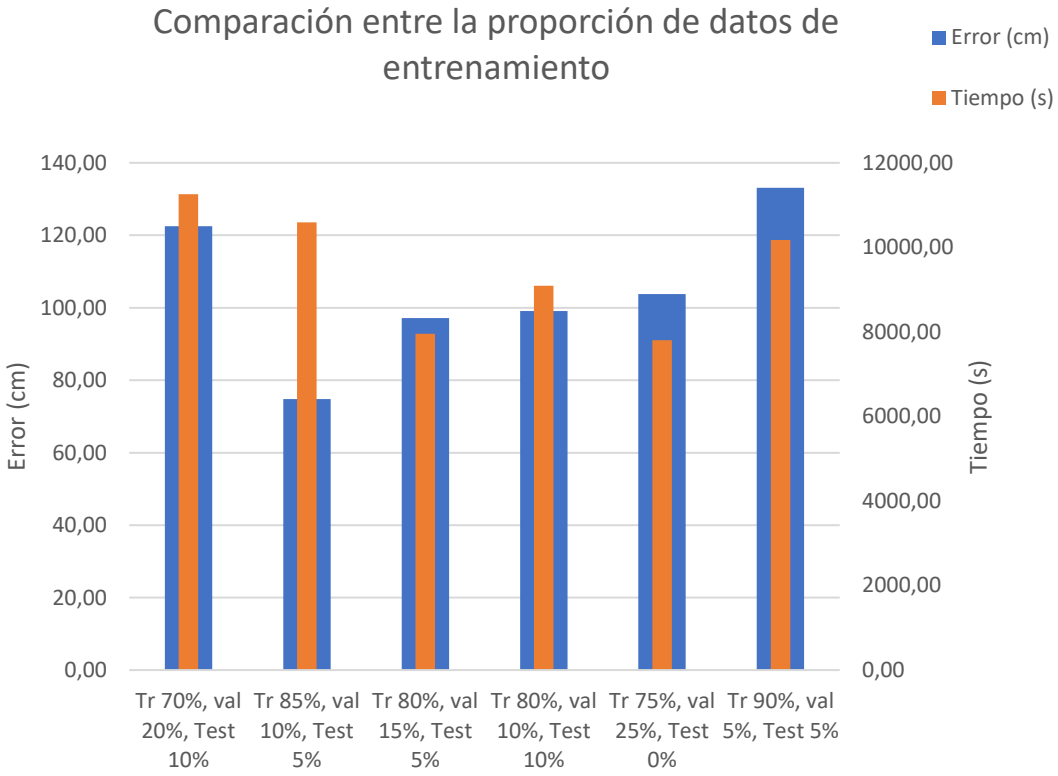


Figura 45. Comparativa entre proporciones de datos de entrenamiento utilizando el descriptor ALEXNET, usando *doorlit* como imágenes de prueba e introduciendo las coordenadas de la posición real del robot por separado.

En este experimento de proporciones de datos de entrenamiento utilizando el descriptor ALEXNET e introduciendo las coordenadas de posición reales del robot por separado, se observa que, el error mínimo de localización se consigue con la proporción de datos de 85% para entrenamiento, 10% para validación y 5%, aunque su coste computacional es el segundo más grande de todas las proporciones. Sin embargo, se ha elegido esta disposición aunque en general tenga un mayor coste computacional, ya que, consigue el error mínimo con una diferencia de 25 cm aproximadamente con la segunda mejor proporción.

