Universidad Zaragoza
1542

Trabajo Fin de Máster

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

Condition invariant place recognition using recurrent neural networks

Autor

Daniel Olid Hernández

Directores

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

Escuela de Ingeniería y Arquitectura, Universidad de Zaragoza.

2018

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 73020655w 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) Máster, (Título del Trabajo) Reconocimiento de lugares con invarianza a cambios del entorno mediante redes neuronales recurrentes.

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

Zaragoza, 19 de Septiembre de 2018

Fdo: Daniel Olid Hernández

Agradecimientos

Quiero comenzar este trabajo dándoles las gracias a todas las personas que han sido importantes para mí durante este año. En primer lugar, quiero dar las gracias a mis directores, el Dr. Javier Civera y D. José María Fácil. Gracias por haber vuelto a confiar en mí, por ayudarme continuamente y por tener siempre tiempo para revisar los extensos textos novelísticos que llevaba a las tutorías. También quiero darles las gracias por abrirme nuevas oportunidades académicas que me han ayudado a afrontar el trabajo todavía más motivado.

Gracias a mi madre, por aguantar mis angustias existenciales durante todo el año. Gracias también a mi padre y al resto de mi familia, que siempre se interesan por mí y siempre me han apoyado.

Finalmente, quiero darles las gracias a mis amigos. Gracias por hacerme descubrir que sobrevivir al lanzamiento de melocotones en un pueblo de Teruel también sirve para despejar la mente y terminar adecuadamente el trabajo.

Resumen

El reconocimiento de lugares es un problema que tiene gran relevancia gracias a sus aplicaciones en navegación autónoma. Actualmente sigue sin estar resuelto de forma robusta. El reto que se afronta en este trabajo es el cambio en la apariencia visual de los lugares por factores como los fenómenos meteorológicos o la iluminación. La mayoría de aproximaciones se basan en una única vista y estudian diversos algoritmos para procesar las imágenes. Se ha demostrado que los métodos basados en redes neuronales son más robustos ante cambios de apariencia. En este trabajo se emplean redes neuronales y además se propone el uso de múltiples vistas. La hipótesis de partida es que la información que contiene una secuencia de imágenes puede ayudar a mejorar la robustez del reconocimiento de lugares ante cambios de aspecto.

Para poder utilizar la información secuencial en este trabajo se proponen varias estrategias, entre ellas el estudio de las redes neuronales recurrentes. Cada estrategia requiere una fase de análisis, ajuste de parámetros y estudio de las arquitecturas neuronales óptimas. Las estrategias analizadas mejoran con respecto a los resultados obtenidos tanto por aproximaciones del estado del arte que no utilizan la información temporal como por aproximaciones que la utilizan. El reconocedor desarrollado es capaz de identificar correctamente más de un 92% de lugares en tramos de más de 80 kilómetros de recorrido cuando las condiciones son adversas.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Este trabajo se enmarca en el campo de la visión por computador, concretamente en el reconocimiento de lugares a partir de información visual. Este problema consiste específicamente en, ante una imagen de un lugar determinado, encontrar en una base de datos con imágenes aquellas que corresponden al mismo lugar. El reconocimiento de lugares es un problema que actualmente tiene gran relevancia gracias a sus aplicaciones en navegación autónoma. El reconocimiento de lugares ha sido utilizado en mapeo topológico de escenas [1], cierre de bucle en aplicaciones SLAM (Mapeo y localización simultáneos) [2] y aprendizaje de la dinámica de los lugares para la localización permanente [3].

El problema presenta múltiples retos. Por ejemplo, las bases de datos de imágenes suelen ser muy grandes y el tiempo de procesamiento está limitado a la capacidad del robot. El reto que se va a afrontar en este trabajo es el cambio en la apariencia visual de los lugares. En la Figura 1.1 se puede observar cómo la nieve y la iluminación cambian drásticamente la apariencia de un lugar. Estos son solo algunos ejemplos, también es importante considerar fenómenos meteorológicos como la lluvia, la niebla o los objetos dinámicos como los coches. En la Figura 1.2 se puede ver el esquema de un método de reconocimiento de lugares con imágenes de un trayecto de tren. Para reconocer correctamente el lugar, el algoritmo debe ser robusto a los cambios producidos por la nieve y las nubes.

(a) (b)

Figura 1.1: Ejemplos de cambios de apariencia de los lugares. **a:** Dos imágenes del mismo lugar en invierno y verano. **b:** Dos imágenes del mismo lugar de día y de noche.

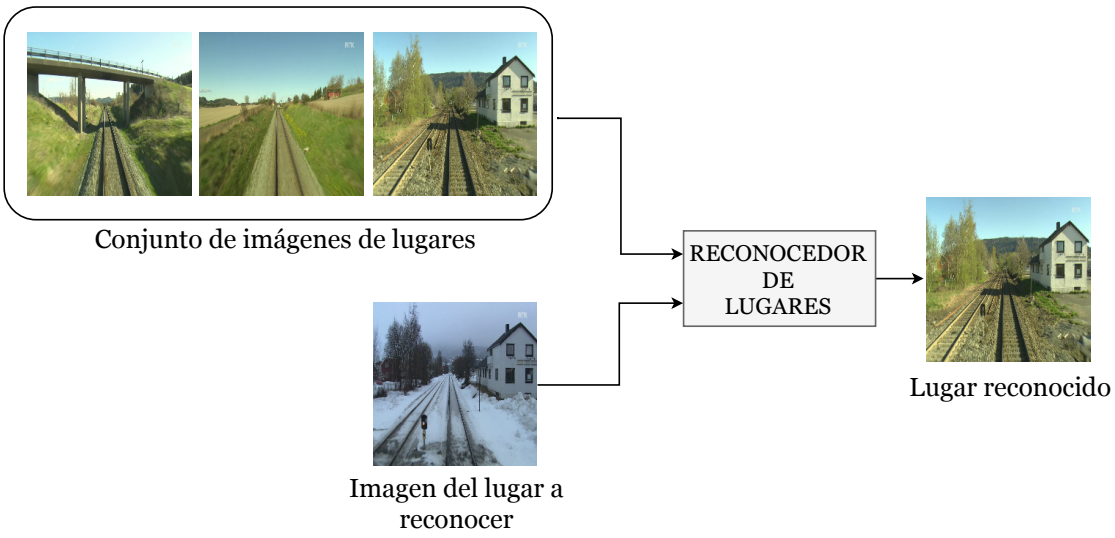

Figura 1.2: Esquema de un método de reconocimiento de lugares. Se utilizan dos entradas. Un conjunto de imágenes de lugares por un lado y la imagen del lugar a reconocer por otro. La salida es el lugar reconocido.

