Universidad Zaragoza
1542

Trabajo Fin de Grado

Reconocimiento de lugares con invarianza a cambios del entorno mediante redes neuronales.

Condition invariant place recognition using neural networks.

Autor/es

Daniel Olid Hernández

Director/es

Javier Civera Sancho (dir.)
José María Fácil Ledesma (codir.)

Escuela de Ingeniería y Arquitectura, Universidad de Zaragoza.

2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Daniel Olid Hernández,

con nº de DNI 73020655-w en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Reconocimiento de lugares con invarianza a cambios del entorno mediante redes neuronales,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 31 de Agosto de 2017

Fdo: Daniel Olid Hernández

Agradecimientos

Me gustaría utilizar este espacio para darles las gracias a mis directores de proyecto, el Dr. Javier Civera y D. José M. Fácil. Gracias a los dos por invertir parte de vuestro tiempo y vuestro esfuerzo en guiarme. Gracias Javier por confiar en mí para llevar a cabo el trabajo y ayudarme a descubrir un área tan interesante. Gracias Chema por no perder la paciencia con mis largos correos informativos y haber tenido siempre un momento para resolver mis dudas en el laboratorio.

También me gustaría darle las gracias a mi madre, a mi padre, a mi tío y a mi abuelo por hacer de mecenas culturales y haberme animado constantemente durante la realización del trabajo. Por último, a mis amigos, por seguir insistiendo en sacarme al mundo exterior a pesar de mis "no puedo, estoy con el tfg".

Resumen

El reconocimiento de lugares es un problema que consiste en, dada una imagen de entrada, encontrar en una base de datos con imágenes aquellas que se corresponden al mismo lugar. Es un área que se encuentra en investigación ya que forma parte de tecnologías en auge como la navegación autónoma y la realidad aumentada. Uno de los retos a resolver es que la apariencia de los lugares cambia por la dinámica de los entornos. Condiciones climatológicas como la nieve o la lluvia pueden hacer que un lugar parezca otro distinto. Algunas aproximaciones que utilizan redes neuronales presentan una mayor invarianza a los cambios de apariencia. Por ello, este trabajo emplea imágenes captadas con una cámara monocular y redes neuronales con estructuras siamesas y triplets para desarrollar el reconocedor.

La implementación del reconocedor conlleva varios desafíos. Por un lado, disponer de datos adecuados para el entrenamiento de las redes y para la comprobación del reconocedor. Para ello se propone un conjunto de datos diseñado a partir de imágenes de los vídeos del trayecto Nordland. Por otro lado, la elección de la estructura y el proceso de entrenamiento de las propias redes neuronales. En este sentido, se ha comprobado que las redes neuronales, especialmente las triplets, son capaces de extraer características de los lugares robustas a los cambios de apariencia vistos en su entrenamiento. El reconocedor desarrollado alcanza resultados del estado del arte con imágenes del conjunto Nordland, rivalizando con aproximaciones tanto similares, como diferentes.

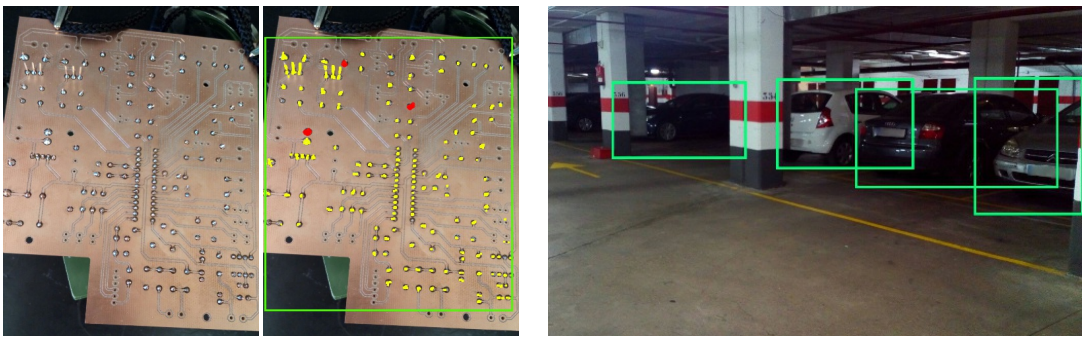
Índice general

Índice de figuras

Índice de tablas

1. Introducción

Este trabajo se enmarca dentro de la visión por computador. Una disciplina que estudia el análisis, procesado e interpretación de las imágenes mediante sistemas computacionales. La visión abarca, entre otros, problemas como el reconocimiento y seguimiento de objetos, la localización y el mapeado simultáneos (SLAM[1]) o la detección de expresiones faciales. Se aplica en campos muy diversos, como en la inspección de la calidad de las soldaduras en una cadena de montaje (Figura 1.1a). También en vigilancia, como el detector de coches en el garaje de la Figura 1.1b.



(a) (b)

Figura 1.1: Ejemplos de aplicación de la visión por computador. **a:** Detección de soldaduras defectuosas en una placa. **b:** Detección y segmentación de coches aparcados en un garaje.

El ámbito en el que se va a aplicar la visión en este trabajo es en el problema del reconocimiento de lugares. Este problema consiste en, dada una imagen, encontrar en una base de datos con un gran número de imágenes la o las que corresponden al mismo lugar. La dificultad del problema radica fundamentalmente en que los lugares son dinámicos y cambian de aspecto. Algunos cambios son climatológicos como la lluvia, la nieve o el color de las hojas entre otoño y verano. También influyen la iluminación, como el cambio de luz del día a la noche, los cambios del punto de vista, las oclusiones y los

[1]SLAM es el acrónimo en inglés de la localización y el mapeado simultáneos (*Simultaneous Localization And Mapping*)

elementos como objetos o personas que no están siempre en el mismo lugar. Estos son solo algunos ejemplos de los fenómenos que pueden hacer que dos imágenes tomadas en el mismo lugar, parezcan lugares distintos. En la Figura 1.2 se puede ver el funcionamiento de un reconocedor de lugares genérico en un garaje. En este entorno, por ejemplo, el reconocedor debe tener en cuenta que los coches pueden no estar siempre aparcados en la plaza.

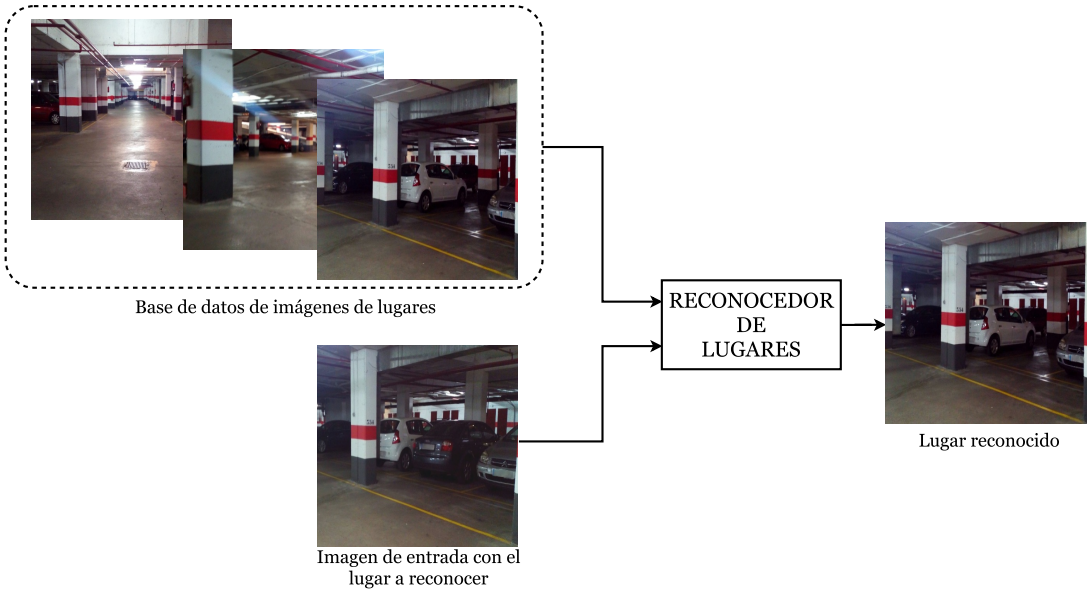


Figura 1.2: Funcionamiento como caja negra de un reconocedor de lugares. La entrada del reconocedor es la información de n lugares por un lado y la del lugar a reconocer por otro. La salida es el lugar reconocido.

La mayoría de aproximaciones al problema utilizan algoritmos de visión por computador que detectan puntos salientes o características relevantes de las imágenes para describir los lugares. Posteriormente comparan las características de unos lugares con otros. Dos técnicas relevantes son FAB-MAP [1] y DBoW [2]. El problema de estos métodos es que los algoritmos de visión utilizados, a pesar de ser robustos a pequeños cambios del punto de vista o de iluminación, empeoran mucho sus resultados ante cambios grandes en la apariencia. Un ejemplo se puede ver en la Figura 1.3, donde se intenta emparejar dos imágenes del mismo lugar de día y de noche. El algoritmo utilizado falla en la mayor parte de los elementos que considera coincidentes por lo que sería difícil reconocer el lugar.

Para solucionar el problema de la apariencia, algunas alternativas se centran en cambiar el sensor utilizado para captar la información. Desde una cámara infrarroja [3] o un sensor LIDAR [4], hasta un sistema de cámaras estéreo [5]. Los sensores más complejos, como los infrarrojos o el LIDAR, son más robustos ante condiciones que dificultan

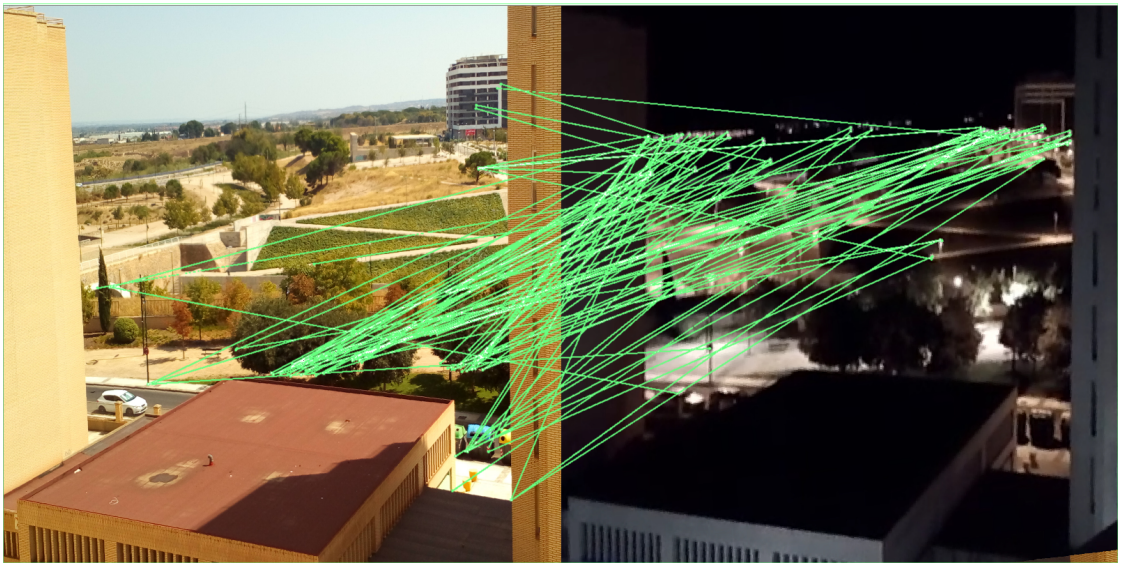


Figura 1.3: Reconocimiento de una escena de día y de noche. Se utiliza el algoritmo ORB para la extracción y detección de características comunes entre las dos imágenes. Las líneas verdes indican los emparejamientos.

la visibilidad como la niebla o la oscuridad. El problema de estos sensores es que son más voluminosos y requieren más potencia de procesamiento. Las imágenes tomadas con cámaras monoculares, se ven más afectadas factores ambientales pero son más baratas y accesibles. Por ello, en este trabajo se decide utilizar una cámara monocular para obtener imágenes en color del entorno a reconocer.

Una alternativa es utilizar redes neuronales para procesar y describir las imágenes. Niko Sünderhauf *et al.* analizaron en [6] el uso de redes con este propósito en el reconocimiento de lugares. Su trabajo demuestra que las redes neuronales pueden mejorar los resultados obtenidos por otras técnicas cuando cambia la apariencia de los lugares. Entre las aproximaciones que han utilizado redes destacan Gómez-Ojeda *et al.* [7] y Chen *et al.* [8], que han obtenido resultados del estado del arte en el reconocimiento de lugares que presentan cambios de apariencia. Por ello se decide utilizarlas en este trabajo.

El reconocimiento de lugares juega un papel importante en áreas como robótica, especialmente en la navegación autónoma. En un proceso de SLAM, por ejemplo, se puede utilizar el reconocimiento de lugares en el cierre de bucle ("*loop closure*") y en la reutilización de mapas. En el cierre de bucle, detectar si el sistema ya ha visitado previamente el lugar en el que se encuentra ayuda a corregir el mapa creado. En la reutilización de mapas, reconocer una zona que ya ha sido mapeada permite reutilizar los datos ya disponibles en lugar de mapear de nuevo.

Una vez introducido el problema y argumentado su complejidad, los principales objetivos de este trabajo son los siguientes:

- Implementar un reconocedor de lugares invariante a cambios en las condiciones del entorno. Se desea que sea robusto a factores climatológicos y estacionales. El sistema se basa en imágenes y utiliza redes neuronales para procesarlas y extraer descriptores. Estos descriptores se pueden comparar para determinar el parecido entre lugares. Se puede ver un ejemplo del tipo de cambios de apariencia que se desea considerar en la Figura 1.4.

- Elaborar un conjunto de datos ("*dataset*") para desarrollar y comprobar el funcionamiento del reconocedor. Para ello se partirá de las imágenes de un conjunto de vídeos ya existente. El conjunto final se propondrá como entorno de referencia en igualdad de condiciones para cualquier investigación relacionada con el reconocimiento de lugares.

- Evaluar el sistema en el conjunto de datos desarrollado y comparar el resultado con otras técnicas del estado del arte en el reconocimiento de lugares.

Figura 1.4: Imagen de un mismo lugar en invierno, verano, primavera y otoño. Se pueden apreciar grandes cambios de apariencia entre las imágenes a pesar de haber sido tomadas en la misma ubicación. Las imágenes han sido extraídas de los vídeos del trayecto Nordland.

1.1. Estructura del trabajo

En este capítulo se han introducido el problema y los objetivos. En el capítulo 2 se introducen los conceptos de redes neuronales necesarios para comprender el resto del trabajo. En el capítulo 3 se explica la estructura y construcción del conjunto de datos utilizado. En el capítulo 4 se plantean y explican distintas alternativas de red neuronal. En el capítulo 5 se presentan los resultados experimentales obtenidos. En el capítulo 6 se extraen las conclusiones del trabajo. Finalmente, en el capítulo 7 se aporta información acerca de las herramientas utilizadas y la organización temporal de la realización del trabajo.

2. Introducción a las redes neuronales

En este capítulo se desarrollan conceptos de las redes neuronales que son necesarios para entender el resto del trabajo.

Las redes neuronales forman parte del aprendizaje automático ("*machine learning*"); la rama de las ciencias de la computación que trabaja con algoritmos capaces de solucionar problemas a partir del análisis de datos. Las redes neuronales aparecieron hace más de 70 años [9] pero han resurgido gracias al desarrollo tecnológico. Especialmente al incremento de la potencia y memoria de los ordenadores, desarrollo de las tarjetas gráficas (GPUs) y a la creciente disponibilidad de grandes conjuntos de datos.

Las redes neuronales son algoritmos capaces de aprender funciones complejas sobre los datos de entrada mediante el entrenamiento con conjuntos de datos. Tienen aplicaciones muy diversas desde el reconocimiento de letras y dígitos escritos a mano, hasta la predicción de movimientos de los mercados bursátiles. En muchas de sus áreas de aplicación han mejorado los resultados que se obtenían con otros métodos como en el reconocimiento de objetos [10].

2.1. Neurona artificial

La unidad básica de las redes neuronales son las neuronas artificiales. Cada neurona es un elemento capaz de procesar individualmente un vector n-dimensional de entrada, $\boldsymbol{x} = (x_1, x_2, ..., x_n)^\intercal$, para producir una salida determinada, y.

Cada neurona tiene una serie de pesos (tantos como entradas) o parámetros internos, $\boldsymbol{w} = (w_1, w_2, ..., w_n)^\intercal$. La neurona multiplica las entradas con sus pesos mediante el producto escalar y añade un sesgo ("*bias*"), b. Finalmente aplica una función de activación no lineal, $f : \mathbb{R} \to \mathbb{R}$, para obtener la salida. Tanto los pesos como el sesgo se establecen mediante el aprendizaje.

Se puede ver un ejemplo de la representación típica de una neurona artificial en la Figura 2.1. En la imagen se pueden apreciar las entradas, la salida y la función matemática

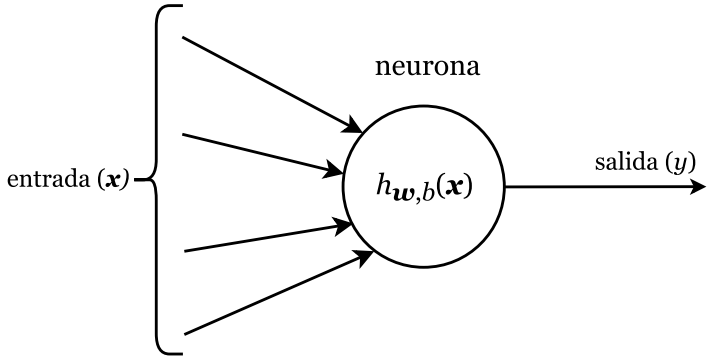


Figura 2.1: Representación de una neurona artificial. Donde $\mathbf{x}$ es la entrada, y la salida y $h_{\boldsymbol{w},b}(\boldsymbol{x})$ la función que implementa cada neurona. Esta representación está inspirada en la estructura de una neurona real.

completa que implementa cada neurona, la cual es la siguiente:

$$h_{\boldsymbol{w},b}(\boldsymbol{x}) = f(\boldsymbol{w}^\intercal \boldsymbol{x}) = f\left(\sum_{i=1}^{n} w_i x_i + b\right). \tag{2.1}$$

2.2. Estructuras

Para formar una red neuronal se agrupan varias neuronas artificiales formando capas. Entre las neuronas de la red se establecen relaciones no lineales que permiten implementar funciones más complejas que las que implementan las neuronas individualmente. La salida de cada capa sirve de entrada a la capa siguiente generalmente. Las neuronas de una misma capa comparten la misma entrada pero cada neurona tiene sus propios pesos, por lo que cada una puede tener una salida diferente. A los parámetros como el número de neuronas de cada capa, el número de capas o la combinación de diferentes capas se les denomina hiperparámetros. Estos hiperparámetros no se aprenden mediante el entrenamiento de la red sino que se fijan antes mediante un proceso que se conoce como validación (se explica el proceso en la sección 2.4 de este capítulo.