6.6.4. Parámetro *epochs*

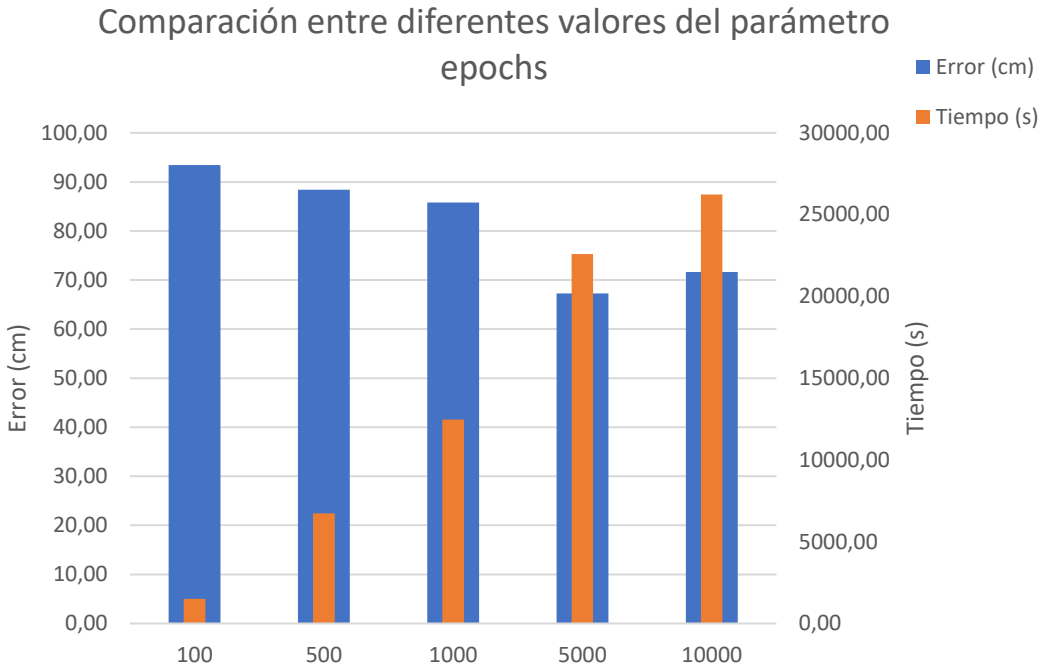


Figura 46. Comparativa entre diferentes valores del parámetro *epochs* utilizando el descriptor ALEXNET, usando *doorlit* como imágenes de prueba e introduciendo las coordenadas de la posición real del robot por separado.

En el caso de la variación del parámetro *epochs* cuando se utiliza el descriptor ALEXNET y se introducen las coordenadas *ground truth* del robot por separado, se puede apreciar que, al incrementar las *epochs* o iteraciones el tiempo de cómputo aumenta. El error de localización decrece hasta llegar a los 5000 *epochs,* y por encima de este valor el error vuelve a incrementarse hasta llegar a los 10000 epochs.