La manera clásica de afrontar el reconocimiento de lugares consiste en aplicar algoritmos basados en características salientes de las imágenes (p.ej. SIFT, SURF Y ORB). Estos algoritmos extraen características locales de las imágenes, es decir, información relevante que permite comparar unos lugares con otros. La aproximación que representa el estado del arte en este tipo de técnicas es FAB-MAP [4]. Estos métodos presentan invarianza a ligeros cambios del punto de vista y de iluminación, pero no son robustos a cambios drásticos de apariencia.

Las aproximaciones más novedosas proponen utilizar redes neuronales convolucionales para extraer información de las imágenes. En [5], Niko Sünderhauf *et al.* demostraron que las redes neuronales superan a otras técnicas, especialmente ante cambios drásticos de apariencia. Desde entonces, han surgido múltiples estudios que analizan el uso de redes neuronales en el reconocimiento de lugares como [6, 7, 8]. Destaca especialmente el trabajo de Gómez-Ojeda *et al.* [9], ya que fueron los primeros en entrenar redes neuronales específicamente para este problema.

Existen otro tipo de aproximaciones que estudian la posibilidad de utilizar la información temporal de las imágenes para mejorar el desempeño de los algoritmos. Es decir, en lugar de comparar la información extraída de una imagen, utilizar la extraída de una serie de imágenes consecutivas. En la Figura 1.3 se muestran varias imágenes tomadas en el Paseo de la Independencia de Zaragoza. Resulta complicado averiguar el lugar en el que ha sido tomada la primera fotografía por la izquierda de la figura. La segunda imagen y especialmente la tercera contienen detalles como la forma típica de las farolas. En la cuarta imagen también se pueden apreciar las vías del tranvía. Aunque la primera imagen no contiene información suficiente, considerar el resto de imágenes permite identificar el lugar. Esto demuestra que la información secuencial puede ayudar a mejorar la robustez del reconocimiento de lugares.

Figura 1.3: Secuencia de imágenes de un lugar. De izquierda a derecha se muestra una secuencia de imágenes tomadas consecutivamente en el Paseo de la Independencia de Zaragoza.

DBoW [10] y especialmente SeqSLAM [11], son dos propuestas que aprovechan la información secuencial para obtener resultados del estado del arte en la materia. A pesar de ello, en el momento de realización de este trabajo los autores no tienen constancia de que se hayan utilizado redes neuronales capaces de utilizar la información temporal de las secuencias de imágenes.

Este trabajo de fin de Máster (TFM) parte del desarrollo previo realizado en el trabajo de fin de Grado (TFG) [12] realizado por del mismo autor y los mismos directores. En el TFG se utilizaron redes neuronales con estructuras siamesas y triplets para obtener un reconocedor de lugares capaz de alcanzar resultados del estado del arte. En el Capítulo 3 se resumirá brevemente el trabajo previo realizado para comprender mejor el punto de partida. Posteriormente a la entrega del TFG, se realizó un artículo resumiendo el trabajo [13] que fue aceptado en el décimo workshop en planificación, percepción y navegación para vehículos inteligentes en la conferencia internacional de robots inteligentes (IROS) del 1 al 5 de Octubre de 2018 en Madrid [14].

El objetivo general de este trabajo es estudiar el reconocimiento de lugares con redes neuronales profundas e incorporando información secuencial. Se han propuesto y evaluado varias alternativas que han demostrado experimentalmente tasas de aciertos superiores al estado del arte. Más en detalle, las tareas concretas son las siguientes:

- Desarrollar un reconocedor de lugares invariante a cambios de apariencia partiendo de los resultados obtenidos en el TFG. El reconocedor utiliza imágenes que procesa mediante redes neuronales para extraer descriptores. Comparando los descriptores se determina el parecido entre lugares.

- Estudiar el uso de diversas estrategias basadas en redes neuronales capaces de aprovechar la información temporal. Entre ellas, entrenar redes neuronales recurrentes de tipo LSTM ("*Long Short Term Memory*") capaces de procesar secuencias de imágenes.

- Mejorar el conjunto de datos ("*dataset*") que se desarrolló en el TFG partiendo de vídeos del trayecto de tren Nordland.
- Evaluar y comparar el resultado con el trabajo previo y con otras aproximaciones del estado del arte en el reconocimiento de lugares.

Los resultados alcanzados en este TFM superan a los alcanzados en el trabajo previo, demostrando la conveniencia de utilizar la información secuencial en el reconocimiento visual de lugares.

1.1. Estructura del trabajo

En este capítulo se introduce el reconocimiento de lugares y se desarrollan los objetivos del trabajo. En el Capítulo 2 se explican los principios básicos de las redes neuronales utilizadas. En el Capítulo 3 se presentan los métodos de reconocimiento de lugares basados en una única vista. En el Capítulo 4 se presentan los métodos de reconocimiento de lugares propuestos basados en múltiples vistas. En el Capítulo 5 se muestran los resultados obtenidos. Finalmente, en el Capítulo 6 se plantean las conclusiones finales del trabajo.

1.2. Herramientas

En este trabajo se ha utilizado el lenguaje de programación Python junto con las librerías de redes neuronales de Keras [15] y TensorFlow [16]. Se han elegido estas librerías porque permiten trabajar de forma versátil con distintos modelos de redes recurrentes. Para los resultados del TFG se utilizaron las librerías de redes neuronales de Caffe [17], más rígidas y de uso menos común. También ha sido útil la librería Numpy [18] para implementar los algoritmos. Las figuras de la memoria se han diseñado con la herramienta Draw.io [19].

2. Introducción a las redes neuronales

En este capítulo se desarrollan los principios básicos de las redes neuronales necesarios para comprender el resto del trabajo.

Las redes neuronales son algoritmos capaces de implementar funciones diversas aprendiendo a partir de ejemplos y pertenecen al campo del aprendizaje automático. Desde los primeros modelos de neuronas artificiales en 1943 [20], estos algoritmos han evolucionado drásticamente. A día de hoy, se utilizan en áreas como los vehículos autónomos, el reconocimiento de caracteres, el reconocimiento de voz e incluso en la medicina [21].