Un ejemplo de red neuronal es la de la Figura 2.2. Esta red dispone de dos capas ocultas (las capas que no son entrada ni salida se denominan capas ocultas), con dos y tres neuronas cada una respectivamente. La capa de salida está formada por una única neurona. En esta red, cada neurona está conectada a todas las salidas de la capa anterior, lo cual no siempre es así. Existen varios tipos de capas con las que formar una red neuronal en función de las conexiones que presentan. Un ejemplo de las capas utilizadas en este trabajo se puede ver en la Figura 2.3 y son las siguientes:

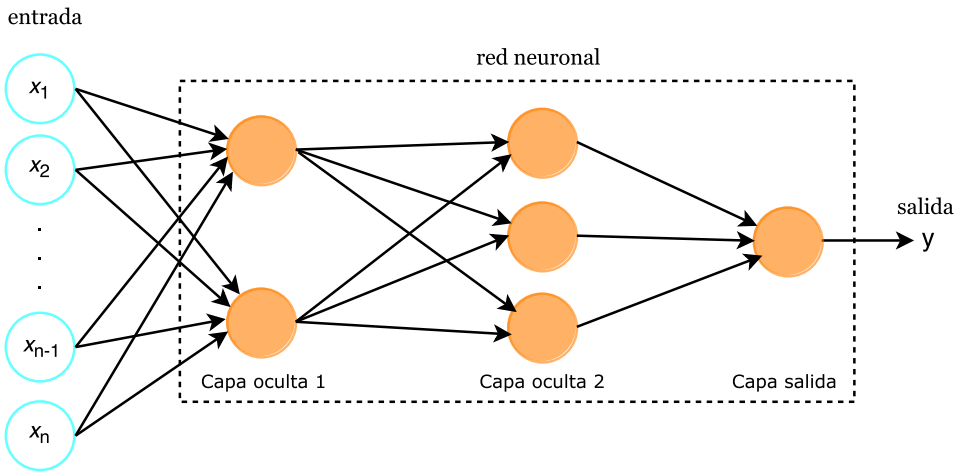


Figura 2.2: Ejemplo de red neuronal con varias capas. Las neuronas se representan con círculos naranjas. La red cuenta con dos capas ocultas. La primera con dos neuronas y la segunda con tres. La capa de salida cuenta con una única neurona.

Capa totalmente conectada ("*fully-connected*"): La Figura 2.3a muestra una capa totalmente conectada. En estas capas, cada neurona se conecta a todas las de la capa anterior. Esto permite que puedan tener en cuenta la entrada al completo en su procesamiento, con la desventaja de tener un elevado número de parámetros. Esto se debe a que cada neurona artificial tiene tantos pesos como entradas, junto a un sesgo que deben aprender y almacenar. Si el tamaño de la entrada es M y el número de neuronas es N, el número de parámetros de una capa totalmente conectada es $(M + 1) \cdot N$.

Si las imágenes de entrada del reconocedor de lugares, como se verá más adelante, son de $224 \times 224 = 50176$ y se quisiera utilizar una capa de este tipo con la mitad de neuronas que de entradas $(50176/2 = 25088)$. Se necesitarían $(M + 1) \cdot N = (50176 + 1) \cdot 25088 = 1258840576$ parámetros. Utilizando 32 bits para cada parámetro, la capa ocuparía 4,69 gibibytes (GiB) en memoria. Al aumentar el número de parámetros, se complica el proceso de aprendizaje ya que se necesitan más ejemplos para entrenar la red y evitar el sobreajuste[1]. También se incrementa el coste computacional y la memoria requerida, pues hay que almacenar y procesar más elementos.

Capa localmente conectada: Muchos tipos de datos, como las imágenes o los usados para representar el lenguaje, presentan relación local. En una fotografía de un coche, por ejemplo, una rueda está formada por píxeles que se encuentran próximos entre ellos. Las capas localmente conectadas pueden aprovechar esta relación. En estas capas, cada

[1]El sobreajuste es el fenómeno que se da cuando un método de aprendizaje presenta buenos resultados solo con los datos que ha visto durante el entrenamiento, siendo incapaz de generalizar su funcionamiento a otros datos nuevos aunque sean similares a los de entrenamiento.

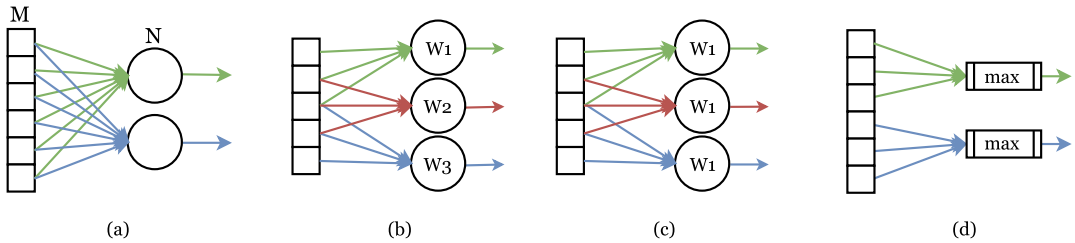


Figura 2.3: Varios tipos de capas de una red neuronal. Las conexiones de cada neurona se marcan en distinto color para facilitar la visualización. **a**: Totalmente conectada. Donde M es la capa anterior y N la capa actual. **b**: Localmente conectada. Cada neurona se conecta a 3 entradas de la capa anterior. Cada neurona tiene sus propios parámetros: W_1, W_2, W_3. **c**: Convolucional. Las neuronas comparten los mismos parámetros: W_1. **d**: Reductora máxima.

neurona se conecta solo a una parte de las de la capa anterior. De esta forma, si los datos de entrada presentan relación local, estas capas pueden alcanzar un rendimiento similar al de las capas totalmente conectadas. Además son más fáciles de entrenar, ya que utilizan menos parámetros. Esto se debe a que disminuir el número de conexiones es equivalente a reducir las entradas y por tanto disminuye el número de parámetros que debe aprender cada neurona. La Figura 2.3b muestra una capa localmente conectada formada por 3 neuronas. Cada neurona se conecta solo a 3 entradas de la capa anterior. Generalmente las neuronas se conectan al mismo número de entradas.

Para comparar el número de parámetros con las capas totalmente conectadas se va a utilizar un ejemplo similar. Se utiliza la misma imagen de entrada de $224 \times 224 = 50176$. Ahora se emplea una capa localmente conectada, con la mitad de neuronas que de entradas ($50176/2 = 25088$) y con cada neurona conectada a una décima parte de las entradas ($50176/10 \approx 5018$). Denominando N al número de neuronas de la capa actual y C al número de entradas de cada neurona, se necesitarían $(C + 1) \cdot N = (5018 + 1) \cdot 25088 = 125906636$ parámetros. Diez veces menos parámetros que con la capa totalmente conectada. Utilizando 32 bits para cada parámetro, la capa ocuparía 0,47 GiB en memoria, a diferencia de los 4,69 de la capa totalmente conectada.

Capa convolucional: Estas capas son un tipo de capa localmente conectada. Tienen una conexión especial que hace que implementen la operación matemática de convolución. Las neuronas de estas capas se conectan al mismo número de neuronas de la capa anterior y todas comparten los mismos pesos. Estas capas presentan invarianza espacial, es decir, si una capa convolucional se especializa en detectar ruedas, la capa podrá detectar las ruedas en cualquier parte de la imagen. Las capas total y localmente conectadas también pueden llegar a alcanzar la invarianza espacial mediante el entrenamiento, si bien es mucho más complicado ya que requieren más tiempo y más datos para ello. Las convolucionales lo consiguen directamente, gracias a su estructura y sus conexiones,

utilizando también menos parámetros para ello.

La Figura 2.3c muestra una capa convolucional. Se puede apreciar que la principal diferencia con respecto a la Figura 2.3b, es que cada neurona comparte los mismos parámetros (pesos y sesgos). Para realizar la convolución, se puede considerar la entrada como una señal discreta y los pesos de una neurona como los valores de un filtro. Desplazando el filtro invertido sobre la entrada, multiplicando y sumando los elementos que coinciden en cada posición, se obtiene la convolución. Las tres neuronas de la Figura 2.3c realizan el mismo procedimiento sin el desplazamiento, de ahí el nombre de capas convolucionales.

Para comparar el número de parámetros con las otras capas, se va a utilizar otro ejemplo. Se emplea la misma imagen de entrada de $224 \times 224 = 50176$. Ahora se utiliza una capa convolucional, con la mitad de neuronas que de entradas ($50176/2 = 25088$) y con cada neurona conectada a una décima parte de las entradas ($50176/10 \approx 5018$). Se denomina N al número de neuronas de la capa actual y C al número de entradas de cada neurona. Al compartir todas las neuronas los mismos parámetros, se necesitaría almacenar solo los de una neurona, $C + 1 = (5018 + 1) = 5019$ parámetros. Cuatro órdenes de magnitud menos que los de la capa localmente conectada. Utilizando 32 bits para cada parámetro, la capa ocuparía menos de 20 Kibibytes (KiB) en memoria.

Capa reductora (*pooling*): La Figura 2.3d muestra una capa reductora[2]. Estas capas no tienen parámetros que aprender, aplican una función conocida sobre la entrada. En el caso de la Figura 2.3d, se realiza la reducción máxima o "*max pooling*", que funciona dejando pasar los valores máximos de las entradas y descartando el resto. Al reducir el tamaño de los vectores con los que trabaja la red, se disminuye el número de conexiones y por tanto de parámetros que se necesitan. En el caso de trabajar con imágenes también se aumenta la robustez ante la traslación local.

2.3. Redes convolucionales

Las redes convolucionales actuales fueron propuestas en un artículo del investigador Yann LeCun de 1998 [11]. Estas redes están pensadas para funcionar mejor con cierto tipo de datos con estructuras locales, jerárquicas e invariantes a la localización. Las redes convolucionales trabajan en dos dimensiones mediante matrices y conexiones locales a la capa anterior. Esto impone el conocimiento de la estructura espacial de datos como las imágenes directamente desde el principio.

Estas redes se componen de capas totalmente conectadas, convolucionales y reductoras como las ya explicadas. Trabajar en dos dimensiones afecta a las capas convolucionales

[2]Es más común referirse a las capas reductoras con su nombre en inglés: Capas *pooling*. En el resto del trabajo se decide emplear el término traducido para mayor claridad.

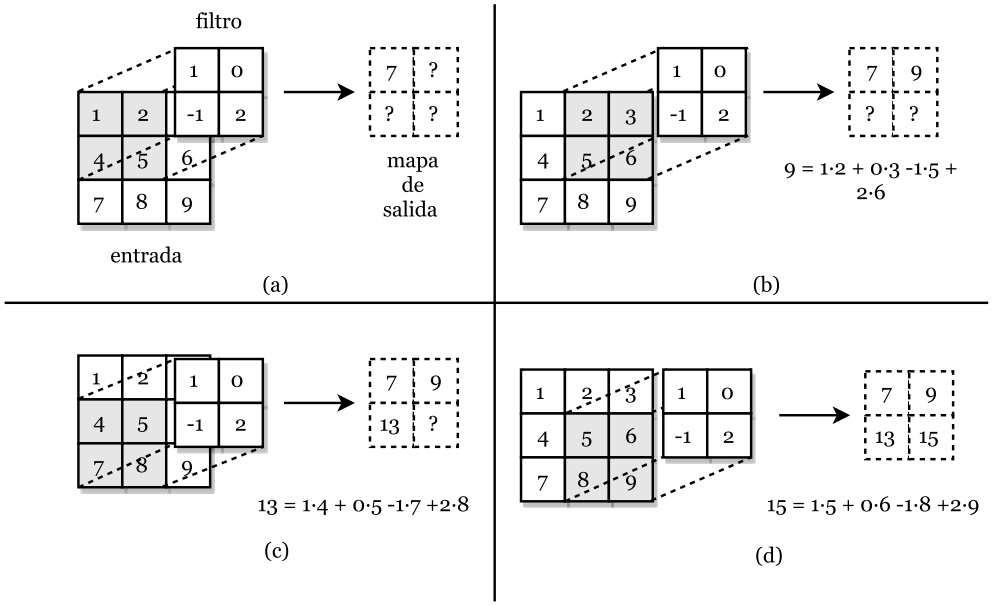


Figura 2.4: Proceso de convolución en dos dimensiones. La matriz de entrada es de tamaño 3×3 y el filtro (matriz con los pesos de la neurona) de tamaño 2×2. La convolución consiste en desplazar el filtro a lo largo y ancho de la matriz de entrada siguiendo los pasos *a*, *b*, *c* y *d*. En cada posición se multiplican los elementos coincidentes y se suman para obtener el valor de la convolución en el mapa de salida.

y reductoras. Para aplicar las capas reductoras sobre matrices, se toman subregiones de la matriz de entrada y se aplica una función sobre ellos, como dejar pasar solo los valores máximos de cada submatriz. Las capas convolucionales se van a explicar con mayor detalle ya que son las más importantes.

Capa convolucional: Al trabajar con dos dimensiones, hay que aplicar la convolución sobre matrices. Se puede ver un ejemplo del proceso en la Figura 2.4. La matriz de entrada pueden ser los valores de los píxeles de una imagen y el filtro es una matriz formada con los pesos de las neuronas. En la Figura 2.4a, la entrada es de 3×3 y el filtro es de tamaño 2×2. Para obtener la salida, se multiplican los elementos coincidentes de la entrada y el filtro y se suman. Desplazando el filtro y repitiendo el proceso a lo largo y ancho de la matriz de entrada (Figuras 2.4b, c y d) se obtiene un valor para cada posición que se almacena en una matriz o mapa de salida. El tamaño del filtro ($k \times k$) es un hiperparámetro que se decide en validación[3], mientras que los valores del filtro se aprenden, puesto que son los pesos de las neuronas que forman la capa.

En el caso de las imágenes, en general no basta con considerar dos dimensiones. Las imágenes RGB, por ejemplo, necesitan tres matrices para ser representadas, una por cada

[3]El proceso de validación se explica en la sección 2.4 de este capítulo

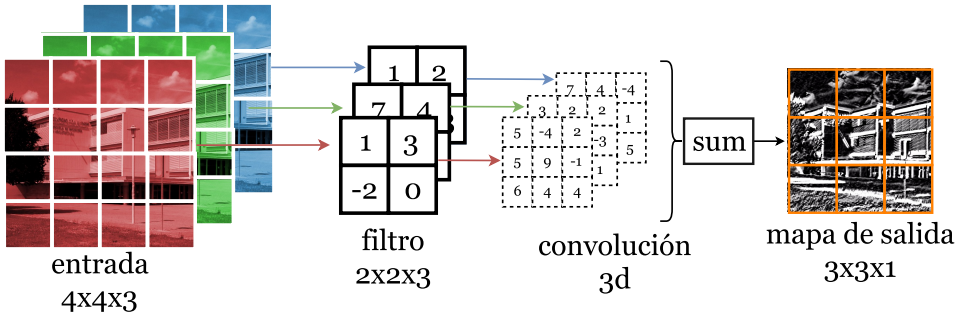


Figura 2.5: Proceso de convolución en tres dimensiones. La imágen de entrada es de tamaño 4×4 y tiene 3 canales de color. El filtro es de tamaño $2 \times 2 \times 3$ puesto que se extiende a los tres canales. La convolución consiste en desplazar el filtro a lo largo y ancho de las 3 matrices de entrada. En cada posición se multiplican los elementos coincidentes y se suman para obtener el valor de la convolución. Se suman todos los elementos para obtener un único mapa de salida.

canal de color (rojo, verde y azul). El proceso de convolución resulta similar solo que aumentando las dimensiones del filtro. Cada neurona se conecta a la misma región local de todas las matrices de entrada. En la Figura 2.5 se puede ver el conjunto de matrices de entrada de una imagen RGB, cada matriz es de tamaño 4×4 por lo que forman un volumen de $4 \times 4 \times 3$. Cada neurona se conecta a la misma región local de tamaño 2×2 las 3 matrices, formando un filtro de tamaño $2 \times 2 \times 3$. Cuando se realiza la convolución se obtiene un mapa de salida de 3×3 por cada canal del filtro, sumando los elementos coincidentes se obtiene un único mapa de salida de 3×3.

2.4. Entrenamiento

El entrenamiento de una red neuronal es el proceso mediante el cual se intenta que la red aprenda a resolver el problema para el cual ha sido diseñada. Consiste en hacer evolucionar los parámetros de las neuronas que forman la red, tratando de obtener los que mejor funcionan. Para ello, además de la propia red, se necesitan varios elementos.

Por un lado, se necesita disponer de un conjunto de datos o "*dataset*". Es la información que será procesada por la red durante el entrenamiento y debe ser un ejemplo representativo de aquello que se quiere resolver. Si se quieren clasificar caras, se deberá disponer de imágenes de caras y de etiquetas que permitan distinguir cada cara para determinar si la red clasifica correctamente. Es recomendable disponer de dos conjuntos de datos: entrenamiento y test. El de entrenamiento sirve para que la red aprenda los parámetros necesarios. Durante el proceso de diseño de la red, se extrae del propio conjunto de entrenamiento una parte de los datos para formar un subconjunto de validación.