6.6.5. Parámetro *validation*

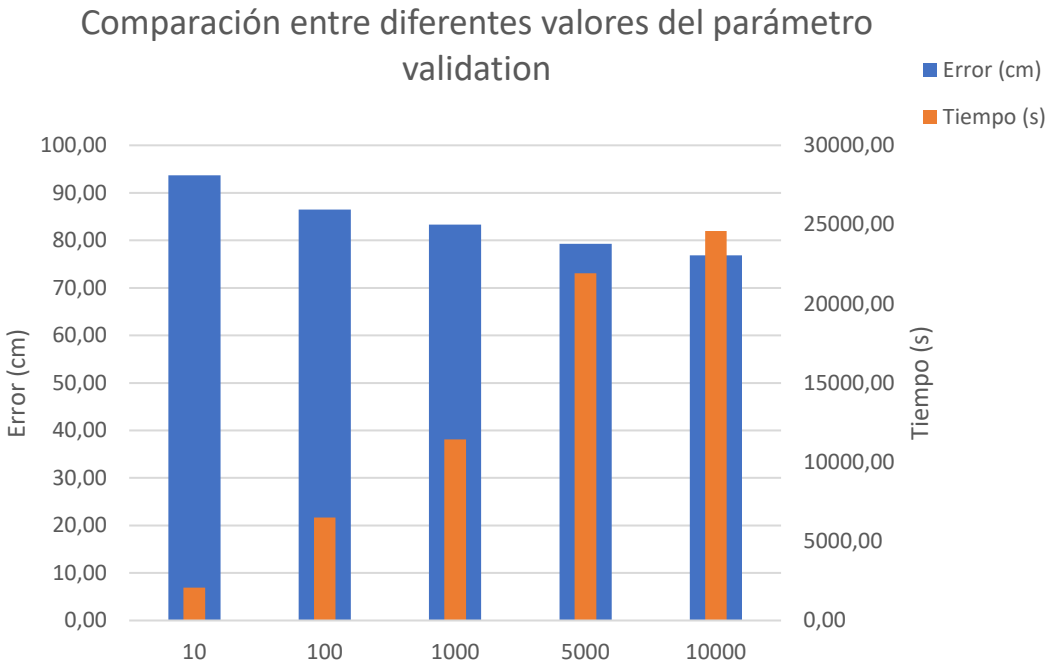


Figura 47. Comparativa entre diferentes valores del parámetro *validation* utilizando el descriptor *ALEXNET*, usando *doorlit* como imágenes de prueba e introduciendo las coordenadas de la posición real del robot por separado.

En el experimento del parámetro *validation* con el descriptor ALEXNET e introduciendo las coordenadas *ground truth* del robot por separado, podemos observar que el error de localización a medida que aumentan las validaciones disminuye de forma progresiva hasta alcanzar su mínimo cuando las validaciones son igual a 10000. Por otro lado, el tiempo de cómputo aumenta cuanto más se incrementa el valor del parámetro validation.

6.7. Comparación del descriptor basado en ALEXNET usando *doorlit* como imágenes de prueba con coordenadas de *ground truth* introducidas juntas y por separado y utilizando la técnica de data *augmentation*

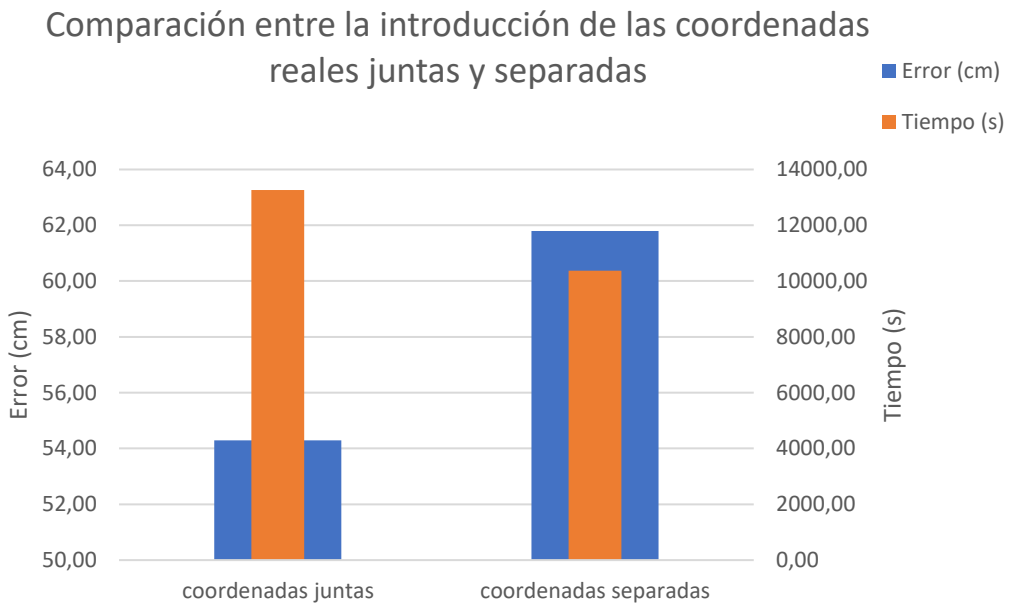


Figura 48. Comparativa entre la introducción a la red neuronal de las coordenadas reales juntas o por separado utilizando el descriptor ALEXNET y usando *doorlit* como imágenes de prueba.

La mejor configuración que se obtiene utilizando el descriptor ALEXNET, con *doorlit* como imágenes de test y con *data augmentation* se describe en la subsección 6.2.3.