2.1. Neurona artificial

Las neuronas artificiales son el elemento básico de procesamiento de las redes neuronales. En la Figura 2.1 se muestra el procesamiento interno que realiza una neurona artificial típica. La función de las neuronas es procesar un vector de entrada $\boldsymbol{x}$ (que tiene una dimensión determinada) mediante una serie de operaciones internas para obtener la salida, y. Las operaciones internas que realiza la neurona, es decir, la matemática completa que implementan, es la siguiente:

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

Donde $\boldsymbol{x}$ es el vector de entradas, y es la salida, $\boldsymbol{w}$ es el vector de pesos internos, b es el sesgo y f, la función de activación. De todas las variables, solo el vector de pesos y el sesgo se aprenden mediante el entrenamiento. Para obtener la salida, la neurona realiza el producto escalar de los pesos y la entrada, añadiendo después un sesgo y aplicando una función de activación (generalmente no lineal).

Aunque internamente implementan una serie de operaciones básicas, el poder de las redes neuronales reside en la agrupación de varias neuronas. La agrupación se basa en formar capas, donde cada capa está compuesta de una serie de neuronas que procesan la misma entrada y cada neurona obtiene una salida distinta. La salida de las neuronas de cada capa forma la entrada de la capa siguiente y así sucesivamente hasta la salida final de la red. Al agrupar varias capas, se incrementa la capacidad de representar funciones cada vez más complejas.

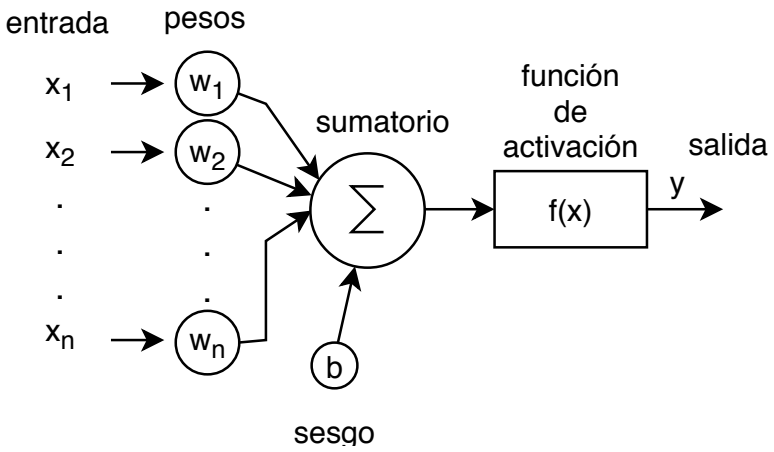

Figura 2.1: Bloques internos de una neurona artificial típica. $(x_1,x_2,...,x_n)^\intercal$ es el vector de entrada de tamaño n. y es la salida. $(w_1,w_2,...,w_n)^\intercal$ es el conjunto de pesos internos. b es el sesgo. Se representa también el sumatorio interno y la función de activación no lineal.

En la Sección 2.2 se muestran las distintas topologías de conexionado de las capas.

Uno de los aspectos más importantes de las redes neuronales es el aprendizaje. Para que una neurona pueda aprender, se debe seguir un proceso de entrenamiento. Para ello se procesan los ejemplos de entrada y se modifican los pesos de las neuronas para que la salida se acerque a la deseada. Se necesita un conjunto de datos o "*dataset*", con ejemplos representativos del problema a resolver y una función de coste o "*loss*" a optimizar, que mide la diferencia entre la salida de la red y la salida deseada.

Además de los pesos, existen otro tipo de parámetros que también son fundamentales en las redes neuronales, los hiperparámetros. Son las variables que definen varios aspectos de las redes, como su arquitectura. Un ejemplo es el número de capas de la red o el número de neuronas en cada capa. Para ajustar estos hiperparámetros, se suelen seguir técnicas como entrenar la red con varios valores distintos de un hiperparámetro determinado y evaluar el funcionamiento en parte de los datos disponibles.

2.2. Arquitecturas

En la Figura 2.2 se muestra un ejemplo de red neuronal multicapa básica. Esta red tiene como entrada un vector de dimensión tres y está formada por una capa oculta con tres neuronas y una capa de salida con dos neuronas. Toda capa que no es entrada ni salida se denomina capa oculta. Se puede ver que todas las neuronas están conectadas a todas las salidas de la capa anterior y no existe ningún tipo de realimentación de la salida, 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

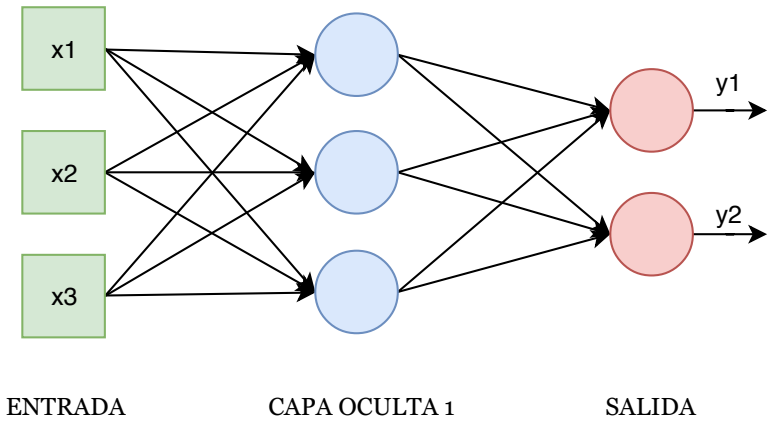

Figura 2.2: Red neuronal con una capa oculta. La entrada es de dimensión tres. Las tres neuronas de la capa oculta se representan en azul. Las dos neuronas de al salida se representan en rojo. La salida tiene dimensión dos.

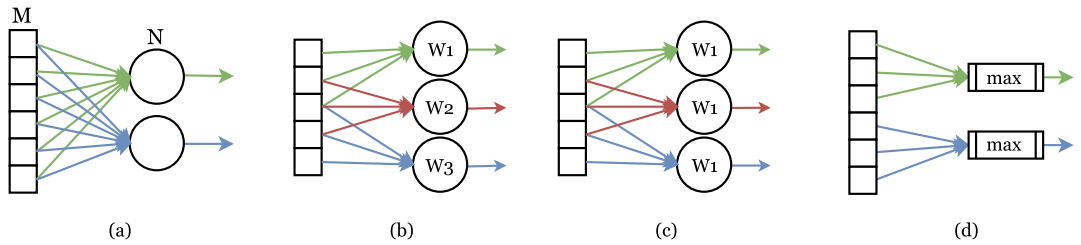

Figura 2.3: Olid, D. (2017). Varios tipos de capas de una red neuronal. [Figura]. Recuperada del TFG [12]. Las conexiones de cada neurona se marcan en distinto color para facilitar la visualización. **a**: Totalmente conectada. Donde M es la capa anterior y N la capa actual. **b**: Localmente conectada. Cada neurona se conecta a 3 entradas de la capa anterior. Cada neurona tiene sus propios parámetros: W_1,W_2 y W_3. **c**: Convolucional. Las neuronas comparten los mismos parámetros: W_1. **d**: Reductora máxima.