Este proceso consiste en modificar los hiperparámetros de la red, buscando los que mejor funcionan con los datos de validación y entrenando con los restantes. Posteriormente se vuelven a unir en un único conjunto de entrenamiento y se comprueba la red final con el conjunto de test. De esta manera se asegura que la configuración de la red no hace sobreajuste a los datos en los que se va a evaluar y aprende de los datos de entrenamiento.

Por otro lado, se necesita la función de coste. Es la función que se quiere optimizar ya que permite determinar numéricamente el error cometido por la red con los datos de entrenamiento. La función de coste tiene dos términos, el de error o "*loss*" y el de regularización. El término de error es generalmente función de la salida de la red y de lo que debería ser su salida para un ejemplo de entrenamiento dado. El término de regularización es función de los parámetros de la red, permite controlar y limitar su evolución, ayudando a evitar el sobreajuste.

El objetivo del entrenamiento es minimizar la función de coste para que la red mejore. Para ello, se obtiene la salida de la red ante uno o varios ejemplos del conjunto de entrenamiento y se calcula el valor de la función de coste. Posteriormente se siguen procedimientos como la propagación hacia atrás o "*backpropagation*". Este procedimiento consiste en calcular la variación de la función de coste en función de los parámetros de la red y transmitir las modificaciones necesarias a lo largo de la red para que el error se reduzca. El entrenamiento consiste en repetir este proceso de forma iterativa con los ejemplos de entrenamiento, tratando de minimizar la función de coste.

Para realizar el entrenamiento hay que ajustar una serie de hiperparámetros por validación. Uno de los que se ha comprobado en este trabajo es el ratio de aprendizaje. El ratio de aprendizaje o "*learning rate*" controla la magnitud de la modificación de los parámetros de la red. Es importante ajustarlo bien porque valores bajos implican un entrenamiento lento y valores demasiado altos pueden llevar a que la red diverja.

Una buena referencia para ver desarrollados con mayor extensión estos y otros conceptos de las redes neuronales es el libro de Goodfellow *et al.* [12].

3. Diseño de la base de datos

El objetivo de este trabajo es conseguir que el reconocedor sea invariante a cambios de apariencia, especialmente a cambios climatológicos y estacionales. Al estar basado en imágenes y redes neuronales, se necesita un conjunto de datos de lugares suficientemente grande y cuya distribución sea similar a la que presenten los lugares reales. Esto permite evaluar el reconocimiento de manera realista y entrenar las redes evitando el sobreajuste, dado el gran número de parámetros de los que disponen.

En este trabajo se han utilizado varios vídeos del trayecto de una línea ferroviaria de Noruega para generar el conjunto de datos necesario.

3.1. Vídeos del trayecto Trondheim - Bodø

En 2012, la compañía de radiodifusión noruega (NRK) realizó un documental [13] del trayecto ferroviario entre las ciudades de Trondheim y Bodø. Grabaron los 729 kilómetros del viaje con una cámara ubicada en la cabina del tren. Las secuencias fueron tomadas en invierno, primavera, otoño y verano, con una duración aproximada de diez horas cada una. Registraron las coordenadas a lo largo de cada viaje con un GPS, pudiendo determinar el instante al que acudir para observar la misma ubicación en los cuatro vídeos.

Los aspectos a considerar para trabajar con estos datos son los siguientes:

- El tren se detiene en varias paradas a lo largo del recorrido. Hay que detectarlas para evitar que haya más imágenes de un lugar determinado que del resto de lugares.
- El tren atraviesa varios túneles en los que la visibilidad del vídeo es nula por la oscuridad. Hay que filtrarlos para evitar que la red se entrene con ejemplos extraños.
- En la toma de datos, se capturó un dato de GPS por segundo, a diferencia del vídeo que fue grabado a 25 imágenes por segundo. Para disponer de coordenadas en todo instante, o bien se interpolan las coordenadas, o bien se toma solo una imagen de cada 25.

- La señal del GPS se perdía durante el trayecto por lo que los datos registrados presentan saltos. Hay que detectarlos e interpolar las coordenadas en esos tramos para obtener un mapa continuo del recorrido.

- Los vídeos están sincronizados con respecto al vídeo original del viaje en verano. Basta con observar el mismo instante temporal para ver la misma localización en los otros tres vídeos.

3.2. Extracción del conjunto de datos

El primer paso para crear el conjunto de datos es extraer el máximo número posible de imágenes de los vídeos, asegurando que no contengan túneles ni paradas. El procedimiento ha sido el siguiente:

- Se descargan los vídeos y se observa que cada vídeo tiene una duración de 9 horas, 53 minutos y 27 segundos (35607 segundos). Al estar grabados a 25 imágenes por segundo, cada vídeo contiene 890175 imágenes.

- Se necesita extraer imágenes del mismo lugar en cada vídeo. Gracias a la sincronización basta con tomar el mismo instante. Posteriormente se pueden filtrar elementos como las paradas gracias a los datos del GPS.

- Se descargan los datos del GPS. Vienen en un fichero en el que aparece la latitud, longitud, velocidad y el segundo exacto en el que se tomó el dato. Sin embargo, aparecen solo 33974 datos en el fichero, en lugar de los 35607 que deberían si fueron tomados a un dato por segundo. Esto se debe a la pérdida de la señal en partes del trayecto.

- Se corrigen los datos del GPS interpolando las coordenadas que corresponderían en cada hueco. Se consigue emparejar cada segundo del vídeo con la coordenada en la que fue tomado.

- Para filtrar las paradas se usan los datos del GPS y se eliminan los lugares en los que la velocidad del tren es nula.

- Para filtrar los túneles se eliminan los instantes en los que se supera un umbral de oscuridad determinado en el vídeo.

- Después de este proceso quedan 28865 segundos de los 35607 originales que no contienen túneles ni paradas. Por lo que se pueden extraer hasta $28865 \times 25 = 721625$ imágenes de cada vídeo. En este trabajo solo se trabaja con las 28865 por motivos de tamaño y cómputo pero se extraen las 721625 para hacerlas disponibles públicamente.

3.3. División del conjunto de datos

Las imágenes de los vídeos del conjunto Nordland han sido utilizadas por otros grupos de investigación en sus propios métodos de reconocimiento de lugares como [7] y [14]. Cada grupo utiliza las imágenes de manera distinta, evaluando y entrenando sus algoritmos en tramos diferentes de los vídeos. Esto dificulta la comparación de resultados y conclusiones entre métodos. Por ello, en este trabajo se propone una división concreta del conjunto para que los resultados sean directamente comparables y se trabaje en igualdad de condiciones.

Como se menciona en la sección 2.4 del capítulo 2, se necesita disponer de un conjunto de entrenamiento y otro de test para entrenar las redes neuronales. En el reconocimiento de lugares existen algoritmos que tienen en cuenta el lugar visitado anteriormente para buscar entre los más cercanos primero. Si se dispone de un recorrido continuo como en este caso, conviene respetar cierta agrupación en el conjunto de test. Sin embargo, si se toma un único tramo demasiado grande, el conjunto puede carecer de variedad tanto en las condiciones climatológicas como en la zona atravesada por el tren en esos puntos.

Teniendo esto en cuenta, el procedimiento seguido para elaborar los conjuntos ha sido el siguiente:

- Se eliminan las 45 imágenes iniciales del recorrido pues se observa que los trenes comienzan en vías diferentes en cada vídeo. Se corresponden con la primera zona azul del gráfico de la izquierda de la Figura 3.1, todavía en la ciudad de Trondheim.
- Se decide utilizar tres tramos del recorrido total para el conjunto de test. Para asegurar que la red generaliza a datos nuevos, es necesario que los datos de entrenamiento y los de test se diferencien. Por ello, se decide que entre los tramos de test y los de entrenamiento haya cierta separación. De esta manera, los lugares capturados en cada tramo tendrán menos características en común. Se establece una separación de 200 segundos entre cada tramo de test y de entrenamiento. Los segundos de la separación se descartan y no se utilizan, se corresponden con los tramos azules del gráfico de la izquierda de la Figura 3.1.
- Para utilizar el mínimo número de separaciones dos de los tramos se ubican al principio y al final. De esta forma se asegura también que se diferencian al ser zonas lejanas. Los dos tramos se corresponden con el primer y el último tramo amarillo del gráfico de la izquierda de la Figura 3.1.
- El tercer tramo se ubica en el medio del conjunto de datos. Se corresponde con el tramo medio amarillo en el gráfico de la izquierda de la Figura 3.1. Se opta por que cada uno de los tres tramos contenga 1150 imágenes, el conjunto de test está formado en total por 3450 imágenes de cada vídeo.
- Para el entrenamiento se utilizan las 24569 imágenes restantes de cada vídeo. Se

corresponden con los dos grandes tramos rojos del gráfico de la izquierda de la Figura 3.1.

- Para validación se separan durante el diseño 2600 imágenes aleatorias de cada trayecto del conjunto de entrenamiento. No se muestran en la Figura 3.1 puesto que ya se incluyen entre las de entrenamiento.

Se propone la división de los conjuntos de entrenamiento y test explicados anteriormente como estructura básica del conjunto Nordland. De esta forma, todo aquel que quiera trabajar con el reconocimiento de lugares pueda comparar su trabajo con el de otros fácilmente. Se puede ver el resultado general de la división y la referencia visual geográfica del trayecto en la Figura 3.1. Las imágenes procesadas y divididas serán publicadas una vez finalizado este trabajo.

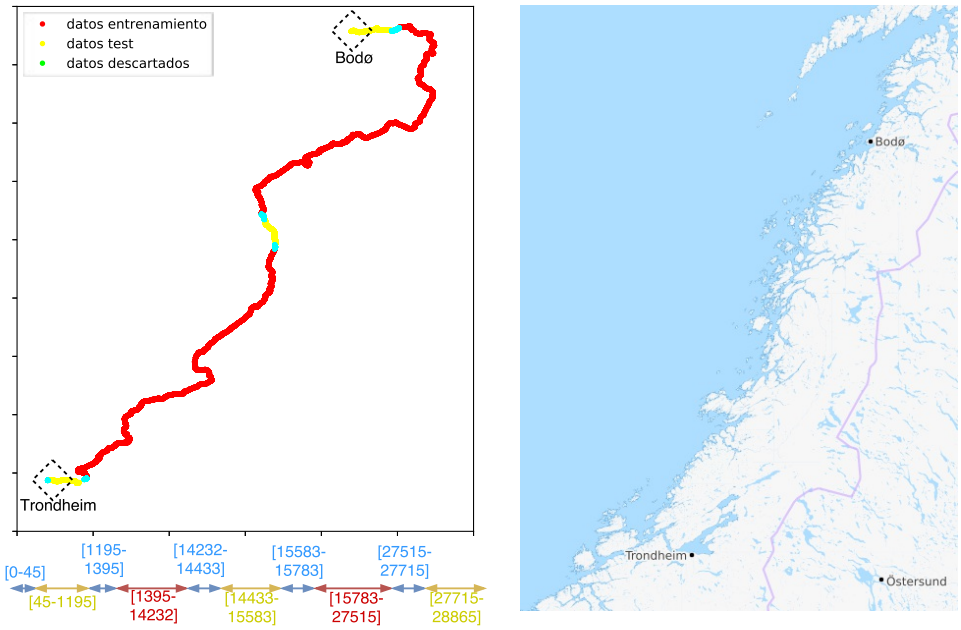


Figura 3.1: División propuesta del conjunto de datos Nordland. **Izquierda (encima):** Representación de la distribución con las coordenadas de cada lugar. **Izquierda (debajo):** Índices de separación de cada tramo. **Derecha:** Mapa extraído de Open Street Map [15] de la región de Noruega entre Trondheim y Bode.

3.4. Lugares en el conjunto de datos

Hace falta definir aquello que se considera un lugar para poder usar el conjunto con el reconocedor. Un lugar se puede entender como una zona geográfica cuya extensión no

está limitada por definición, sino por el contexto. Si se quiere navegar por un edificio, una habitación puede ser un lugar. Si se quiere navegar por una habitación, cada parte de la misma (p. ej.: un escritorio o un armario) puede ser un lugar diferente.

En la Figura 3.2 se pueden apreciar ocho imágenes extraídas del conjunto de datos creado. Las ocho imágenes se tomaron consecutivamente con un segundo de diferencia, hay un cierto desplazamiento del tren y el paisaje cambia. Una opción sería considerar cada imagen como un lugar diferente, sin embargo, los cambios del paisaje no son abruptos sino incrementales y suaves. En la Figura 3.2 se puede apreciar que hasta la aparición del puente en la quinta imagen, el paisaje apenas cambia.

Cuando se utilizan imágenes de trayectos continuos, un parámetro de diseño del reconocedor es el número de imágenes consecutivas que se consideran un mismo lugar. A este parámetro se le conoce como ventana. En este trabajo se estima que la apariencia de cinco imágenes consecutivas es suficientemente similar como para considerarlas un único lugar. Por ello se decide utilizar una ventana con las imágenes tomadas en cinco segundos consecutivos.

Para entender el funcionamiento de la ventana se puede seguir la Figura 3.3. Se establece la ventana y se agrupan las imágenes que están dentro de ella para formar el primer lugar. A continuación, se desplaza la ventana una única posición y se agrupan las imágenes formando el segundo lugar, siguiendo este proceso sucesivamente. Es importante apreciar que parte de las imágenes de un lugar, forman parte del lugar anterior y posterior.

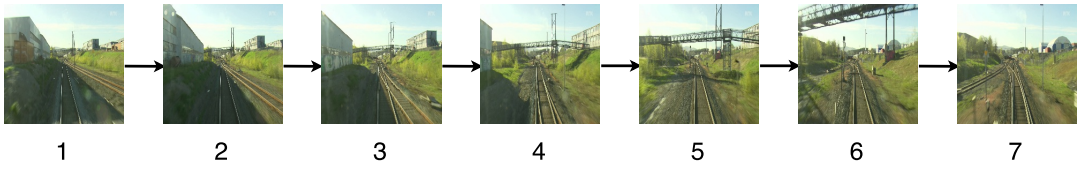


Figura 3.2: Siete imágenes consecutivas del conjunto de datos. Las imágenes fueron tomadas en el trayecto de primavera. La figura se puede apreciar mejor en color y en formato electrónico.

En este conjunto de datos se dispone de imágenes de la misma ubicación en cuatro estaciones distintas. La ventana de cinco se aplica por igual a las imágenes de las cuatro estaciones, como se puede ver en la Figura 3.4.

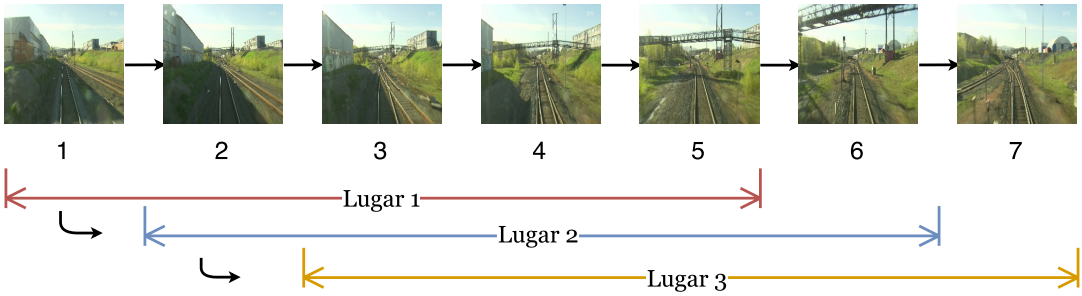


Figura 3.3: Ejemplo del funcionamiento de la ventana de agrupación de imágenes para formar lugares con siete imágenes consecutivas. La ventana agrupa cinco imágenes consecutivas y se va desplazando para agruparlas y formar los lugares. La figura se puede apreciar mejor en color y en formato electrónico.

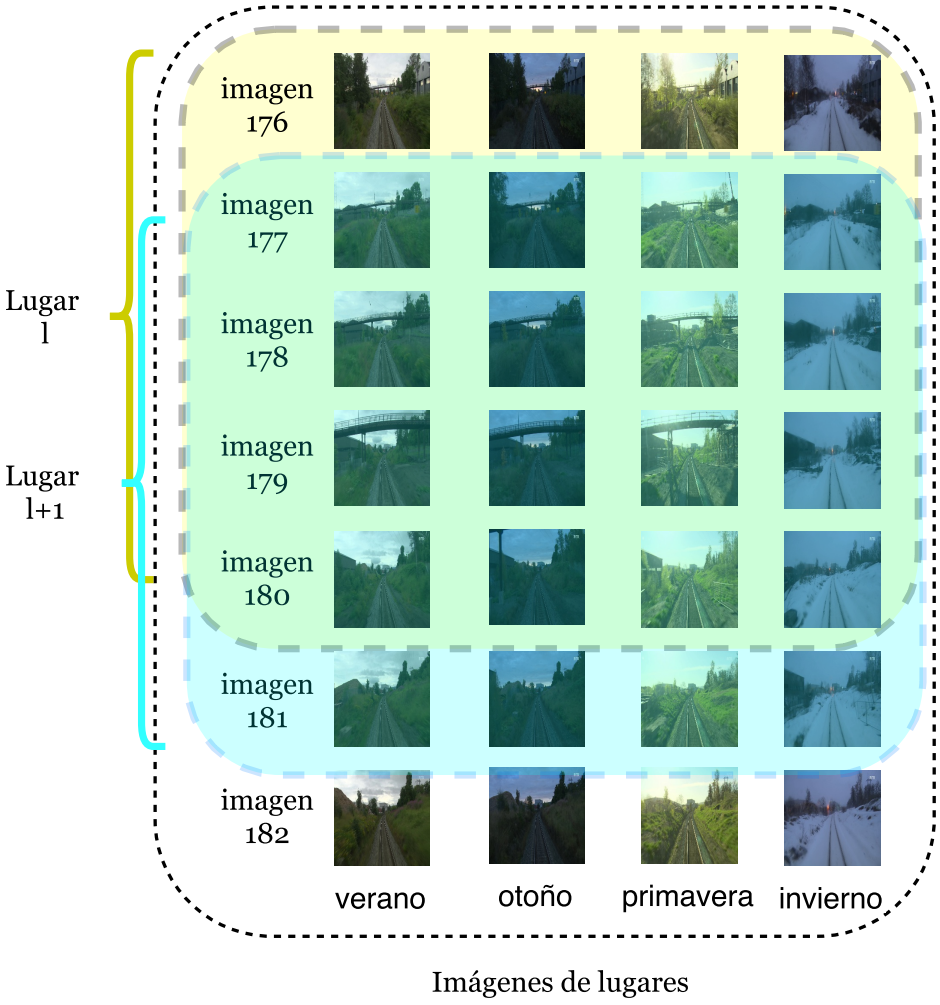


Figura 3.4: Ventana de agrupación de las imágenes del conjunto final de las cuatro estaciones. La ventana forma lugares con cinco imágenes consecutivas de cada estación. La figura se puede apreciar mejor en color y en formato electrónico.