Mientras que, la configuración óptima que se consigue utilizando el descriptor GIST y con *doorlit* como imágenes de test es la siguiente: se ha elegido el criterio de parada por defecto, es decir, el parámetro performance, la función de entrenamiento scg, una disposición de 4 capas ocultas de 10, 15, 15 y 10 respectivamente y una proporción de datos de entrenamiento de 85% para entrenamiento, 10% para validación y 5% para test.

Analizando los resultados de este experimento descritos en la figura 48, podemos apreciar que cuando realizamos el experimento introduciendo las coordenadas de la posición real del robot en la red neuronal por separado, se consiguen disminuir el coste computacional aproximadamente 3000 segundos respecto a la introducción de las coordenadas juntas. Sin embargo, introduciendo las coordenadas juntas y realizando solo el entrenamiento de una red neuronal nos aporta un error de localización alrededor de 7 cm menor. Por tanto, siendo los errores de los dos métodos razonables, la elección de una u otra dependerá de si el coste computacional añadido que tiene el método de introducción conjunta de las coordenadas es asumible o no.

7. Conclusiones y líneas futuras

Este proyecto tiene como objetivo el estudio de la aplicación de redes neuronales mediante descriptores de apariencia global para resolver tareas de localización en robots móviles en interiores con condiciones de trabajo reales, como cambios en la iluminación de la sala u oclusiones provocadas por objetos tales como sillas o plantas. Para este trabajo el robot dispone de una cámara como sensor de visión para la captación de información del entorno, la cual esta montada sobre el robot junto a un espejo convexo hiperbólico hacia el que apunta. El montaje del espejo y la cámara se denomina sistema catadióptrico y consigue un ángulo de visión de 360º alrededor del robot obteniendo imágenes omnidireccionales.

Una vez se tienen las imágenes omnidireccionales, se eligen los métodos de descripción GIST y el método de descripción basado en la CNN ALEXNET para la extracción de la información de dichas imágenes. Una vez terminada la tarea de *mapping*, es decir, se ha conseguido la descripción de las imágenes omnidireccionales, empezamos la resolución del problema de localización mediante el entrenamiento de redes neuronales de regresión (o de ajuste de datos) que estimen las coordenadas (x,y) del robot con dichos descriptores. También se ha estudiado la resolución de dicho problema introduciendo a la red neuronal las coordenadas *ground truth* del robot por separado.

En este trabajo se ha realizado un estudio de los diferentes parámetros de entrenamiento y criterios de parada de las redes neuronales con ambos descriptores utilizando el programa Matlab. También se ha estudiado el uso de diferentes grupos de imágenes para probar las redes neuronales. Además, se ha comparado la resolución de la tarea de localización mediante el descriptor GIST y el descriptor basado en la CNN ALEXNET. Posteriormente se cotejan el método por defecto que se utiliza para introducir las coordenadas *ground truth,* es decir, introducirlas juntas, con el método de introducirlas por

separado, y todo esto realizado con el descriptor ALEXNET y aplicando *data augmentation* a la base de datos.

Las conclusiones extraídas de estos experimentos mencionados se describen a continuación:

- Uno de los parámetros de entrenamiento estudiado es la función de entrenamiento de la red neuronal. Las mejores funciones de entrenamiento han sido la cgp y scg, siendo la scg en casi todos los experimentos la que conseguía un coste computacional menor y muchas veces también el error de localización más pequeño.

- Otro parámetro de entrenamiento investigado ha sido el número de capas ocultas y de neuronas del que debe constar nuestra red neuronal. El resultado más interesante de este experimento se basa en el número de capas ocultas de la red neuronal siendo las mejores redes, aquellas que están formadas por 3 o 4 capas ocultas.

- El último parámetro de entrenamiento es la proporción de datos de entrenamiento de la red neuronal. En los experimentos de este parámetro el mejor resultado siempre ha sido la proporción de 85% de imágenes para entrenamiento, 10% para validación y 5% para test, siendo en la mayoría de los casos el óptimo en cuanto a error de localización y también en tiempo de cómputo.