ver en la Figura 2.3 y son las siguientes:

Capa totalmente conectada (“*fully-connected*”): Este tipo de capas (Figura 2.3a) se caracteriza porque cada neurona se conecta a todas las neuronas de la capa anterior. Al no imponer ninguna restricción sobre las conexiones, estas capas pueden aprovechar las relaciones internas que presentan los datos. Su mayor desventaja es que tienen un elevado coste computacional y de memoria. Las neuronas artificiales tienen tantos pesos como entradas. Ante una entrada de tamaño M y una capa con N neuronas, el número de parámetros de la capa totalmente conectada será $(M+1)\cdot N$. Cuando se trabaja con un número elevado de dimensiones, el número de neuronas de la capa estará limitado por la memoria y la capacidad de cómputo disponible.

Capa localmente conectada: En las capas localmente conectadas (Figura 2.3b) las neuronas de cada capa no se conectan con todas las de la capa anterior. Estas capas son menos costosas computacionalmente, pues tienen menos parámetros. Se suelen utilizar cuando los datos de entrada tienen algún tipo de relación local. Un ejemplo son las imágenes, donde los píxeles que representan un objeto, por ejemplo, se encuentran próximos entre ellos. De esta forma, aunque las capas totalmente conectadas también pueden explotar estas relaciones, las localmente conectadas pueden hacerlo de forma similar y con mayor eficiencia.

Capa convolucional: Dentro de las capas localmente conectadas, existe un tipo de capa que se denomina capa convolucional. En estas capas, las neuronas tampoco se conectan a todas las de la capa anterior. El patrón de conexionado y el hecho de que las neuronas de la capa tienen los mismos pesos, hacen que las capas procesen la entrada aplicando una operación similar a la convolución. Una de las ventajas de estas capas es que utilizan todavía menos parámetros que las localmente conectadas. La ventaja fundamental es que estas capas presentan invarianza espacial. Como las neuronas de la capa comparten los pesos, aunque las conexiones sean locales, estas capas pueden detectar el mismo patrón en cualquier parte del vector de entrada. Aunque las capas explicadas previamente también pueden lograr esto, las capas convolucionales lo hacen de manera más eficiente.

En las Figuras 2.3a, b y c se puede apreciar gráficamente la diferencia entre las capas totalmente conectadas, localmente conectadas y las convolucionales. La totalmente conectada es la única en la que las neuronas se conectan a todas las de la capa anterior. En la localmente conectada y la convolucional, cada neurona se conecta solo a tres elementos de la capa anterior. Se observa que la localmente conectada tiene pesos distintos para cada neurona (W_1,W_2 y W_3) mientras que en la convolucional las neuronas comparten los pesos (W_1)

Capa reductora (*pooling*) y Dropout: La Figura 2.3d muestra una capa reductora o "*pooling*". A diferencia de las anteriores, estas capas no contienen realmente neuronas ni pesos internos que se entrenen. La función de estas capas es procesar la entrada aplicando algún tipo de operación. Existen distintos tipos de capa reductora pero todas reducen la dimensión de la entrada. En la figura se representa la reductora máxima. La operación aplicada en esa capa consiste en seleccionar partes de la entrada y dejar pasar solo los valores máximos de cada parte. Su ventaja fundamental es reducir el número de parámetros de la red, reduciendo los costes computacionales. Además, dotan a la red de cierta invarianza a traslación.

La función de las capas Dropout solo se suele aplicar durante el entrenamiento de la red y son eliminadas en la fase de ejecución o implementación[1]. Esta capa fue propuesta en [22] como un mecanismo para mejorar el aprendizaje de la red. La operación que aplica se basa en eliminar conexiones entre la capa previa y la capa posterior de manera aleatoria durante

[1]Se considera que las redes neuronales tienen varios modos de funcionamiento. Los más importantes son el modo de entrenamiento y el de ejecución. En el entrenamiento la red aprende los parámetros. En el de ejecución los parámetros no se modifican, se utiliza la red para procesar la entrada y obtener la salida.

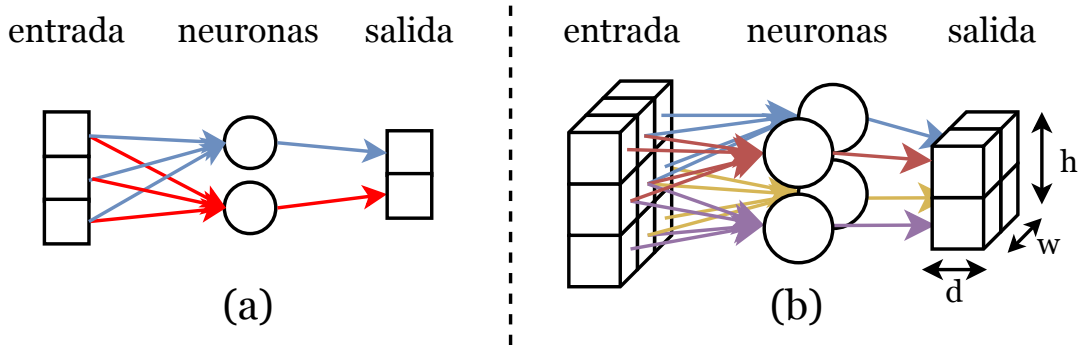

Figura 2.4: Representación gráfica de una red neuronal típica y una red convolucional aplicada a imágenes. Se muestran las conexiones de las neuronas en colores distintos. **a**: Red neuronal típica con una capa, un vector de entrada y uno de salida. **b**: Red neuronal Convolucional con una capa, una matriz de entrada y una de salida. Se indican las dimensiones que aparecen al trabajar con matrices (d: profundidad, w: anchura, h: altura).

el entrenamiento. Esto favorece que las neuronas se adapten para cooperar mejor en cualquier circunstancia y mejora los resultados del entrenamiento.

2.3. Redes convolucionales

Las redes convolucionales desarrolladas por Yann LeCun en 1998 [23] son un tipo de redes neuronales que se utilizan mayoritariamente con imágenes. Las redes convolucionales pueden estar formadas por varios tipos de capas como las comentadas en la sección previa. La diferencia principal de estas redes es que incluyen alguna capa convolucional al principio de la red y que trabajan en dos y tres dimensiones.