4. Redes neuronales propuestas

En la Figura 4.1 se pueden apreciar los bloques que componen el reconocedor propuesto. El sistema comienza procesando la base de datos con imágenes de los lugares (que ya se han visitado) y la imagen del lugar a reconocer. Este procesamiento se realiza utilizando redes neuronales para extraer descriptores de las imágenes. Un descriptor es un elemento, en este caso formado por un vector o conjunto de valores numéricos, que almacena información relativa a otro elemento. Los descriptores suelen presentar propiedades especiales, como tener menor dimensión que el elemento del que se extraen. En este caso, se considera descriptor al vector de salida de la red neuronal obtenido al procesar imágenes de lugares. El descriptor extraído del lugar que se quiere reconocer se puede comparar con los descriptores extraídos de la base de datos. La comparación se realiza calculando una métrica determinada (p.ej.: la distancia euclídea o la similitud coseno) entre los descriptores que permite encontrar los que más se parecen. Utilizando la información de la comparación, se puede determinar el lugar que el sistema cree haber reconocido.

Hay que desarrollar el método de utilización de las redes para extraer los descriptores de las imágenes y el método para compararlos.

Se estudian tres formas de utilizar las redes. En primer lugar, se evalúa una red que ya ha sido entrenada para clasificación de categorías de objetos, un problema diferente al reconocimiento de lugares. Posteriormente, se tratará de entrenar la red para el problema concreto del reconocimiento, utilizando estructuras particulares denominadas siamesas y triplets. Se explican los tres métodos en las secciones 4.1, 4.2 y 4.3.

Para la comparación se decide utilizar la distancia euclídea entre descriptores. Entre los descriptores que representan el mismo lugar debe haber menos distancia que entre los de lugares distintos. Con la información de los lugares más parecidos se puede realizar un procesamiento final, filtrar el resultado y elegir el lugar reconocido.

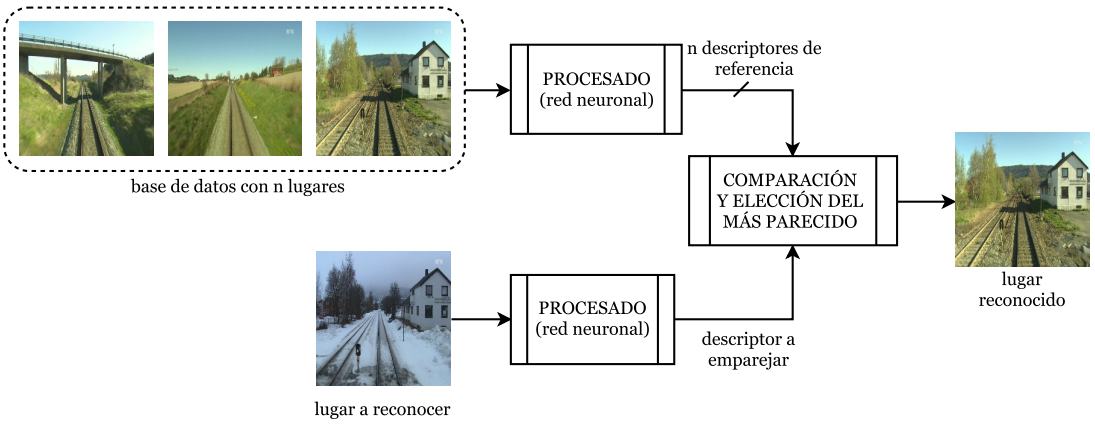


Figura 4.1: Estructura del reconocedor propuesto.

4.1. Redes neuronales pre-entrenadas

Entrenar una red neuronal es un proceso complejo. Cuando se requieren resultados competitivos, se pueden necesitar recursos económicos o temporales exigentes. Una alternativa es utilizar redes pre-entrenadas para resolver problemas diferentes. Aunque la red haya sido pre-entrenada para clasificación, es posible darle otros usos. Las redes procesan los datos de entrada de forma jerárquica internamente, obteniendo información cada vez más compleja capa a capa. Se puede utilizar la salida de las capas internas de la red como descriptores de imágenes que permiten resolver problemas diferentes. En [6], Niko Sünderhauf *et al.* analizaron el funcionamiento de los descriptores extraídos por varias redes pre-entrenadas en el reconocimiento de lugares.

En este trabajo se decide utilizar la red VGG-16 para extraer los descriptores del sistema reconocedor. Esta red fue desarrollada por el Oxford Geometry Group [16] para clasificar objetos.

Una vez elegida la red a utilizar, hay que estudiar las capas que mejor funcionan. Se decide estudiar las salidas de la cuarta y la quinta capas reductoras ($pool4$ y $pool5$). También de la primera y la segunda capas totalmente conectadas ($fc6$ y $fc7$). En la Figura 4.2 se muestran las capas que se decide utilizar y las que se descartan. Las capas previas a la cuarta capa reductora ($pool4$) se descartan por su excesivo tamaño. Hasta la cuarta capa reductora, el vector que se obtiene transformando la salida de las capas tiene dimensiones superiores a 100352. Calcular las distancias euclídeas con tantas dimensiones es costoso computacionalmente. Las capas posteriores a la segunda capa totalmente conectada ($fc7$) se descartan porque están demasiado cerca de la salida. Esto hace que extraigan información más específica del problema original para el que fue entrenada la red.

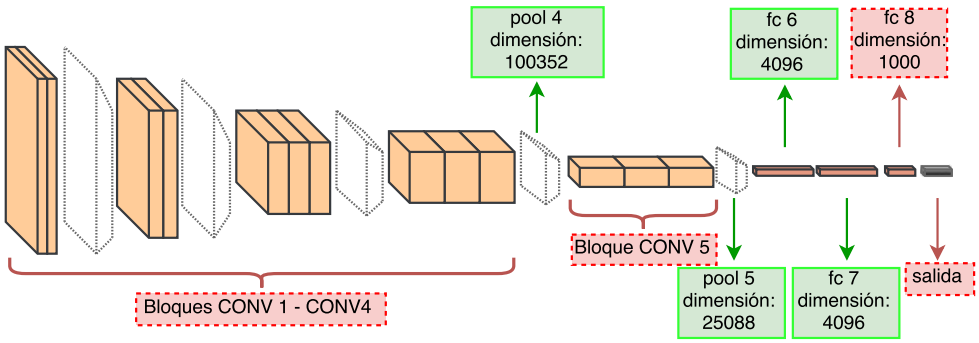


Figura 4.2: Capas utilizadas de la red VGG-16. *Rojo y borde discontinuo:* Capas no utilizadas. *Verde y borde continuo:* Capas utilizadas.

Las capas convolucionales que se encuentran entre las dos reductoras utilizadas tampoco se tienen en cuenta. Se descartan porque los resultados que se obtienen con ellas en las primeras pruebas son intermedios entre los resultados de las dos reductoras.

4.2. Red siamesa

En la sección 4.1 se explica la utilización de una red neuronal pre-entrenada. Utilizando sus capas intermedias se pueden procesar imágenes para obtener un vector capaz de describir imágenes. El siguiente paso es evolucionar la red para que los descriptores extraídos funcionen mejor para el problema concreto del reconocimiento de lugares desarrollado en este trabajo.

El sistema funciona midiendo las distancias entre los descriptores. Se necesita entrenar una red que permita reducir las distancias entre imágenes del mismo lugar y que aleje las de lugares distintos. Existe un tipo de red que se encarga de llevar las imágenes a espacios vectoriales donde las distancias entre vectores son relevantes; las denominadas redes siamesas.

4.2.1. Introducción a las redes siamesas

La primera estructura siamesa propuesta [17] trataba de resolver el problema de verificar si una firma es falsa a partir de dos imágenes, una de ellas de la firma verdadera. Una red siamesa es un tipo de red neuronal formado por dos copias idénticas de una misma red. Cada red tiene su propia entrada pero ambas tienen la misma estructura y comparten los pesos internos de sus neuronas. Al procesar su entrada, cada red obtiene un vector de salida pero al ser dos redes idénticas, se obtendría el mismo vector de salida si las entradas fueran iguales.

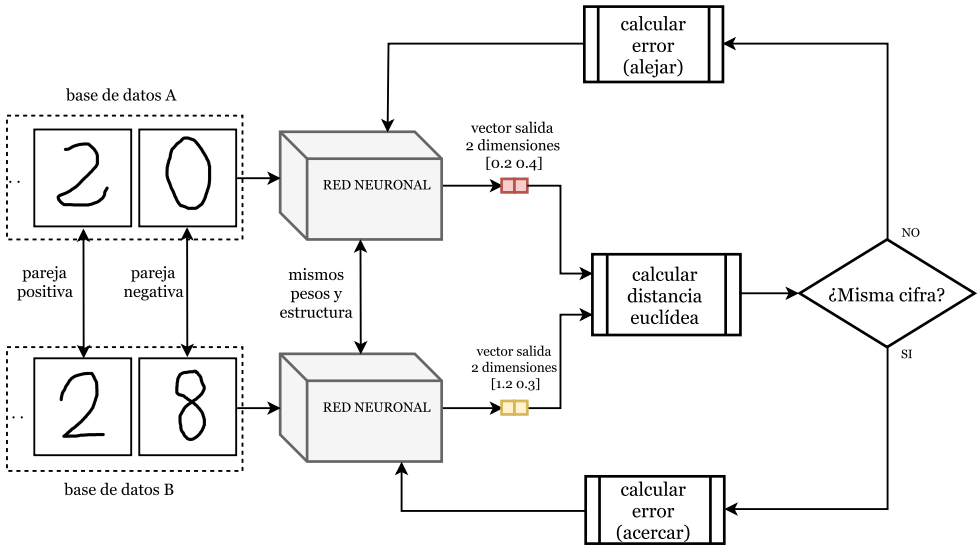


Figura 4.3: Diagrama de bloques del entrenamiento de una red siamesa para determinar si dos cifras son la misma. Si las dos cifras son idénticas, el entrenamiento trata de acercarlas. Si son diferentes, el entrenamiento trata de alejarlas.

Se puede ver el funcionamiento de una red siamesa en la Figura 4.3. Este ejemplo trata de determinar si dos cifras escritas a mano son la misma cifra. Cuando una de las dos cifras es distinta, como el cero y el ocho, se trata de una pareja negativa. Se procesan las dos imágenes y se calcula la distancia euclídea entre los vectores de salida. Como son distintas cifras, la distancia debería ser mayor que si fueran la misma. Durante el entrenamiento se modifican los parámetros de la red para alejar los ejemplos negativos y acercar los positivos.

Otro aspecto importante de las redes siamesas es la función de error que utilizan. En una red siamesa, el error cometido depende de las distancias entre los vectores que salen de la red. Una de las funciones de error más usadas es la función contrastiva o "*contrastive loss*" propuesta en [18]. Su fórmula matemática es la siguiente:

$$E = (y)d^2 + (1-y)max(margen - d, 0)^2 \tag{4.1}$$

Donde E es el error, y es la etiqueta que determina si la pareja de datos de entrada es positiva o negativa, d es la distancia euclídea de los vectores de salida de la red y el $margen$ es un parámetro que permite determinar la distancia mínima que debe haber entre parejas negativas.

En el caso de las parejas positivas ($y = 1$), el error es directamente la distancia al cuadrado. En el caso de las negativas ($y = 0$), el error solo tiene valor cuando es mayor que el margen. Minimizar el error E implica que todas las parejas positivas se intentan acercar, mientras que solo se intentan alejar aquellas parejas negativas que estén

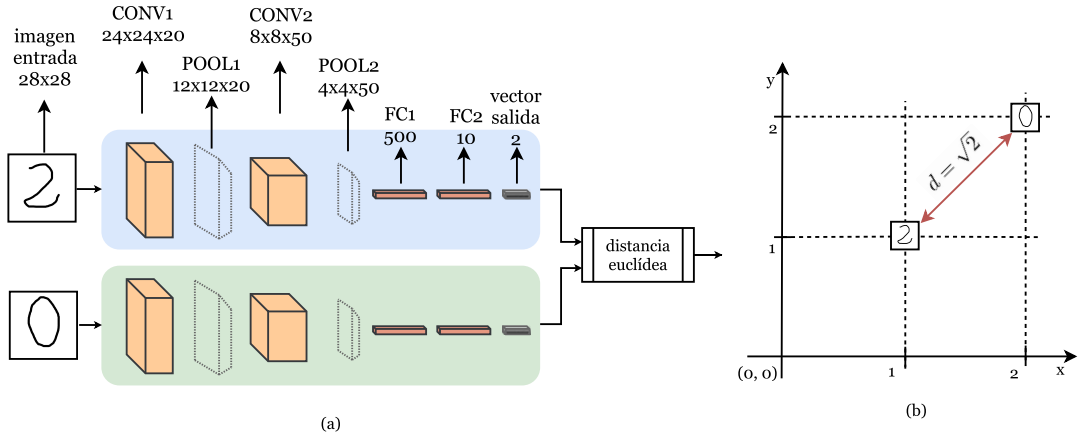


Figura 4.4: Red siamesa de ejemplo y representación de los descriptores. **a:** Red siamesa del paquete de ejemplos de Caffe. La red cuenta con dos capas convolucionales (CONV), dos reductoras (POOL) y dos totalmente conectadas (FC). **b:** Representación de los descriptores obtenidos al procesar las dos imágenes. Primer descriptor en el eje x y segundo en el eje y.

demasiado cerca. Este margen de separación afecta a la evolución del entrenamiento por lo que hay que elegirlo cuidadosamente.

4.2.2. Prueba de concepto con redes siamesas

Antes de aplicar la estructura siamesa al reconocimiento de lugares se va a trabajar con un problema más sencillo, el reconocimiento de cifras. De esta forma se pueden estudiar las redes siamesas de forma aislada, sin la propia complejidad del reconocimiento de lugares.

Para ello se utiliza el conjunto de datos MNIST [19], formado por 70000 imágenes de cifras del cero al nueve escritas a mano. La red siamesa a entrenar viene con el paquete de ejemplos de Caffe, el entorno de desarrollo de redes neuronales utilizado en este trabajo. Se trata de una red más sencilla y con menos capas que la red VGG-16, se puede ver su estructura en la Figura 4.4a. La entrada de la red son las imágenes de las cifras de tamaño 28×28, cuenta con 6 capas ocultas y la salida es un vector de 2 dimensiones.

En la Figura 4.4b se han representado los descriptores extraídos al procesar la imagen de un dos y un cero. Se pueden representar como si fueran puntos en un eje de coordenadas porque tienen dos dimensiones. El descriptor obtenido de la cifra dos es [1, 1] y el de la cifra cero es [2, 2], por lo que la distancia euclídea entre ambos es $\sqrt{2}$. Si se procesaran más imágenes de ceros, los descriptores obtenidos deberían estar más cerca del [2, 2] que del [1, 1].

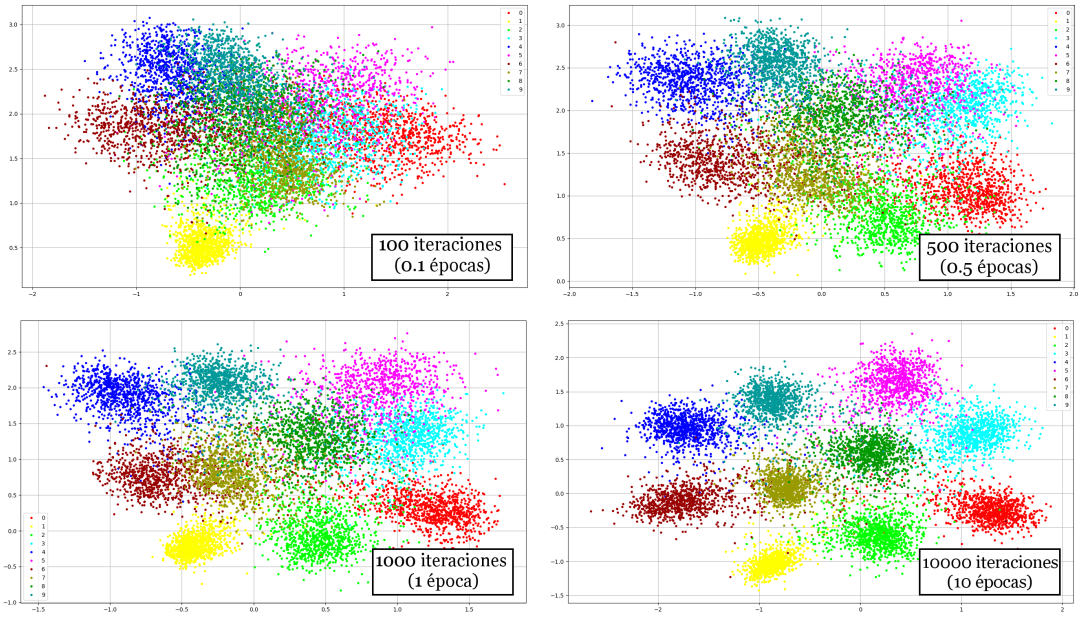


Figura 4.5: Evolución con el entrenamiento de los descriptores extraídos con el conjunto MNIST. Cada color representa el tipo de cifra entre 0 y 9. Se presenta el resultado para 0.1, 0.5, 1 y 10 épocas. La figura se aprecia mejor en color.

El objetivo del entrenamiento de esta red es que las cifras distintas estén lejos y las cifras iguales estén cerca. Se entrena la red neuronal con 60000 de las imágenes de cifras durante varias épocas [1]. Conforme avanza el entrenamiento se procesan las 10000 imágenes de cifras restantes y se representan los descriptores obtenidos. Se puede ver el resultado en la Figura 4.5.