- En cuanto a criterios de parada del entrenamiento de la red neuronal tenemos que, el criterio utilizado ha sido en todos los casos el parámetro *performance*. Este criterio de parada fue el elegido por defecto para los experimentos en los que se necesitaba un criterio de parada con el fin de estudiar los parámetros de entrenamiento. Dicho criterio dio mejores resultados que los demás con las mejores configuraciones de los parámetros de entrenamiento. Los otros criterios estudiados fueron los

parámetros *epochs* y *validation*. Aunque estos criterios mejoraban el error de localización progresivamente cuanto más aumentaban sus valores, no se llego a un error menor que el que se conseguía con el parámetro *performance*.

- Una vez se tienen las mejores configuraciones de las redes neuronales se realiza la comparación entre los experimentos realizados utilizando *doorlit* y *day* como imágenes de prueba, de la cual se concluye que la utilización de *doorlit* para test y los demás grupos de imágenes para entrenamiento conseguía el menor error de localización además de un coste computacional más pequeño. Estos resultados son iguales tanto con la utilización de *data augmentation* en la base de datos como sin el uso de esta técnica.

- También se realiza una comparación entre el descriptor GIST y ALEXNET, en la cual se demuestra que el descriptor basado en la CNN ALEXNET es superior en todos los aspectos consiguiendo mejores resultados tanto en coste computacional como en error de localización.

- En cuanto a la técnica de *data augmentation*, cabe destacar que los resultados conseguidos por esta técnica son mejores en todos los experimentos en cuanto a error de localización, siendo este aproximadamente la mitad que sin el uso del aumento de datos. Sin embargo, el coste computacional es muy alto en todos los experimentos. Por tanto, el uso de esta técnica dependerá de la aplicación que se requiera, teniendo en cuenta si es asumible o no ese coste computacional.

- Por último, se realiza otra comparación con el descriptor ALEXNET entre el entrenamiento de la red neuronal introduciéndole las coordenadas *ground truth* juntas o separadas. De esta comparación se obtiene la

conclusión de que al introducir las coordenadas por separado se consigue un coste computacional notablemente más bajo, pero se obtiene un error de localización mayor.

Los resultados obtenidos muestran que la resolución de la tarea de localización mediante redes neuronales con descriptores holísticos es un método muy viable, sobre todo si se combinan otras técnicas como el *data augmentation*. Mediante estos métodos un robot puede construir un modelo del entorno y utilizarlo para localizarse en el espacio con cierta precisión. Esto podría suponer aplicaciones muy interesantes dentro de la robótica móvil, por tanto, se debe profundizar más en este campo todavía muy inexplorado.

También se debe comentar que este trabajo se podría extender en líneas futuras de investigación entre las que se pueden destacar:

- Realizar un estudio de posibles métodos para disminuir el coste computacional del entrenamiento de redes neuronales usando el descriptor basado en la CNN ALEXNET cuando se utiliza la técnica de *data augmentation*.

- Intentar superar la barrera del coste computacional del algoritmo GIST cuando se le aplica a la base de datos la técnica de *data augmentation*.

- Realizar el mismo estudio de este proyecto con una base de datos de imágenes panorámicas y comparar resultados.

- Realizar el estudio de estos métodos con una base de datos captada en un entorno de exterior con el objetivo de conocer la viabilidad de estas técnicas en esos entornos.

Referencias

[1]. L. Payá, A. Gil and O. Reinoso, “A State-of-the-Art Review on Mapping and Localization of Mobile Robots Using Omnidirectional Vision Sensors”, *Journal of Sensors*, vol.2017, pp.1-20, April 2017.

[2]. A. Oliva and A. Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope”, International journal of computer vision, vol.42, no.3, pp.145-175, 2001.

[3]. E. Menegatti, T. Maeda, and H. Ishiguro, “Image-based memory for robot navigation using properties of omnidirectional images”, *Robot. Auto. Syst.*, vol.47, no.4, pp.251-267, Jul 2004.