Para entender esto mejor, en la Figura 2.4 se compara la estructura de una red neuronal clásica y una red convolucional. Se observa que las neuronas en una red convolucional se agrupan tridimensionalmente, conectándose a lo largo y ancho con la matriz de entrada (las imágenes se pueden entender como matrices de números). También se aprecia que al trabajar con matrices, la salida de las capas también son matrices. Al utilizar capas convolucionales aplicadas en dos dimensiones, estas redes consiguen aprovechar las relaciones espaciales que aparecen en las imágenes con mayor eficiencia [12], ya que utilizan menos parámetros. A continuación, se va a explicar el proceso de aplicación de una capa convolucional sobre una imagen 2D.

Capa convolucional aplicada a imágenes: Las capas convolucionales están formadas por filtros que procesan las matrices de entrada para producir las matrices de salida o mapas de características. Los filtros son matrices cuyos valores se aprenden mediante el entrenamiento. Las dimensiones del filtro (anchura y altura) y la cantidad de filtros de cada capa son

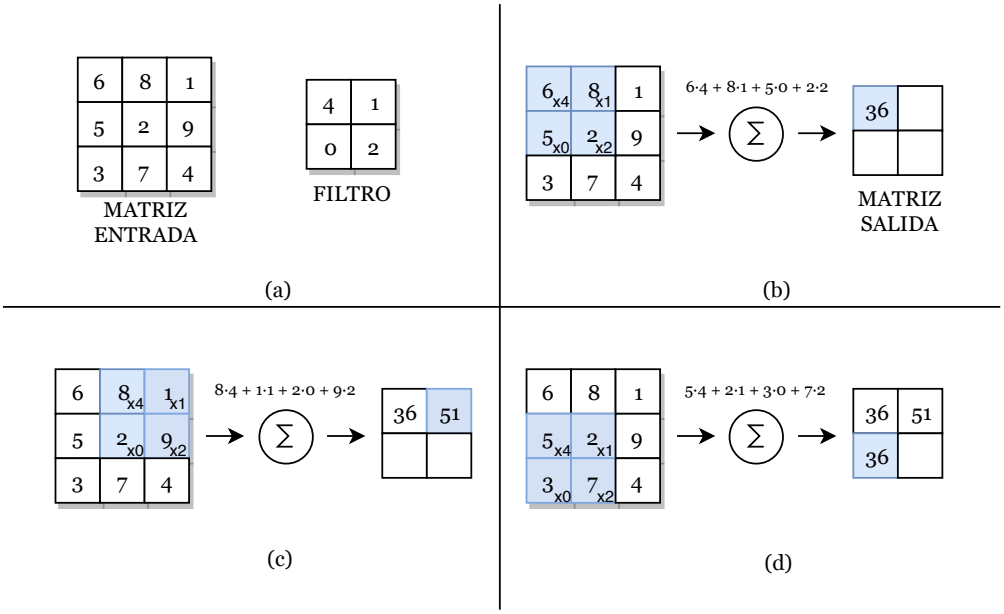

Figura 2.5: Convolución en dos dimensiones. **a:** Se muestra la matriz de entrada de tamaño 3×3 y el filtro de tamaño 2×2. **b:** Se desplaza el filtro a la primera posición sobre la matriz de entrada. Se multiplican y suman los elementos coincidentes para obtener el valor de salida. Se muestra la matriz de salida de tamaño 2×2. Para obtener el resto de valores se siguen los mismos pasos desplazando el filtro. **c:** Se repite el proceso en la segunda posición. **c:** Se repite el proceso en la penúltima posición. Volviendo a desplazar el filtro se obtendría la matriz de salida al completo.

hiperparámetros que se establecen en el diseño de la red. El procesamiento realizado por estas capas consiste en convolucionar los filtros con la entrada y aplicar la función de activación.

En la Figura 2.5 se muestra la aplicación de la convolución 2D sobre una matriz de entrada. Si se tratara de una imagen en color, habría que tener en cuenta también la profundidad. Por motivos de claridad se va a explicar únicamente en dos dimensiones, aunque la convolución en tres dimensiones seguiría un procedimiento muy similar. La Figura 2.5a muestra la matriz de entrada (dimensión 3×3) y el filtro a aplicar (dimensión 2×2). En las siguientes figuras se muestran los tres primeros pasos de la operación, en los cuales se desplaza el filtro por la matriz de entrada realizando la convolución. La Figura 2.5b muestra el primer paso de la secuencia. Se multiplican los elementos que coinciden entre el filtro y la matriz de entrada y se suman. El proceso se repite desplazando el filtro para obtener la matriz de salida.

La salida de una capa convolucional aplicada sobre una imagen real se puede ver en la Figura 2.6. Se puede observar que las capas convolucionales actúan como detectores de características. En el caso de la Figura 2.6, la capa ha aprendido a detectar bordes en la escena. Las primeras capas de una red convolucional se suelen especializar en detectar patrones básicos como rectas o formas. Las capas posteriores pueden aprender a detectar ojos, ruedas o cualquier otro elemento más complejo que resulte útil para resolver el problema.

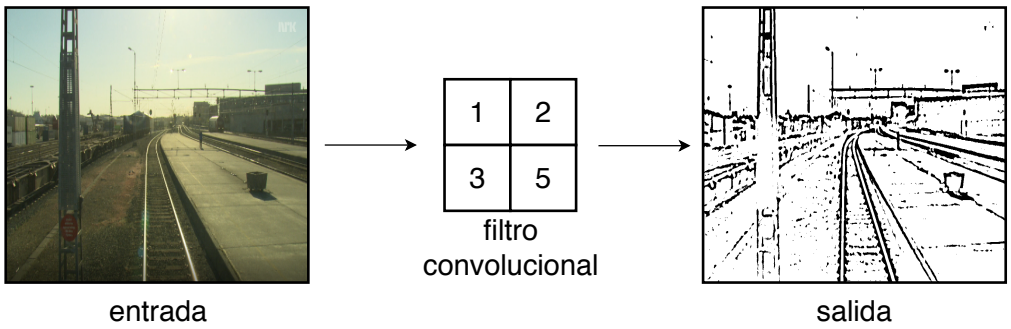

Figura 2.6: Efecto simulado de aplicar una capa convolucional. Se muestra la simulación de la salida de una capa convolucional cualquiera. El filtro tiene la capacidad de detectar características relevantes en la imagen como líneas o incluso caras.