En el gráfico de la esquina superior izquierda de la Figura 4.5, el entrenamiento acaba de comenzar y los descriptores de las cifras están mezclados. A medida que progresa, la red trata de acercar los descriptores de cifras iguales. En el gráfico de la esquina superior derecha de la Figura 4.5 se aprecia que al acercarse, comienzan a formarse grupos. La red también trata de alejar los descriptores de cifras distintas por lo que los grupos que se forman se alejan a su vez de los grupos de cifras diferentes. Al final del entrenamiento, como se puede ver en el gráfico de la esquina inferior derecha de la Figura 4.5, los descriptores correspondientes a la misma cifra se encuentran agrupados, formando grupos claramente diferenciables.

[1] Durante el entrenamiento de una red neuronal, se considera que ha pasado una época cuando el algoritmo ha procesado todos los ejemplos de entrenamiento al menos una vez. En este caso una época implica que se han procesado las 60000 imágenes.

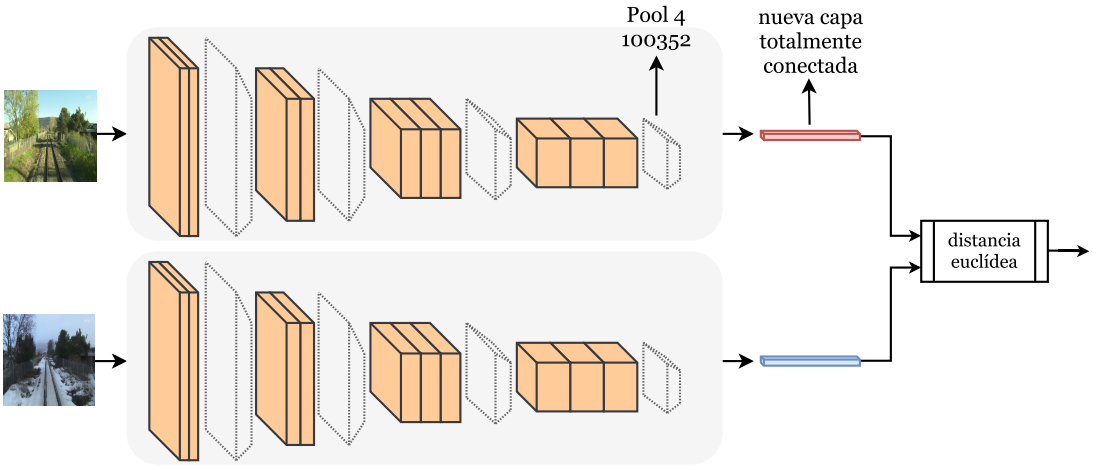


Figura 4.6: Red siamesa utilizada para el reconocimiento de lugares. Se marcan en gris los bloques de la red VGG-16 pre-entrenada, cuyos parámetros no se modifican. La capa totalmente conectada añadida tiene dimensión n.

4.2.3. Aplicación al reconocimiento de lugares

Se pretende mejorar los resultados obtenidos al utilizar la red VGG-16 pre-entrenada. Se modifica la red para utilizar una estructura siamesa. El objetivo es obtener descriptores más robustos e invariantes ante cambios de apariencia de las imágenes de los lugares.

La capa de la red pre-entrenada que mejores resultados ofrece es la cuarta capa reductora como se verá explicado en el capítulo 5. Se decide añadir una nueva capa totalmente conectada a partir de esa capa, se puede ver la estructura de la red en la Figura 4.6. Los parámetros de la nueva capa se aprenden utilizando la estructura siamesa y el resto de capas de la red no se modifican durante el entrenamiento. Esto se hace por dos motivos. Por un lado, se puede comprobar si es posible mejorar los resultados sin riesgo de estar empeorando el funcionamiento de la propia red. Por otro lado, se puede comprobar la capacidad de generalización de las características ya aprendidas en la red VGG-16.

Los descriptores extraídos de la cuarta capa reductora tienen una dimensión de 100352. Se estudian varias dimensiones (128, 256, 512 y 1024) del vector de salida de la nueva capa totalmente conectada para reducir el tamaño y mejorar la funcionalidad. Utilizando 128 dimensiones, por ejemplo, se reduce hasta 784 veces el tamaño de los descriptores originales. En cuanto a la velocidad, utilizar 128 dimensiones en lugar de 100352 permite reducir en más de dos órdenes de magnitud el tiempo de ejecución del algoritmo de reconocimiento.

4.2.4. Bases de datos siamesas de lugares

Además de decidir el modelo de red a utilizar, hay que preparar los ejemplos de entrenamiento. Para entrenar la red siamesa se necesitan parejas de imágenes positivas (imágenes que representan el mismo lugar) y negativas (imágenes que representan lugares distintos). Dada la ventana utilizada, cada lugar está formado por cinco imágenes consecutivas del trayecto en las cuatro estaciones del año. Tras emparejar todas las imágenes que forman cada lugar se obtienen 834746 parejas de imágenes positivas.

Las parejas negativas son más abundantes, ya que se puede emparejar cualquier lugar con un lugar distinto. Pese a ello, se decide emparejar el mismo número de lugares distintos que de lugares iguales, 834746. La simetría de parejas positivas y negativas en el entrenamiento ayuda a que la red no favorezca alejar lugares frente a acercarlos.

Se crean dos bases de datos para introducir las parejas positivas y negativas. En cada base se introduce un ejemplo diferente de las parejas de forma secuencial. Primero una positiva y luego una negativa, sucesivamente. Se puede ver un ejemplo de la estructura de la dos bases en la Figura 4.7.

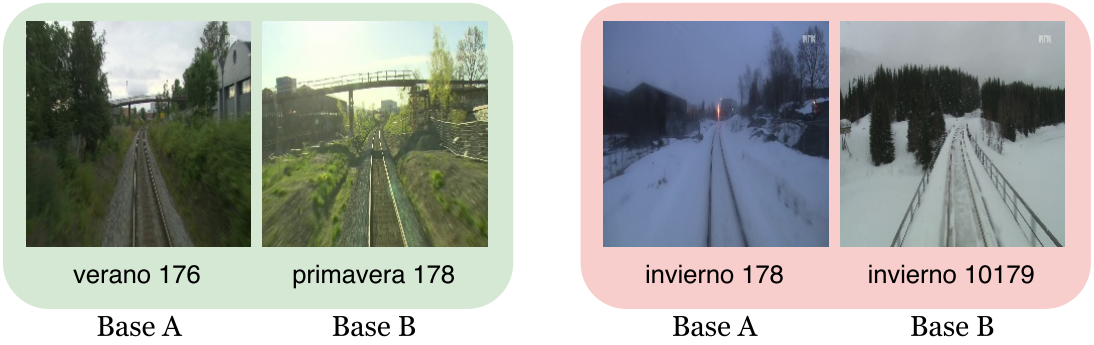


Figura 4.7: Bases de datos creadas para las redes siamesas. En verde se marca un ejemplo de pareja positiva y en rojo una negativa. En la base de datos A se introduce el primer ejemplo de la pareja. En la base de datos B se introduce el segundo ejemplo de la pareja.

4.3. Redes triplets

Las redes triplets[2] tienen un funcionamiento similar al de las siamesas pero suelen ofrecer mejores resultados. Gómez-Ojeda *et al.* [7] ya entrenó redes triplets con el objetivo de reconocer lugares, por lo que va a ser un punto de referencia para este trabajo.

[2]La traducción literal del término en inglés es redes trillizas. Dado que no se han encontrado ejemplos que utilicen esa nomenclatura en español, se ha decidido utilizar el anglicismo en el resto del trabajo.

Se decide utilizar redes triplets con el mismo objetivo que las siamesas, mejorar los descriptores extraídos por la red VGG-16 pre-entrenada.

4.3.1. Introducción a las redes triplets

Una de las primeras estructuras triplets propuestas fue [20]. Trataba de resolver el problema de clasificación de grano fino. Para ello propusieron las redes triplets, que transformaban las imágenes a espacios vectoriales donde se podía medir el parecido entre ellas.

Uno de los problemas de las redes siamesas es que, durante su entrenamiento, trabajan sucesivamente con un ejemplo positivo y después con uno negativo. Puede darse el caso de que al tratar de acercar imágenes similares, los parámetros evolucionen de tal manera que parte de las negativas también se acerquen. También puede suceder que al alejar las negativas, se alejen parte de las positivas.

Las redes triplets trabajan a la vez con el ejemplo positivo y el negativo. Estas redes están formadas por tres copias de la misma red. Tienen tres entradas, dos ejemplos que se consideren similares junto a uno que se considere distinto. La red trata de evolucionar sus parámetros para acercar unos y alejar otros a la vez. De esta forma, se favorece que la evolución de los parámetros no acerque ni aleje los ejemplos que no se desean. Se puede ver el funcionamiento en la Figura 4.8. Como las tres redes son idénticas, se sigue teniendo un único conjunto de parámetros.

4.3.2. Aplicación al reconocimiento de lugares

El objetivo es obtener descriptores más robustos e invariantes ante cambios de apariencia de las imágenes de los lugares. Se pretende mejorar tanto los resultados obtenidos al utilizar la red VGG-16 pre-entrenada, como al utilizar las redes siamesas. Para ello se modifica la red VGG-16 para utilizar una estructura triplet.

Al igual que con las siamesas, se decide añadir una nueva capa totalmente conectada a partir de la cuarta capa reductora de la red VGG-16. Los parámetros de la nueva capa se aprenden utilizando una estructura triplet y el resto de capas de la red no se modifican durante el entrenamiento inicialmente. Se estudian varias dimensiones (128, 256, 512 y 1024) del vector de salida de la nueva capa totalmente conectada.

Una vez elegida la dimensión que resulta óptima, se repite el entrenamiento modificando también el resto de parámetros de las otras capas de la red. De esta forma se podrá comprobar si las características extraídas por las capas previas de la red pueden adaptarse mejor para reconocer lugares. Se decide realizar el ajuste fino[3] las capas pre-

[3]El proceso de ajuste fino o "*fine-tuning*" de una red neuronal consiste en entrenar la red de nuevo con otro objetivo, partiendo de los parámetros que la red ya presenta tras su entrenamiento original.

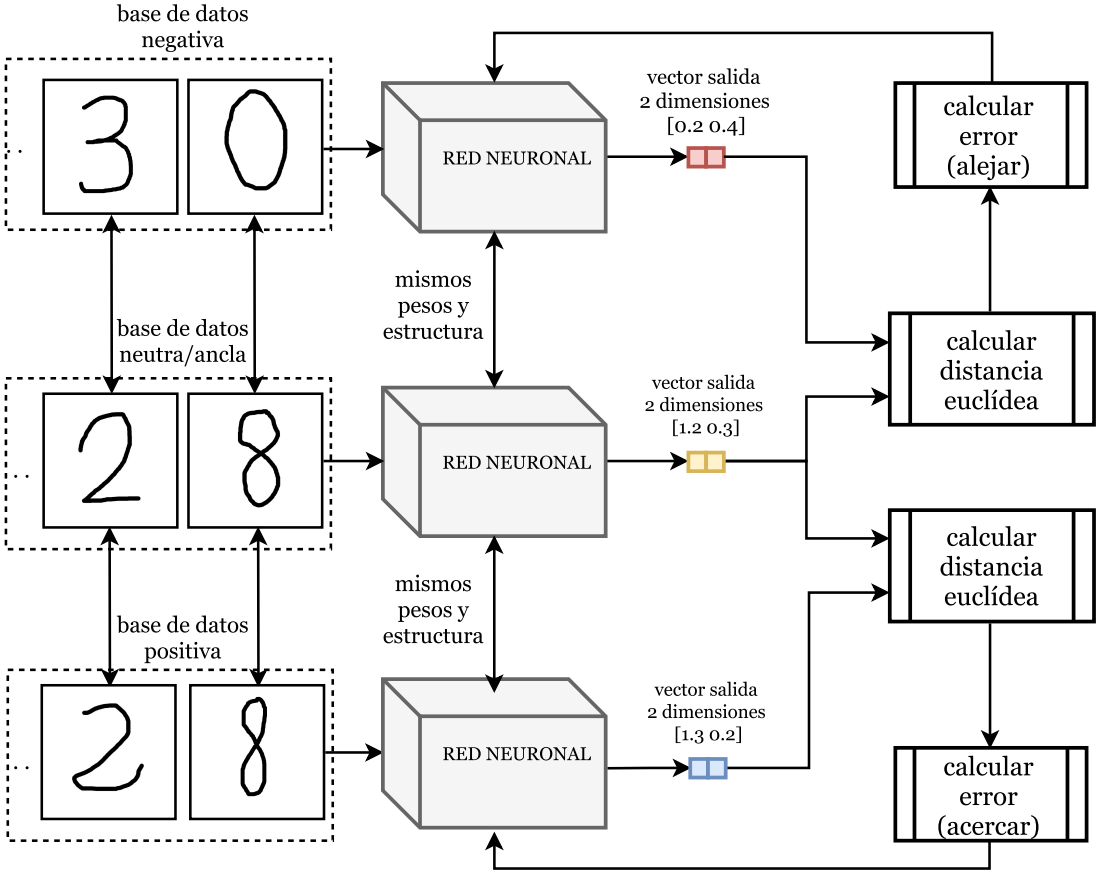


Figura 4.8: Diagrama de bloques del entrenamiento de una red triplet.

vias a la totalmente conectada salvo el primer bloque convolucional. Esto se hace porque las características extraídas por las primeras capas convolucionales son más genéricas. Modificarlas para que se especialicen en este problema puede empeorar el resultado.

Para entrenar las redes triplets se necesita una función de error que dependa de la distancia entre los ejemplos positivos y negativos a la vez. En este trabajo se decide entrenar la red con las siguientes funciones de error:

1. Función de error triplet de Wohlhart-Lepetit: Esta función es la utilizada por el grupo de investigación de Málaga que ha trabajado usando triplets en el reconocimiento de lugares [7]. Fue propuesta originalmente por Wohlhart y Lepetit [21] y su fórmula es la siguiente:

$$E = max\left\{0, 1 - \frac{d_n}{margen + d_p}\right\} \tag{4.2}$$

Donde E es el error, d_p es la distancia entre los vectores de la pareja positiva, d_n la distancia con el ejemplo negativo y el *margen* es el parámetro que permite limitar la diferencia entre las dos distancias.

En esta función, el error es nulo cuando la distancia negativa supera a la positiva por cierto margen. Además, el error se encuentra siempre entre 0 y 1, lo cual puede tener efectos positivos sobre la estabilidad de la evolución del entrenamiento.

2. Función de error triplet propuesta: Esta función se propone como alternativa a la anterior en este trabajo. Su fórmula matemática es la siguiente:

$$E = \frac{1 + d_p}{1 + d_n} \tag{4.3}$$

Donde E es el error, d_p es la distancia entre los vectores de la pareja positiva y d_n la distancia con el ejemplo negativo.

Las distancias positivas se encuentran en el numerador y las negativas en el denominador. Esto hace que la función penalice que las parejas de lugares iguales sean mayores que las de lugares distintos. Se les suma un 1 a los términos de numerador y denominador para evitar problemas numéricos en el caso de que las distancias negativas del denominador sean muy pequeñas.

Esta función se propone en base a lo observado durante el entrenamiento con la otra función de error. En la función de Wohlhart-Lepetit, el error es nulo si la pareja positiva ya está suficientemente cerca y la negativa suficientemente lejos. Gracias al aprendizaje, la red triplet consigue acercar y alejar las imágenes lo suficiente en menos de una época de entrenamiento. Esto implica que el error es prácticamente nulo con muchos de los tripletes al poco tiempo de comenzar el entrenamiento y el aprendizaje se ralentiza. En la función propuesta, en cambio, el error puede hacerse muy pequeño pero no anularse. Por tanto, es posible que permita que el aprendizaje sea más estable durante más épocas.

4.3.3. Bases de datos triplets de lugares

Estas redes utilizan un triplete de imágenes de entrada. Se dispone de los mismos lugares que los utilizados para las bases siamesas por lo que las combinaciones de lugares positivos siguen siendo las mismas. Basta con agrupar las parejas obtenidas de otra forma.

En las bases siamesas, las parejas se mezclan intercalando una pareja positiva y una negativa. Para las bases de las triplets, en cambio, se introduce en una base la imagen neutra, en otra la pareja y en otra la imagen del lugar contrario. Se puede ver un ejemplo del emparejamiento de las tres bases en la Figura 4.9. Se realiza el proceso con todas las imágenes del conjunto y se obtienen 834746 tripletes con los que entrenar la red.

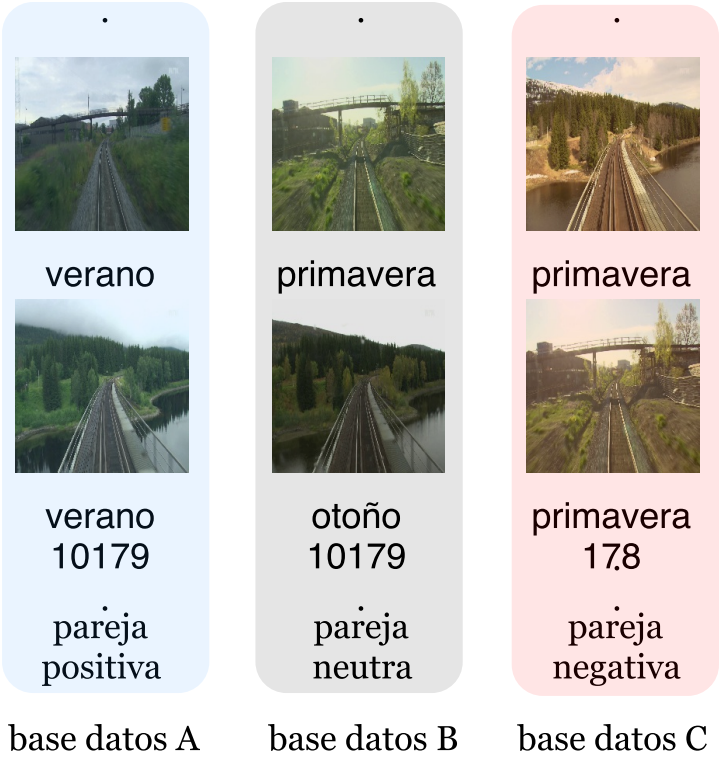


Figura 4.9: Bases de datos creadas para las redes triplets. **Izquierda:** Base de datos A. Se introduce el ejemplo positivo. **Centro:** Base de datos B. Se introduce el ejemplo neutro. **Derecha:** Base de datos C. Se introduce el ejemplo negativo.