[4]. S. Cebollada, L. Payá, V. Roman, and O. Reinoso, “Hierarchical localizationin topological models under varying illumination using holistic visual descriptors”, *IEEE Access*, vol.7, pp. 49580-49595, 2019.

[5]. F. Amorós, L. Payá, J. M. Marín, and O. Reinoso, “Trajectory estimation and optimization through loop closure detection, using omnidirectional imaging and global appearance descriptors”, Expert Syst. Appl., vol.102, pp.273-290, Jul 2018.

[6]. H. Bay, T. Tuytelaars, and L. VanGool, “Surf: Speeded up robust features”, *Computer visión-ECCV 2006*, pp.404-417, 2006.

[7]. H. Bay, A. Ess, T. Tuytelaars, and L. VanGool, “Speeded-up robust features (surf)”, *Computer vision and image understanding*, vol.110, no.3, pp.346-359, 2008.

[8]. D. G. Lowe, “Distinctive image features from scale-invariant keypoints”, *International journal of computer vision*, vol.60, no.2, pp.91-110, 2004.

[9]. M. Flores, “Desarrollo de algoritmos de odometría visual mediante una cámara de 360 grados”, Trabajo de Fin de Máster, Ingeniería de Sistemas y Automática, Universidad Miguel Hernández de Elche, Elche, Alicante, España, 2018.

[10]. A. M. Elmoogy, X. Dong, T. Lu, R. Westendorp and K. R. Tarimala, “SurfCNN: A Descriptor Accelerated Convolutional Neural Network for Image-Based Indoor Localization”, *IEEE Access*, vol.8, pp.59750-59752, 2020.

[11]. V. Román, “Localización de un robot móvil mediante visión por computador ante grandes cambios de iluminación”, Trabajo de Fin de Máster, Ingeniería de Sistemas y Automática, Universidad Miguel Hernández de Elche, Elche, Alicante, España, 2017.

[12]. S. Cebollada, V. Román, L. Payá, M. Flores, L. M. Jiménez and O. Reinoso, “Uso de técnicas de machine learning para realizar mapping en robótica móvil”, in *Proceedings of the XL Jornadas de Automática*, Ferrol, La Coruña, España, pp. 686-693, 2019.

[13]. C. Cortes and V. Vapnik, “Support-vector networks”, *Machine Learning*, pp. 273-297, Sep 1995.

[14]. A. Carrio, C. Sampedro, A. Rodríguez and P. Campoy, “A Review of Deep Learning Methods and Applications for Unmanned Aerial Vehicles”, *Journal of Sensors*, vol. 2017, no. 2, pp. 1-13, August 2017.

[15]. L. Payá, “Técnicas de descripción de la apariencia global de escenas: Aplicación a la creación de mapas y localización de robots móviles”, Tesis Doctoral, Ingeniería de Sistemas y Automática, Universidad Miguel Hernández de Elche, Elche, Alicante, España, 2014.

[16]. N. Winters, J. Gaspar, G. Lacey and J. Santos-Victor, “Omni-directional vision for robot navigation”, in *Proceedings of the IEEE Workshop on Omnidirectional Vision*, Hilton Head Island, SC, USA, pp. 21-28, 2000.

[17]. L. Payá, O. Reinoso, Y. Berenguer and D. Úbeda, “Using omnidirectional vision to create a model of the environment: A comparative evaluation of global-appearance descriptors”, *Journal of Sensors*, 2016.

[18]. N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection”, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, June 2005.

[19]. V.Grassi Junior and J. Okamoto Junior, “Development of an omnidirectional vision system," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol.28, pp. 58-68, March 2006.

[20]. L. Payá, A. Peidró, F. Amorós, D. Valiente and O. Reinoso, “Modeling Environments Hierarchically with Omnidirectional Imaging and Global Appearance Descriptors”, *Remote Sensing*, vol. 10, no. 4, pp. 522, March 2018.