2.4. Redes recurrentes

Además de clasificar las redes neuronales en función de sus topologías y estructuras, es posible clasificarlas atendiendo al flujo de información [21]. En las redes comentadas previamente, se ha asumido que la información va desde la entrada hacia la salida. Ese tipo de redes se denominan redes prealimentadas o "*feed-forward*". Cuando parte de la información de la red vuelve a la entrada para ser procesada se trata de una red recurrente o "*feedback*".

Las redes recurrentes son un tipo de redes neuronales utilizadas para procesar secuencias y datos temporales. Las neuronas recurrentes tienen memoria y su salida depende de la entrada actual y del estado previo de la neurona. Un ejemplo de la aplicación de estas redes es en el texto predictivo, donde la palabra siguiente depende tanto de la última palabra escrita, como de las palabras anteriores. En la Figura 2.7a se muestra la representación esquemática típica de estas redes. Se puede apreciar que la salida en un instante t, depende tanto de la entrada en ese instante ($\mathbf{x}_t$) como de la información realimentada.

La Figura 2.7b muestra la representación desplegada. Esta visualización permite entender mejor el funcionamiento de estas redes. Se puede apreciar que la salida en el primer instante ($t=0$) solo depende de la entrada en ese instante. En el instante siguiente ($t=1$), la salida depende de la entrada ($\mathbf{x}_1$) y también de la información realimentada, que resume todos los instantes anteriores. Este proceso se repite hasta obtener la salida en el instante final.

En la Figura 2.7b también se aprecia que la entrada y la salida en cada instante es un vector distinto. Esto implica que teniendo en cuenta todos los instantes, la red trabaja con una secuencia de vectores de entrada y una secuencia de vectores de salida. Esta es una de las mayores ventajas de estas redes. Una red normal se encuentra limitada a trabajar con un vector fijo de entrada y un vector fijo de salida. Gracias a esto, estas redes se pueden utilizar en problemas como la descripción de las acciones realizadas en un vídeo [24].

Uno de los problemas de estas redes se encuentra en su entrenamiento. El algoritmo más uti-

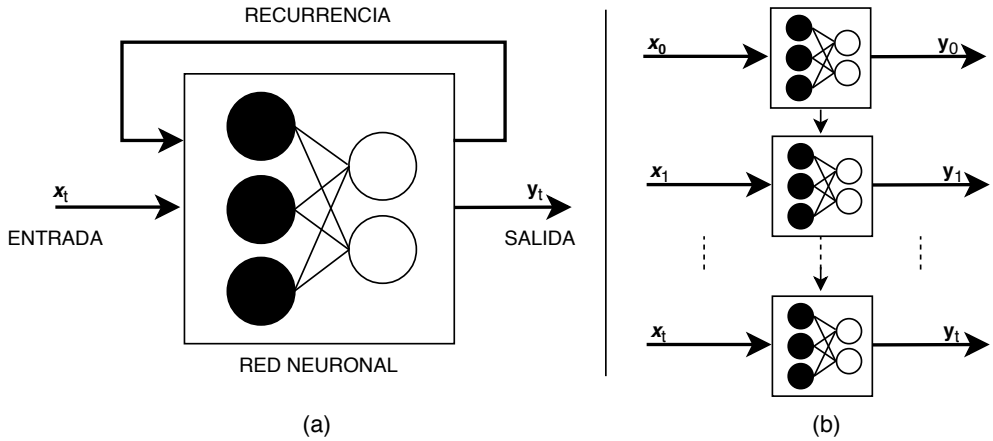

Figura 2.7: Representación esquemática típica de una red recurrente. **a** Representación plegada. Se muestra la red en un único bloque con un lazo de recurrencia. **b**: Representación desplegada. Se muestra la red dividida en tantos bloques como tamaño tenga la secuencia que procesa.

lizado para entrenar las redes neuronales se conoce como propagación hacia atrás o "*backpropagation*". Este algoritmo se basa en calcular derivadas y aplicar diversas operaciones comenzando desde la última capa de la red y retrocediendo hacia la primera capa. Gracias a la propagación hacia atrás, se pueden modificar los pesos de las neuronas para que mejoren durante el entrenamiento. El entrenamiento de las redes recurrentes se realiza con una variante de este algoritmo conocida como la propagación hacia atrás en el tiempo. El algoritmo se basa en el mismo principio pero presenta un problema, las derivadas y las operaciones dependen de instantes anteriores. Cuando ocurre esto, la información que se propaga desde el final de la red en el último instante hacia el principio de la red en el primer instante, se hace cada vez más pequeña y se desvanece. Es lo que se conoce como desvanecimiento del gradiente. Para solucionar esto, se propuso una alternativa de red que no presenta este problema, las redes de memoria a corto y largo plazo (LSTM).

Redes LSTM: Las redes de memoria a corto y largo plazo o "*Long-short term memory*"(LSTM) son un tipo de redes neuronales recurrentes propuestas por Sepp Hochreiter y Jürgen Schmidhuber en 1997 [25]. Explicar detalladamente el complejo funcionamiento de estas capas no es el objetivo de este trabajo, por lo que se van a comentar brevemente sus bloques internos. En la Figura 2.8 se muestra el funcionamiento básico de una capa LSTM. La entrada de estas capas es el estado anterior, la salida anterior y la entrada actual. La salida es el estado actual y la salida actual. El estado es un vector que contiene información relevante de los instantes anteriores y el instante actual para el funcionamiento de la capa, es su memoria.

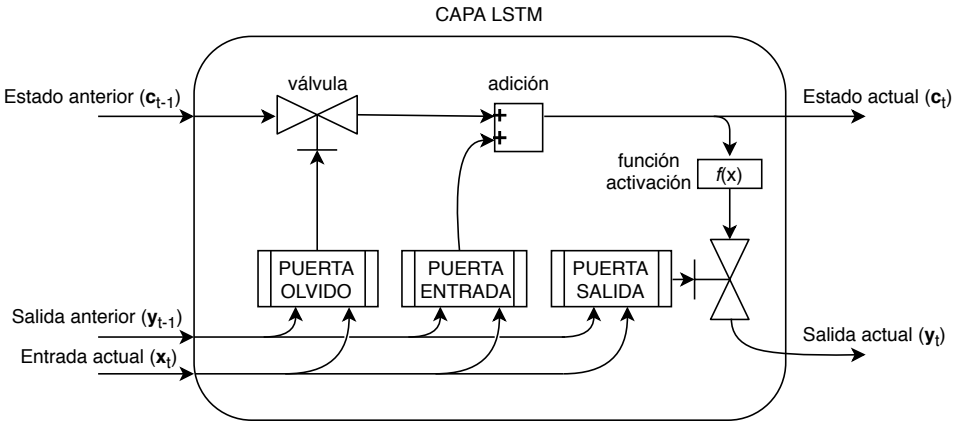

Figura 2.8: Bloques internos de una red LSTM. Se aprecian las entradas (entrada actual, salida anterior y estado anterior) y las salidas (estado y salida actual). En el interior se observan las tres puertas (puerta del olvido, puerta de entrada y puerta de salida) junto a las válvulas, las operaciones de adición y la función de activación.