5. Resultados

En este capítulo se presentan los resultados obtenidos. En la sección 5.1 se explican las métricas utilizadas. En las secciones 5.2, 5.3 y 5.4 se muestran los resultados obtenidos utilizando redes pre-entrenadas, las estructuras siamesas y las estructuras triplets respectivamente. En la sección 5.5 se comparan los métodos y se dan los resultados finales.

5.1. Evaluación del reconocedor y métricas utilizadas

Para comprobar el funcionamiento del reconocedor se necesita una base de datos de referencia y una base de datos de entrada o búsqueda. Por cada imagen de entrada se deberá realizar el procesamiento y la búsqueda del lugar más parecido en la base de referencia. Se puede ver el proceso de comprobación del funcionamiento en la Figura 5.1.

Para la comprobación se dispone del conjunto de test que se menciona en la sección 3.3 del capítulo 3. Se cuenta con imágenes de 3450 lugares distintos, con cada lugar formado por las imágenes extraídas de cada estación durante 5 segundos distintos.

Para evaluar el reconocedor se utilizan las imágenes de los lugares en una estación del año como referencia y en otra estación como entrada. Se anotan los lugares más ceranos para cada imagen de entrada y se obtienen las siguientes métricas:

Fracción de lugares correctos (fc): Es el número de veces que el lugar predicho por el sistema coincide con el lugar de entrada, esto es:

$$\text{fc} = \frac{\text{Nº lugares acertados}}{\text{Nº lugares evaluados}}, \tag{5.1}$$

Se cuenta como lugar acertado si el más cercano en la base de referencia coincide con el de entrada. Como se explica en la sección 3.4 del capítulo 3, en este trabajo se considera una ventana de cinco. Por ello se cuenta como acierto si el lugar predicho está entre dos imágenes anteriores y dos posteriores con respecto a la imagen tomada en la misma ubicación exacta.

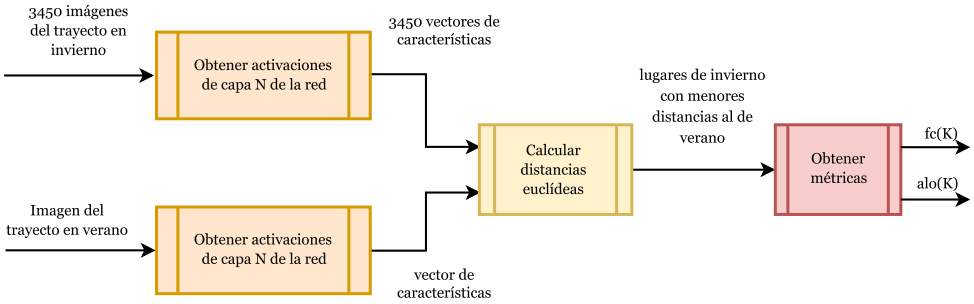


Figura 5.1: Ejemplo del proceso de evaluación del reconocedor. La referencia son las imágenes de los lugares en invierno. La entrada de búsqueda son las imágenes de los mismos lugares en verano.

Fracción de lugares correctos en K (fc(K)): En lugar de considerar únicamente el lugar más cercano, se cuenta el número de lugares correctos entre los K más cercanos. Esta métrica permite medir la constancia del reconocedor, es decir, la capacidad de que las imágenes más cercanas pertenezcan al mismo lugar. De todas las imágenes dentro de la ventana que forman un lugar, generalmente la imagen más cercana es la que fue tomada en la misma posición exacta. La constancia del reconocedor da una idea de la robustez a la hora de acercar el resto de imágenes que forman el lugar. Cuando $K = 1$, esta métrica es la misma que la fracción de correctos (fc).

$$\text{fc(K)} = \frac{(\text{Nº lugares correctos entre los K más cercanos / K})}{\text{Nº lugares evaluados}}, \tag{5.2}$$

Fracción de lugares correctos con uno al menos entre K (alo(K)): Esta métrica considera un acierto solo con que haya un lugar correcto entre los K más cercanos. De esta manera, se tiene en cuenta que el reconocedor puede fallar con el más cercano, pero acertar con varias de las K predicciones sucesivas. Posteriormente se podría filtrar el resultado para mejorar el rendimiento. Se tiene en cuenta la misma ventana que en las métricas anteriores.

$$\text{alo(K)} = \frac{\text{Nº veces al menos uno correcto entre los K más cercanos}}{\text{Nº lugares evaluados}}, \tag{5.3}$$

5.2. Redes pre-entrenadas

En la Figura 5.2 se muestra el comportamiento de las cuatro capas de la red VGG-16 elegidas. Se utilizan el verano y el invierno como referencia y el resto de estaciones en cada caso como entrada.

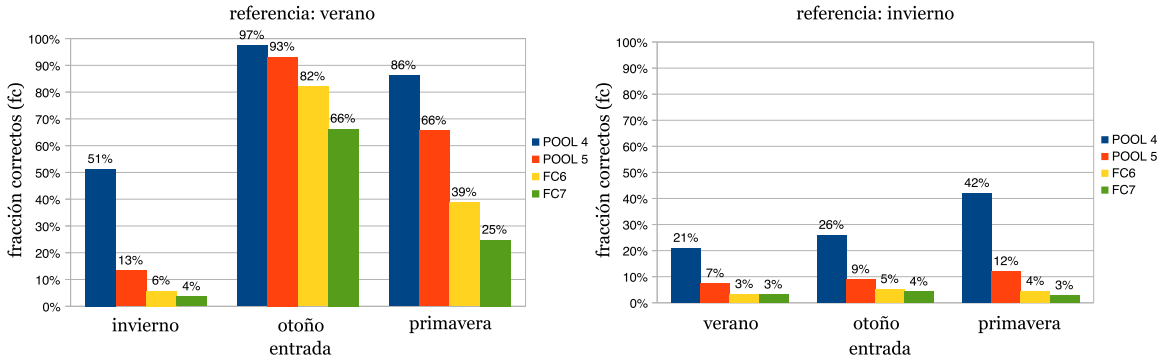


Figura 5.2: Comparativa de los resultados obtenidos utilizando las capas de la red VGG-16 entrenada con Imagenet. **Izquierda:** Fracción de correctos (fc) usando el verano como referencia y el resto de estaciones como entrada. **Derecha:** Fracción de correctos (fc) usando el invierno como referencia y el resto de estaciones como entrada.

La cuarta capa reductora (*pool*4) tiene la mejor fracción de correctos en todas las combinaciones de estaciones estudiadas. El peor resultado se obtiene con la séptima capa totalmente conectada en todos los casos. Se puede observar que el resultado empeora progresivamente con las capas utilizadas, esto tiene dos motivos. Por un lado, la dimensión de las capas disminuye lo que implica descriptores más pequeños. Al reducir la dimensión se pierde parte de la información que contienen. Por otro lado, las capas se acercan a la salida de la red. La información de las capas finales es más específica del problema original de la red y se adapta menos a este problema.

Se concluye que el punto de partida para mejorar el reconocedor es utilizar la cuarta capa reductora de la red VGG-16, con un tamaño de descriptor de 100352.

También se observa que el sistema reconocedor funciona peor cuando se usa como referencia o entrada el invierno. Esto es porque las condiciones ambientales en invierno se diferencian de las del resto del año. Mientras que las de primavera, verano y otoño se parecen más entre ellas. Elementos como la nieve y la niebla ocultan parte de los detalles de los paisajes que permiten diferenciar unos de otros fácilmente. Los descriptores extraídos no son suficientemente robustos y fallan en estas condiciones.

5.2.1. Red VGG-16 alternativa: conjunto de datos Places

La red VGG-16 fue entrenada originalmente para la clasificación de objetos en el conjunto de datos Imagenet. En [22] entrenaron la misma estructura para la clasificación de escenas con el conjunto de datos Places. Clasificar escenas es detectar si una imagen es una playa, una montaña o cualquier otro paisaje. Se decide estudiar si las características que extrae esta red son mejores como descriptores del reconocedor de lugares.

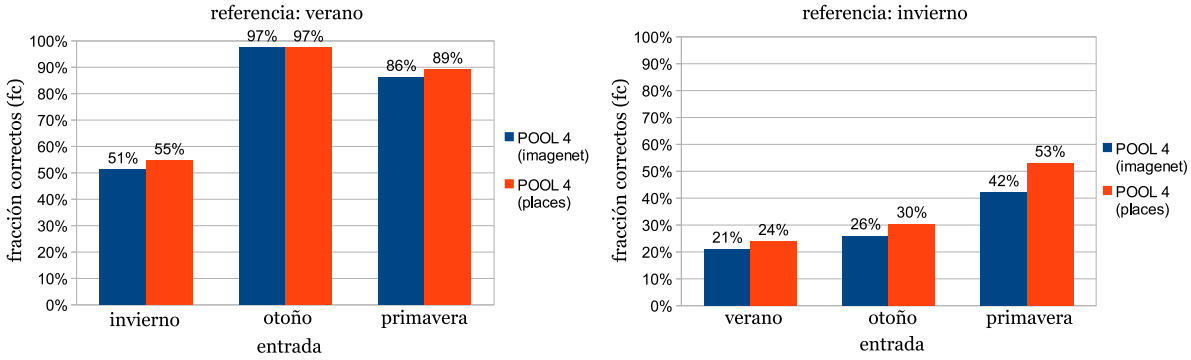


Figura 5.3: Comparativa de los resultados obtenidos utilizando los descriptores de la red entrenada con Imagenet y con Places. **Izquierda:** Fracción de correctos (fc) usando el verano como referencia y el resto de estaciones como entrada. **Derecha:** Fracción de correctos (fc) usando el invierno como referencia y el resto de estaciones como entrada.

En la Figura 5.3 se muestra el comportamiento de la capa *pool*4 de la red VGG-16 entrenada en el conjunto Places frente a la entrenada en el conjunto Imagenet. Se utilizan el verano y el invierno como referencia y el resto de estaciones en cada caso como entrada.

Se puede comprobar que la única combinación en la que empatan las redes es utilizando el verano como referencia y el otoño como entrada. Obteniendo un 97 % de lugares acertados en los dos casos. La red entrenada con el conjunto Places es mejor en el resto de casos. Se concluye que la red VGG-16 entrenada con el conjunto Places extrae descriptores más robustos a los cambios de apariencia que se dan en el conjunto de datos utilizado. Esto se debe a que para clasificar paisajes, las capas internas de la red han aprendido a detectar características del entorno que son más útiles en este problema. Se decide utilizarla como punto de partida para el resto de pruebas.

5.3. Redes siamesas

Antes de elegir el diseño final de la red siamesa se eligen varios de sus hiperparámetros mediante validación. Se comprueba el funcionamiento de los siguientes tamaños del descriptor de salida: 128, 256, 512 y 1024. En la tabla 5.1 se muestra la fracción de correctos obtenido en el conjunto de validación para cada dimensión. El mejor resultado se obtiene para 128 dimensiones y conforme se aumenta el tamaño, empeora el resultado. Esto se debe a que aumenta el número de parámetros y comienzan a aparecer efectos adversos como el sobreajuste (se necesitan más datos para entrenar) o el incremento del tiempo necesario para el entrenamiento.

También se aprecia que al aumentar las dimensiones, la inicialización de los pesos

Tabla 5.1: Fracción de correctos en conjunto de validación (Referencia: verano. Entrada: invierno)

dimensión	fracción correctos (fc)
128	0.5881
256	0.535
512	0.4138
1024	0.275

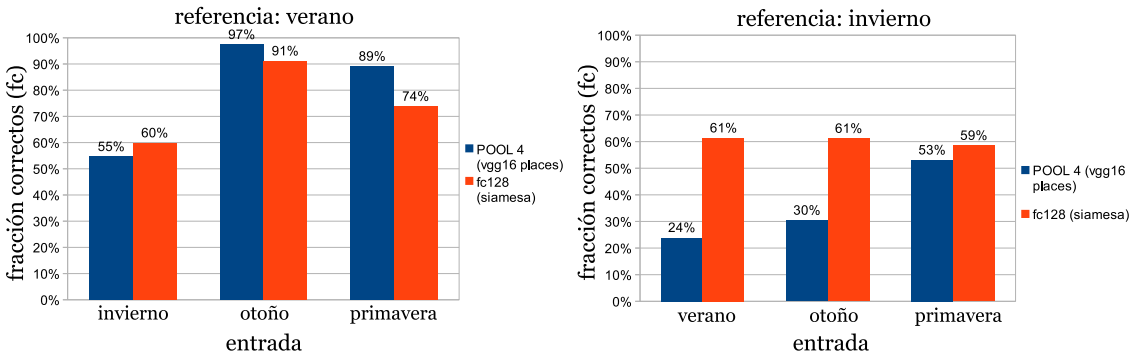


Figura 5.4: Comparativa de los resultados obtenidos con la red siamesa. Se muestran los resultados de la capa pool4 de la red pre-entrenada y de la red siamesa con una capa totalmente conectada de dimensión 128. **Izquierda:** Fracción de correctos (fc) usando el verano como referencia y el resto de estaciones como entrada. **Derecha:** Fracción de correctos (fc) usando el invierno como referencia y el resto de estaciones como entrada.

de las neuronas de la red puede desequilibrar el entrenamiento. Esto se debe a que al aumentar las dimensiones, una mala inicialización aleatoria de los pesos puede hacer que las distancias euclídeas entre vectores sean muy grandes. Distancias grandes provocan errores grandes que pueden desestabilizar el progreso del entrenamiento. Se decide por tanto utilizar 128 dimensiones para la capa totalmente conectada. Se entrena la red durante 5 épocas, con un ratio de aprendizaje de 1^{-6}. Se utiliza la función de error contrastiva con un margen de 15.

En la Figura 5.4 se muestra el comportamiento de las cuatro capas de la red VGG-16 elegidas. Se utilizan el verano y el invierno como referencia y el resto de estaciones en cada caso como entrada. Se puede apreciar que la siamesa no alcanza el resultado de la capa *pool*4 cuando se utiliza el verano como referencia, teniendo otoño o primavera como entrada (dos últimas combinaciones del gráfico de la izquierda). En el resto de combinaciones la siamesa consigue sobrepasar a la red pre-entrenada. Los resultados más destacables se aprecian en el caso de utilizar invierno como referencia en la Figura 5.4. Utilizando el verano como referencia, la red siamesa llega a obtener casi tres veces más lugares correctos que la red pre-entrenada.

Tabla 5.2: Fracción de correctos en conjunto de validación con la red triplet (Referencia: verano. Entrada: invierno)

dimensión	fracción correctos (fc)
128	0.8019
256	0.8061
512	0.8065
1024	0.8116

La red siamesa consigue mejorar en los casos en los que aparece el invierno. Esto es porque el entrenamiento ha conseguido que los descriptores extraídos por la red capturen información que no empeora a pesar de las condiciones ambientales del invierno. El entrenamiento, sin embargo, no ha conseguido que los descriptores de 128 dimensiones contengan información tan compleja y discriminativa como los extraídos por la capa *pool*4 de la red pre-entrenada, empeorando el resultado en el resto de casos.

5.4. Redes triplets

Al igual que con las redes siamesas, se validan varios de los hiperparámetros antes de elegir el diseño final de la red. En este caso, se cuenta con dos funciones de error a comprobar por lo que se prueban hiperparámetros diferentes con cada una. Para la red triplet con la función de error de Wohlhart-Lepetit se prueban varios tamaños de la capa totalmente conectada. Se muestran los resultados en la tabla 5.2 donde se aprecia que, a diferencia de la red siamesa, la red triplet mejora conforme se aumenta el tamaño de los descriptores. La mejora, sin embargo, es menor de un 1 % por lo que se decide utilizar el mismo tamaño que las siamesas, 128 dimensiones. De esta forma se pueden comparar los resultados en condiciones similares.

Se entrena la red durante cinco épocas con un ratio de aprendizaje de 1^{-7}. El margen de la función de error se fija a 1[1]. El resto de hiperparámetros utilizados con la otra función de error se utilizan de nuevo ya que siguen resultando óptimos. Se entrena la red también durante 5 épocas.

En la Figura 5.5 se muestran los resultados obtenidos con las dos funciones de error usando las redes triplets. La función de error propuesta solo consigue mejorar a la red pre-entrenada cuando se utiliza el invierno como referencia. Es probable que haya que modificar alguno de los términos de la ecuación para considerar las distancias de manera distinta.

[1]En las tres funciones de error utilizadas aparece un parámetro de margen distinto. Cada uno tiene una función diferente por lo que no son comparables.

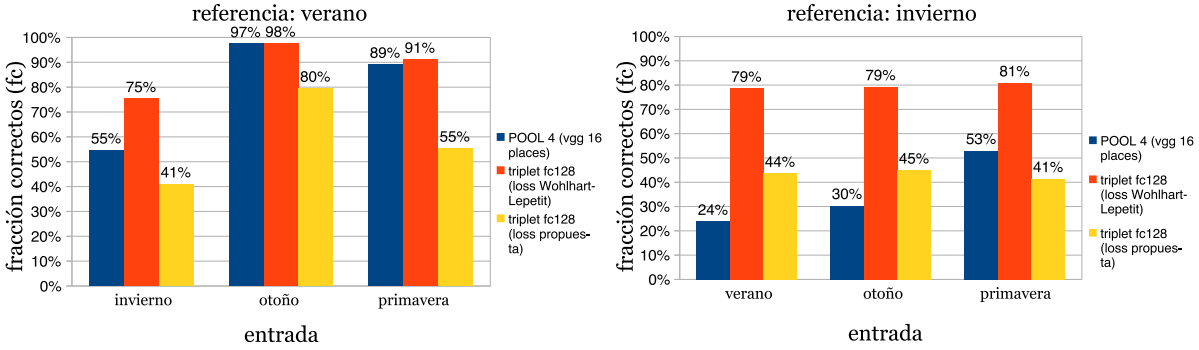


Figura 5.5: Comparativa de los resultados obtenidos con las dos funciones de error triplets. Se muestran también los resultados de la capa pool4 de la red pre-entrenada como referencia. **Izquierda:** Fracción de correctos (fc) usando el verano como referencia y el resto de estaciones como entrada. **Derecha:** Fracción de correctos (fc) usando el invierno como referencia y el resto de estaciones como entrada.

Con la función de error de Wohlhart-Lepetit se consigue mejorar por primera vez todos los resultados de la red pre-entrenada. Por lo que se utilizará esta función en el siguiente paso. Repetir el entrenamiento de la capa totalmente conectada con 128 dimensiones, realizando un ajuste fino del resto de capas de la red salvo el primer bloque convolucional. Se entrena la red durante cinco épocas con un ratio de aprendizaje de 1^{-3} para la capa totalmente conectada. El resto de capas tienen un ratio de aprendizaje de 1^{-4}. Los resultados se muestran en la Figura 5.6. Se aprecia que la red con ajuste fino consigue mejorar los resultados obtenidos con la red triplet sin ajuste fino en todos los casos.

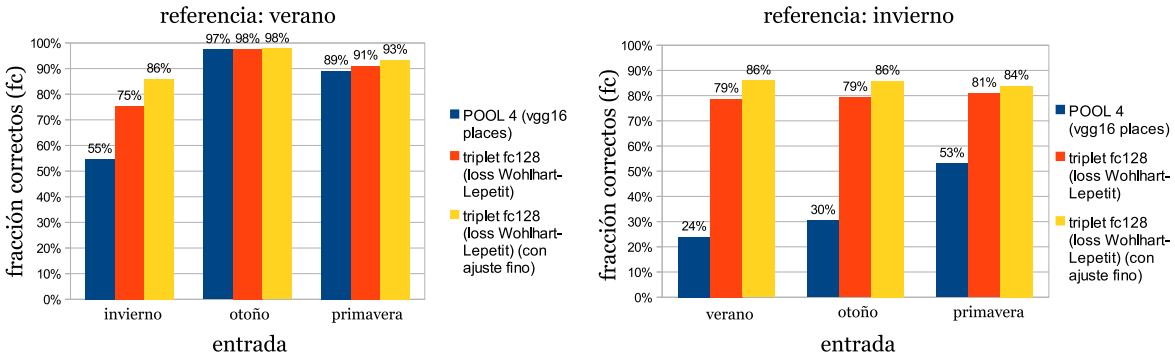


Figura 5.6: Comparativa de los resultados obtenidos con y sin ajuste fino de la red triplet. Se muestran también los resultados de la capa pool4 de la red pre-entrenada como referencia. **Izquierda:** Fracción de correctos (fc) usando el verano como referencia y el resto de estaciones como entrada. **Derecha:** Fracción de correctos (fc) usando el invierno como referencia y el resto de estaciones como entrada.

5.5. Comparativa

En la Figura 5.7 se muestran los resultados obtenidos con las estrategias utilizadas en este trabajo. Los mejores resultados se obtienen con la red triplet entrenada con la función de error de Wohlhart-Lepetit, con ajuste fino, que gana en todas las combinaciones.

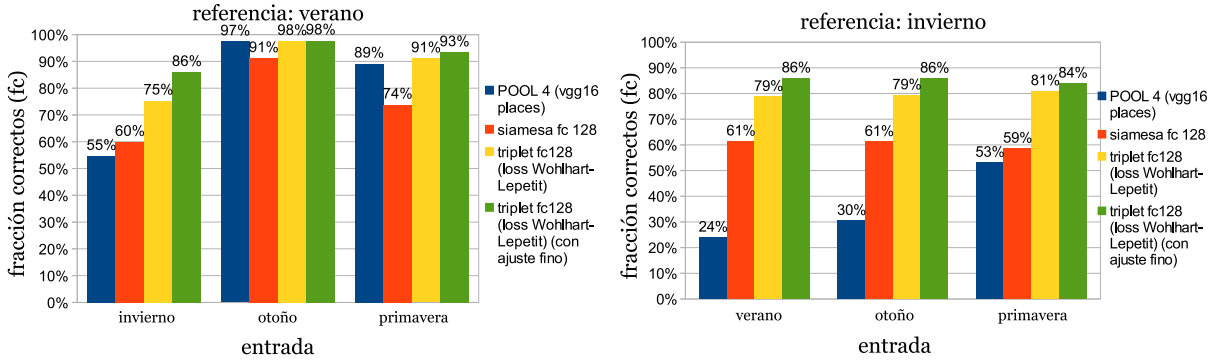


Figura 5.7: Comparativa de los resultados obtenidos con las tres estrategias seguidas: redes pre-entrenadas, redes siamesas y redes triplets (sin y con ajuste fino). **Izquierda:** Fracción de correctos (fc) usando el verano como referencia. Invierno y otoño como entrada. **Derecha:** Fracción de correctos (fc) usando el invierno como referencia. Verano y otoño como entrada.

Utilizando redes siamesas se ha conseguido mejorar drásticamente el resultado de la red pre-entrenada cuando los lugares de referencia presentan condiciones muy diferentes. El descriptor extraído tiene 128 dimensiones, frente a las 100352 del de la cuarta capa reductora. Con lo cual, hasta en los casos en los que no se mejora el resultado, se está obteniendo un descriptor mucho más pequeño y manejable.

Las redes triplets mejoran en las combinaciones más favorables y en las más desfavorables al mismo tiempo. Esto implica que la red consigue aprender a reconocer características más discriminativas y robustas a los cambios de apariencia que los de la red pre-entrenada. Las redes siamesas, por tanto, no tienen la complejidad requerida para resolver este problema. Además, entrenar todas las capas de la red ha conseguido mejorar el resultado frente a entrenar solo la última capa. Este resultado, sin embargo, puede ser a costa de hacer que la red se sobreajuste a datos como los del conjunto Nordland y empeore para reconocer lugares con otras características.

Se concluye que la mejor estrategia resulta utilizar una estructura triplet para el ajuste fino de la red VGG-16 Places. Añadiendo una capa totalmente conectada con 128 dimensiones y entrenando mediante la función de error de Wohlhart-Lepetit. En la tabla 5.3 se muestra la fracción de correctos obtenida para cada combinación posible de estaciones. Las de entrada se muestran en la columna izquierda y las de referencia en la

Tabla 5.3: Fracción de correctos de todas las estaciones contra el resto de estaciones.

entrada \referencia	verano	otoño	invierno	primavera
verano	—	0.8548	0.8591	0.9545
otoño	0.9777	—	0.8583	0.9562
invierno	0.8597	0.9771	—	0.9545
primavera	0.9336	0.94	0.8388	—

fila superior.

En la Figura 5.8 se muestra un ejemplo de las tres imágenes del conjunto de invierno más cercanas a una de entrada en verano. Todos los lugares obtenidos son correctos a pesar de haber desplazamientos y cambios de apariencia notables entre las imágenes. Dada la ventana de cinco imágenes, se considera el mismo lugar si el número de la imagen está entre dos anteriores o dos posteriores al de la pareja.

En la Figura 5.9 se muestra un ejemplo de emparejamientos correctos entre verano e invierno. Se aprecia que el reconocedor funciona a pesar de que la nieve cambia la apariencia y ocluye parte del paisaje. Incluso cuando hay cambios drásticos de iluminación (pareja 3227-3228), cuando la sombra del tren aparece en la imagen (pareja 1315-1315) o cuando el limpiaparabrisas aparece en la fotografía (pareja 858-857).

En la Figura 5.10 se muestra un ejemplo de emparejamientos incorrectos entre verano e invierno. Se aprecia que parte de los fallos se deben a paisajes que apenas cambian cuando el tren se desplaza en línea recta durante un tramo. Como los elementos del fondo del paisaje son los mismos, el reconocedor los confunde, pero en realidad los lugares están bastante cerca entre ellos (pareja 3076-3069). El resto de fallos, se deben a estructuras del paisaje que no tienen los mismos elementos, pero tienen una cierta similitud geométrica. La pareja 1267-1609, por ejemplo, no es el mismo lugar pero la forma de la montañas crea un espacio de cielo visible casi idéntico.

5.6. Estado del arte

Este trabajo se compara con tres técnicas que se consideran estado del arte en la materia: el análisis de componentes principales [14], redes neuronales entrenadas para clasificación de lugares con reducción multiescala *"multiscale pooling"* [8] y redes triplets para reconocimiento de lugares invariante a la apariencia [7]. Todos ellos utilizan en el desarrollo o la evaluación de sus métodos el conjunto Nordland de alguna forma, lo cual permite comparar los resultados.

Con los resultados publicados de la técnica PCA y las redes neuronales Amosnet e Hybridnet (entrenadas para reconocer lugares y publicadas por Chen *et al.* [8]), se pueden

comparar los resultados directamente. Los resultados de Gómez-Ojeda *et al.* [7] solo pueden ser comparados en parte, por lo que se comentan posteriormente. En la Figura 5.11 se presenta la fracción de correctos obtenidos. La red entrenada en este trabajo gana en todas las combinaciones posibles menos en verano-primavera, combinación en la que pierde frente a la técnica PCA. El resultado es especialmente apreciable en las combinaciones con invierno.

Al disponer de las redes Amosnet e Hybridnet, se pueden extraer y comparar el resto de métricas que demuestran el funcionamiento del reconocedor. Solo se muestran los de la red Hybridnet porque obtiene mejores resultados que la red Amosnet. En la Figura 5.12 se muestra la fracción de correctos en 5 de la red entrenada en este trabajo frente a la red Hybridnet y la capa *pool*4 de la red pre-entrenada. Lo primero que se puede observar es que la fracción de correctos en 5 es inferior a la fracción de correctos simple de la Figura 5.11. Esto tiene sentido, al aumentar las imágenes más cercanas que se consideran, aparecen imágenes que pertenecen a otros lugares pero que tienen una apariencia similar. La red triplet entrenada en este trabajo empeora menos que el resto. Esto implica que las redes triplets dan lugar a un reconocedor más constante a la hora de acercar imágenes que pertenecen a los mismos lugares.

Se repite el mismo procedimiento extrayendo la métrica de la fracción de correctos con uno al menos entre 5. Se presentan los resultados en la Figura 5.13. Esta métrica aumenta con respecto a la fracción de correctos simple. Esto tiene sentido puesto que aunque haya algún lugar incorrecto que se parezca mucho al de entrada, tiende a haber algún lugar correcto entre el resto de los más cercanos. La red triplet entrenada en este trabajo vuelve a ganar o empatar en todos los casos.

Con la técnica de Gómez-Ojeda *et al.* [7] es más difícil realizar la comparación. Publican una gráfica en la que se aprecia la fracción de correctos entre 5 para la combinación de verano como referencia e invierno como entrada, obteniendo un 35 %. Con la red triplet entrenada en este trabajo se obtiene un 61 % en esa misma combinación (Figura 5.12 izquierda). Si bien ellos evalúan un mayor número de imágenes, su resultado es peor en principio. Ellos entrenan una red triplet partiendo de una red pre-entrenada diferente a la de este trabajo. Es posible que la red VGG-16 entrenada con el conjunto Places funcione mejor como punto de partida para reconocer lugares.

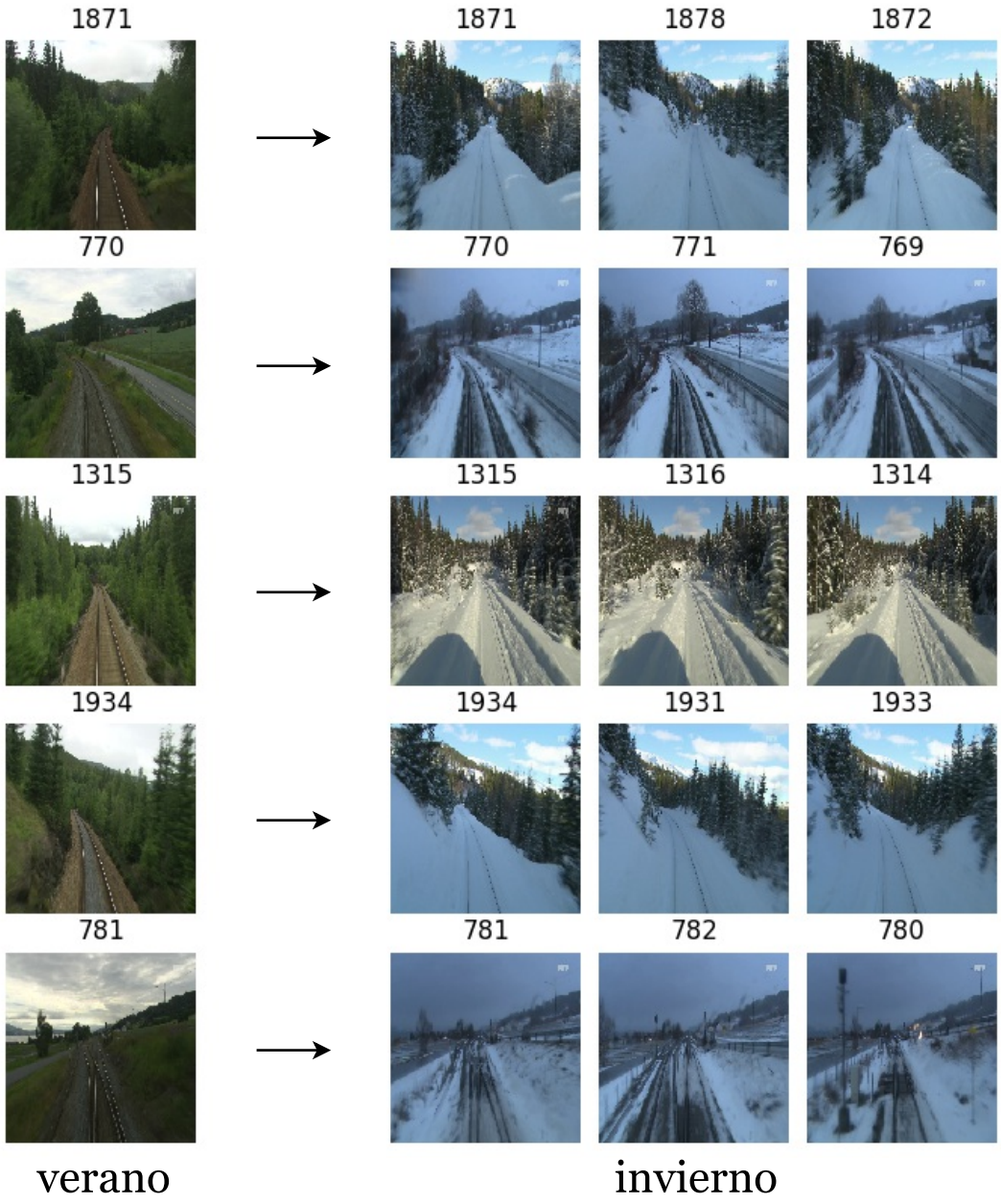


Figura 5.8: Ejemplo de imágenes más cercanas a una de entrada. La imagen de entrada es del conjunto de verano y la base de referencia es el conjunto de invierno. Se indica el número de la imagen en la parte superior de cada una.

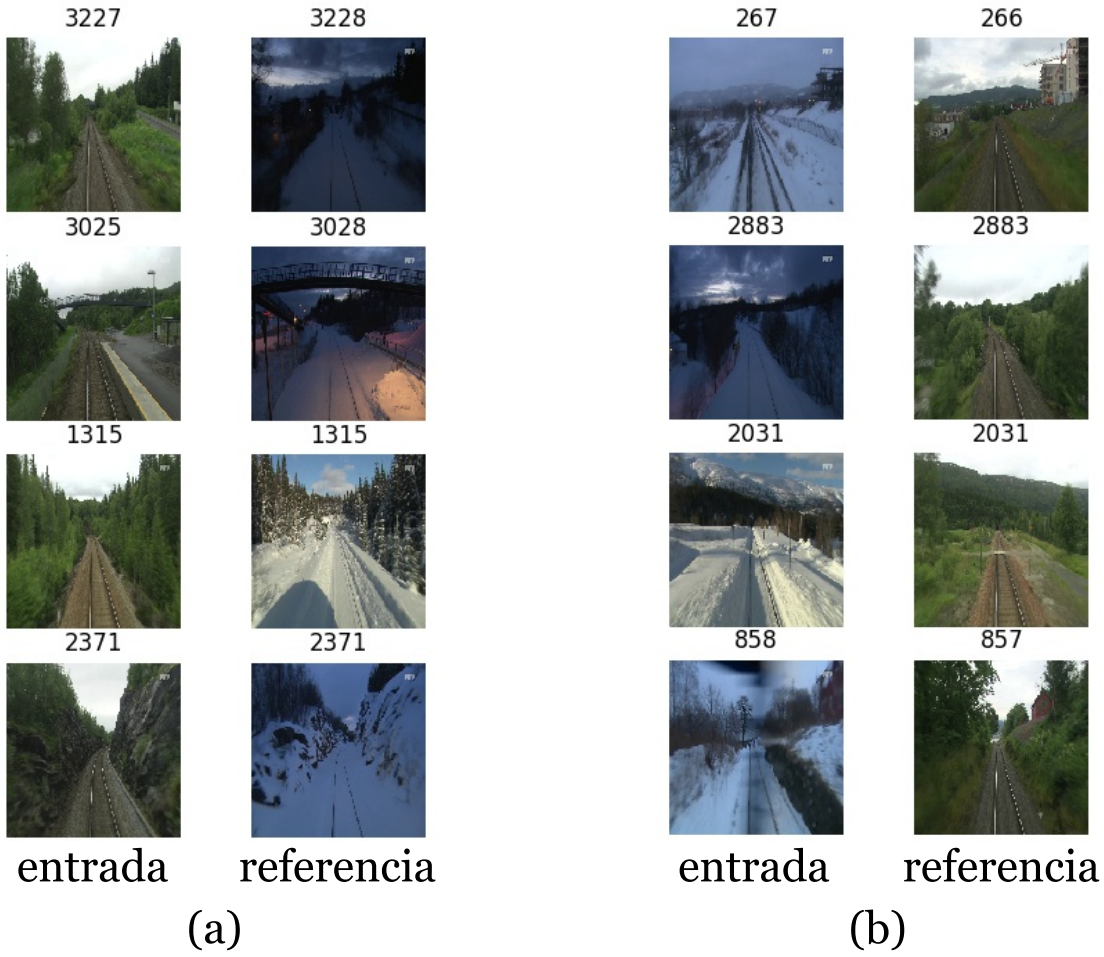


Figura 5.9: Ejemplo de lugares reconocidos correctamente. **a:** Lugares de entrada tomados en verano, buscados en el conjunto de invierno. **b:** Lugares de entrada tomados en invierno, buscados en el conjunto de verano. Se indica el número de la imagen en la parte superior de cada una.