La capa LSTM se compone de una serie de puertas y operaciones. Las puertas son bloques con neuronas en su interior que modifican el flujo de información en la capa y que aprenden mediante el entrenamiento. Existen tres puertas:

- Puerta del olvido o "*forget gate*": Como se observa en la Figura 2.8, esta puerta es la primera de todas. Se encarga de eliminar información del estado anterior que ya no resulte útil. Funciona como una válvula que se abre o se cierra en función de la entrada actual y la salida anterior.
- Puerta de entrada o "*input gate*": Esta capa decide la información de la entrada actual y la salida anterior que va a ser añadida al estado actual. La puerta procesa las entradas y añade el resultado al estado mediante una operación de adición.
- Puerta de salida o "*output gate*": Esta es la última puerta. Se encarga de seleccionar las partes del estado que van a ser utilizadas en la salida de la capa. Al igual que la puerta del olvido, funciona como una válvula que se abre o se cierra en función de la entrada actual y la salida anterior.

3. Reconocimiento de lugares a partir de una vista

En este capítulo se resume brevemente el TFG del que parte este trabajo para comprender mejor el desarrollo realizado.

El objetivo principal del TFG [13] fue implementar un reconocedor de lugares invariante a cambios de apariencia basado en imágenes y redes neuronales. Además del reconocedor, se desarrolló un conjunto de datos para poder entrenar y evaluar el algoritmo. Finalmente, se compararon los resultados con otros algoritmos del estado del arte.

3.1. Conjunto de datos desarrollado

Para implementar el reconocedor se compiló un conjunto extenso de imágenes de lugares con cambios de apariencia realistas y diversos. Para crear el conjunto se partió de varios vídeos del trayecto de una línea ferroviaria de Noruega. Los vídeos forman parte de un documental [26] del trayecto entre las ciudades de Trondheim y Bodø. El recorrido tiene una longitud de 729 kilómetros y se grabó desde la parte frontal del tren en invierno, primavera, otoño y verano. Utilizaron un GPS para sincronizar los cuatro vídeos, lo que permite observar la misma ubicación en las cuatro estaciones del año.

Procesando los vídeos para eliminar los túneles y las paradas del tren se extrajeron 28,865 imágenes de cada vídeo. En el TFG se dividieron los datos en dos conjuntos. Un conjunto para entrenar el algoritmo y otro conjunto de test para evaluar el funcionamiento. El conjunto de test está formado por tres tramos de 1,150 imágenes (3,450 en total) del recorrido total. En la Figura 3.1 se muestra el recorrido del trayecto y se marcan en colores distintos las partes que forman el conjunto de entrenamiento y el de test.

3.2. Redes neuronales utilizadas

En la Figura 3.2 se muestra el funcionamiento del reconocedor planteado. La entrada del algoritmo es un conjunto de imágenes con lugares visitados previamente y una imagen tomada en el lugar que se quiere reconocer. El primer paso es procesar las imágenes con la red neuronal utilizada. La salida de la red es una serie de vectores que se denominan descriptores.

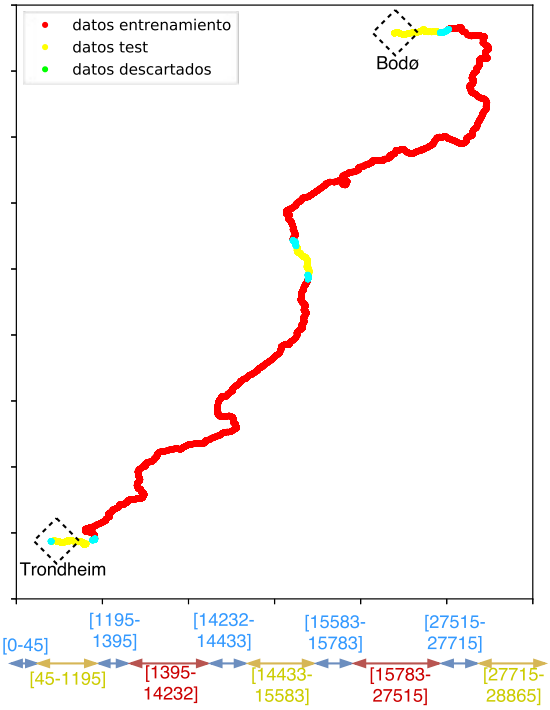

Figura 3.1: Olid, D. (2017). División propuesta del conjunto de datos Nordland. [Figura]. Recuperada del TFG [12]. División conjunto de datos Nordland. El conjunto de test se marca en amarillo, el de entrenamiento en rojo y las imágenes descartas en azul.

Los descriptores resumen la información contenida en las imágenes. Con el entrenamiento de las redes neuronales se desea que los descriptores sean invariantes a los cambios de apariencia. Buscando el descriptor del conjunto de lugares ya visitados que más se parece al descriptor del lugar a reconocer se puede detectar el lugar que aparece en la imagen [12].

Las redes neuronales utilizadas en el trabajo fueron las redes siamesas y triplets, partiendo de arquitecturas pre-entrenadas.