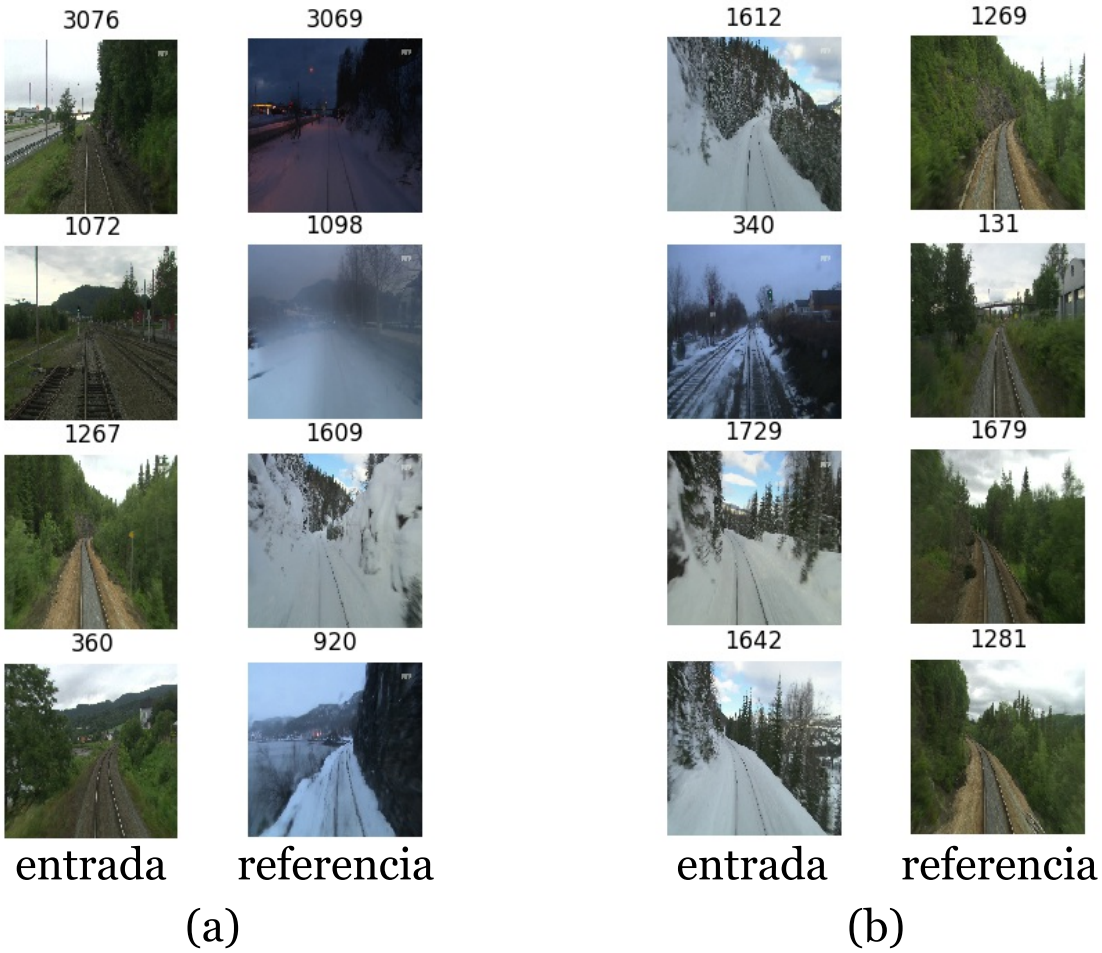


Figura 5.10: Ejemplo de lugares reconocidos incorrectamente. **a:** Lugares de entrada tomados en verano, buscados en el conjunto de invierno. **b:** Lugares de entrada tomados en invierno, buscados en el conjunto de verano. Se indica el número de la imagen en la parte superior de cada una.

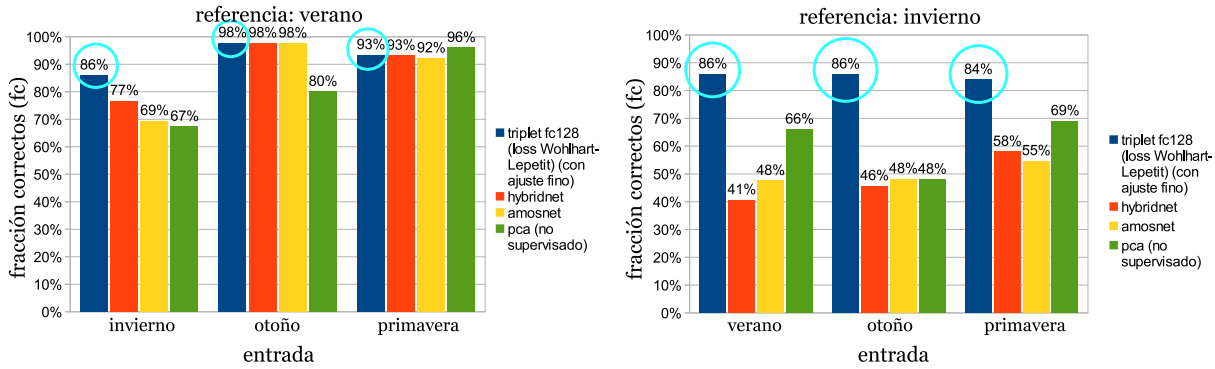


Figura 5.11: Comparativa de la fracción de correctos del reconocedor implementado frente a la red Hybridnet, Amosnet y la técnica basada en PCA. Además, se ha marcado con un círculo azul los resultados de este trabajo para diferenciarlos del resto. **Izquierda:** Fracción de correctos (fc) usando el verano como referencia. **Derecha:** Fracción de correctos (fc) usando el invierno como referencia.

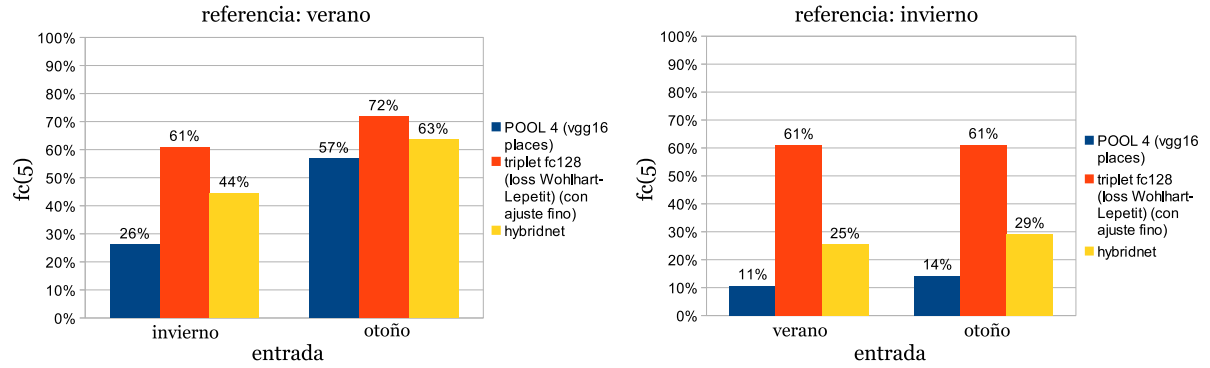


Figura 5.12: Comparativa de la fracción de correctos en 5 del reconocedor implementado frente a la red Hybridnet. Se incluye el resultado de la capa pool4 como referencia. No se muestran las combinaciones con primavera ya que se ha visto en el resto de gráficas que el comportamiento es intermedio. **Izquierda:** Fracción de correctos en 5 usando el verano como referencia. Invierno y otoño como entrada. **Derecha:** Fracción de correctos en 5 usando el invierno como referencia. Verano y otoño como entrada.

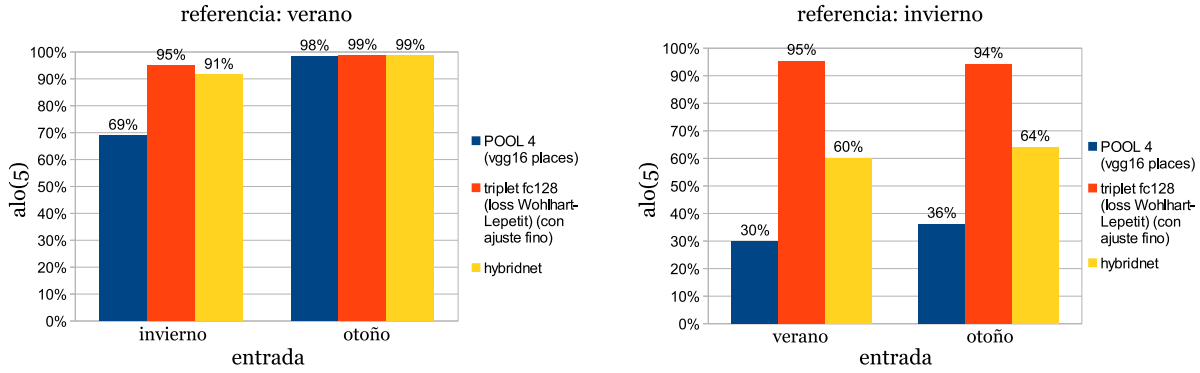


Figura 5.13: Comparativa de la fracción de correctos con uno al menos en 5 del reconocedor implementado frente a la red Hybridnet. Se incluye el resultado de la capa pool4 como referencia. **Izquierda:** Fracción de correctos con uno al menos entre 5 usando el verano como referencia. Invierno y otoño como entrada. **Derecha:** Fracción de correctos con uno al menos entre 5 usando el invierno como referencia. Verano y otoño como entrada.

6. Conclusiones

En este trabajo se planteaba el desarrollo de un método de reconocimiento de lugares robusto a cambios de apariencia el entorno. Especialmente aquellos que generan las condiciones climatológicas y estacionales. La aproximación propuesta funciona empleando una red neuronal para extraer descriptores de las imágenes que se comparan mediante la distancia euclídea. La comparación permite medir el parecido entre lugares y tratar de reconocer el lugar deseado.

Para implementar el reconocedor se ha utilizado un único conjunto de datos, desarrollado exclusivamente a partir de vídeos del trayecto de tren Nordland. Esto difiere con respecto a otras aproximaciones del estado del arte. En la estructuración del conjunto, se ha mostrado la complejidad que implica manejar un conjunto de datos. Especialmente en el minucioso análisis de la información de partida, como prestar atención a detalles como paradas o túneles en un vídeo de varias horas. El conjunto ha sido una parte vital del proceso de entrenamiento y evaluación de las redes neuronales empleadas y del propio reconocedor, lo cual justifica el tiempo invertido.

Los experimentos realizados muestran que las redes neuronales siamesas y triplets son capaces de aprender a extraer descriptores robustos a los cambios de apariencia que han visto durante su entrenamiento. Dando un mejor resultado las redes triplets, por su mayor complejidad. Con respecto a las aproximaciones que utilizan redes neuronales, se ha demostrado que la red VGG-16 pre-entrenada para clasificar escenas supone un buen punto de partida para realizar el ajuste fino de la red al problema concreto de reconocimiento de lugares. Finalmente, se ha demostrado la mejora que supone el reconocedor propuesto frente a otras técnicas que forman parte del estado del arte en el reconocimiento de lugares con el conjunto Nordland. Se ha obtenido un reconocedor que es capaz de obtener más de un 98 % de lugares acertados en tramos de más de 80 kilómetros de recorrido cuando las condiciones son similares y hasta un 86 % cuando los lugares presentan cambios de apariencia drásticos como los que se dan entre verano e invierno.

7. Herramientas y cronograma

7.1. Herramientas

Para trabajar con las redes se han utilizado las librerías de Caffe [23] por su versatilidad a la hora de diseñar y probar distintas estructuras de redes neuronales. Para el resto de procedimientos como la comprobación del funcionamiento o la elaboración de gráficas se ha utilizado Python, especialmente las librerías Numpy [24], SciPy [25] y MatPlotLib [26].

7.2. Cronograma

La organización temporal del desarrollo del trabajo se expone en el cronograma de la Figura 7.1.

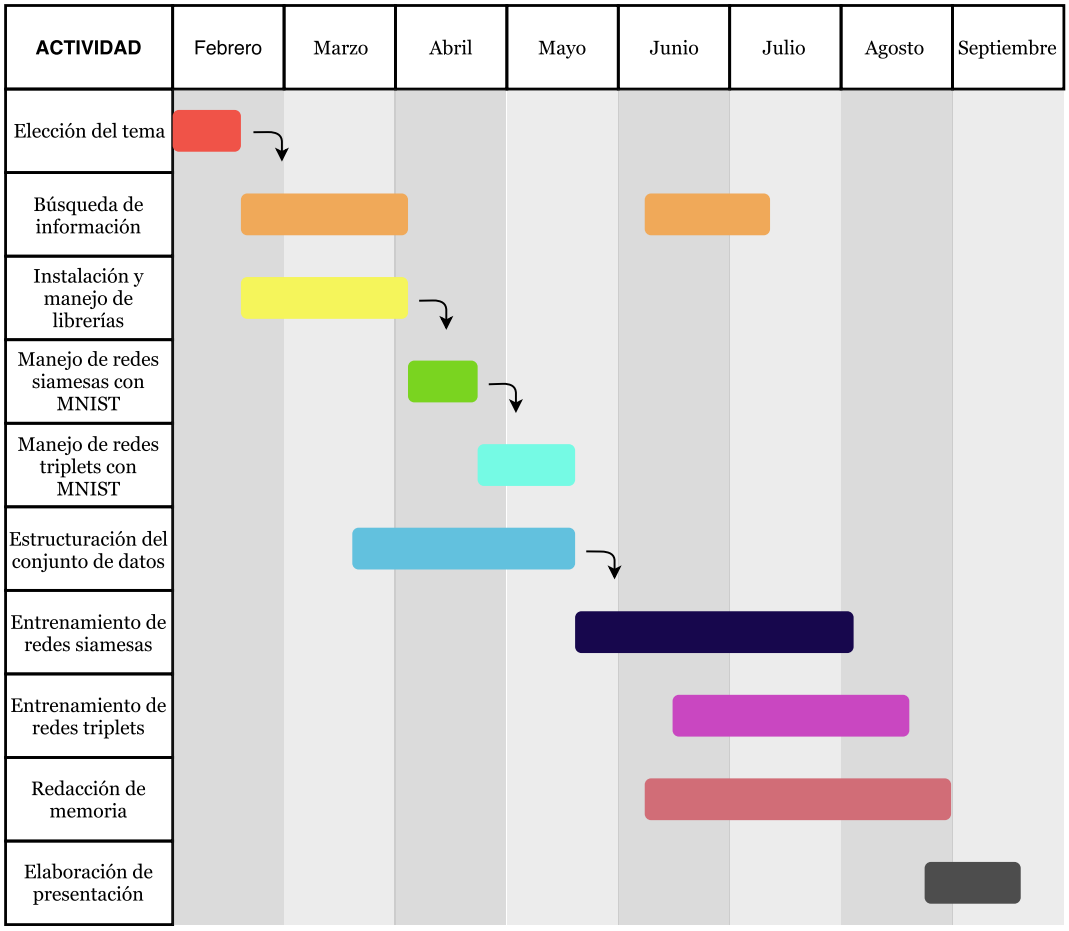


Figura 7.1: Cronograma con el desarrollo del trabajo.

Bibliografía

[1] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[2] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[3] W. Maddern and S. Vidas, "Towards robust night and day place recognition using visible and thermal imaging," *RSS 2012: Beyond laser and vision: Alternative sensing techniques for robotic perception*, 2012.

[4] M. Bosse and R. Zlot, "Keypoint design and evaluation for place recognition in 2d lidar maps," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1211–1224, 2009.

[5] C. Cadena, D. Gálvez-López, F. Ramos, J. D. Tardós, and J. Neira, "Robust place recognition with stereo cameras," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 5182–5189, IEEE, 2010.

[6] N. Sünderhauf, F. Dayoub, S. Shirazi, B. Upcroft, and M. Milford, "On the performance of convnet features for place recognition," *CoRR*, vol. abs/1501.04158, 2015.

[7] R. Gomez-Ojeda, M. Lopez-Antequera, N. Petkov, and J. Gonzalez-Jimenez, "Training a convolutional neural network for appearance-invariant place recognition," *arXiv preprint arXiv:1505.07428*, 2015.

[8] Z. Chen, A. Jacobson, N. Sunderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, and M. Milford, "Deep learning features at scale for visual place recognition," *arXiv preprint arXiv:1701.05105*, 2017.

[9] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[10] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3d object classification," in *Advances in Neural Information Processing Systems*, pp. 656–664, 2012.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[13] N. B. Corporation, "Nordlandsbanen: minute by minute, season by season," 2012. [Online; accessed 19-June-2017].

[14] S. Lowry and M. J. Milford, "Supervised and unsupervised linear learning techniques for visual place recognition in changing environments," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 600–613, 2016.

[15] O. contributors, "Open street map open data (un servicio de datos de acceso libre), con licencia open data commons open database license(odbl) de la fundación openstreetmap (osmf)," 2017. [Online; accessed 08-Aug-2017].

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[17] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a"siamese"time delay neural network," in *Advances in Neural Information Processing Systems*, pp. 737–744, 1994.

[18] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2, pp. 1735–1742, IEEE, 2006.

[19] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," 1998.

[20] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1386–1393, 2014.

[21] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3109–3118, 2015.

[22] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, pp. 487–495, 2014.

[23] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[24] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[25] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed 22-Aug-2017].

[26] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.