3.2.1. Redes pre-entrenadas (VGG-16)

En el campo de las redes neuronales es común usar redes diseñadas y entrenadas por otros investigadores. De esta forma, se puede comenzar a trabajar en el problema partiendo de red neuronal con un funcionamiento conocido y demostrado. Aunque la red original fuera entrenada para resolver un problema distinto, se puede utilizar la salida de las capas internas de la red en otros problemas [12]. En el TFG se partió de la red VGG-16 diseñada por el Oxford Geometry Group [27] y entrenada por [28] para clasificar paisajes. Esta red tiene dieciséis capas. Para utilizarla en el trabajo se estudió el funcionamiento del algoritmo reconocedor

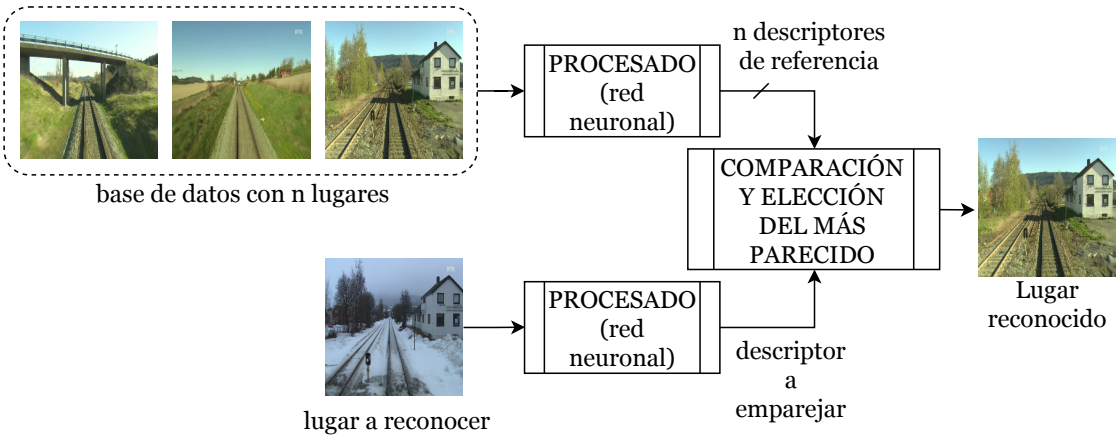

Figura 3.2: Olid, D. (2017). Estructura del reconocedor propuesto. [Figura]. Recuperada del TFG [12].

con el vector de salida de diversas capas de la red. Se concluyó que la salida de la décima capa era la mejor de todas. Los descriptores extraídos por la mayoría de las capas de las redes pre-entrenadas son muy grandes. Se hace necesario añadir nuevas capas de neuronas que reduzcan la dimensión de salida y que se entrenen para el nuevo problema.

3.2.2. Redes Siamesas y Triplets (VGG-16-Siamese y VGG-16-Triplet)

Generalmente, las redes neuronales se utilizan en problemas de clasificación y regresión. En clasificación, la salida de las redes neuronales suele ser un vector que codifica la clase a la que pertenece el dato de entrada. En regresión, la salida de la red es el valor predicho de alguna variable. No siempre es necesario que el vector de salida tenga una interpretación directa.

Las redes siamesas están formadas por dos copias de la misma red que transforman los datos de entrada en vectores donde lo relevante es la distancia entre unos vectores y otros [12]. A los vectores de salida también se les puede llamar descriptores. En la Figura 3.3a se muestra una red siamesa convolucional trabajando con el problema de la clasificación de cifras. En el caso de la figura, el descriptor extraído con la imagen del 2 debería ser más cercano a los extraídos de la misma cifra que al de la cifra 0. La medida de la distancia utilizada es la distancia euclídea.

Se puede apreciar que el descriptor de salida de la red en la Figura 3.3a tiene dimensión 2. Si se interpreta cada dimensión del vector como la posición en el eje x e y, se puede representar el descriptor extraído como un punto en un eje de coordenadas. En la Figura 3.3b se muestra la posición que ocupan los descriptores extraídos de varias cifras. Se puede apreciar que la red tiende a crear agrupaciones de los elementos de la misma clase. Cuando la salida de la red tiene más dimensiones, la representación y su comprensión se hacen más complejas pero el principio es el mismo.

En el entrenamiento de una red siamesa se procesan parejas de entradas de la misma clase

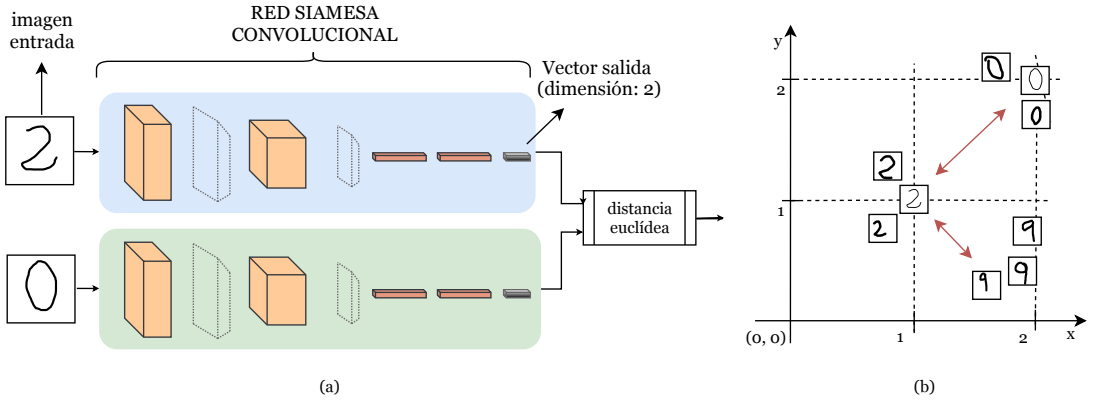

Figura 3.3: Olid, D. (2017). Red siamesa de ejemplo y representación de los descriptores. [Figura] Modificada y recuperada del TFG [12]. **a:** Red siamesa formada por varias capas convolucionales, reductoras y totalmente conectadas. **b:** Representación de los descriptores obtenidos al procesar varias imágenes de varias cifras. La posición se obtiene con el primer descriptor en el eje x y segundo en el eje y.

(parejas positivas) y parejas de entradas de clases distintas (parejas negativas) sucesivamente. El entrenamiento consiste en acercar los descriptores de la misma clase y alejar los de clases distintas [12], formando agrupaciones como las de la Figura 3.3b. El problema es que al acercar parejas de la misma clase también se pueden acercar las parejas negativas por error. La solución es tener en cuenta las parejas positivas y negativas a la vez. Para ello se crearon las redes triplets.

Redes Triplets: Las redes triplets fueron propuestas en [29] y están formadas por tres copias de la misma red. En la Figura 3.4 se representa una red triplet aplicada al problema del reconocimiento de lugares. Durante el entrenamiento de la red de la figura, la red procesa al mismo tiempo la pareja positiva (imágenes del mismo lugar en primavera e invierno) y la pareja negativa (imágenes de lugares distintos en primavera). El algoritmo de entrenamiento puede acercar el descriptor obtenido del mismo lugar y alejarlo a la vez del descriptor de lugares distintos. Gracias a esto, las redes triplets ofrecen mejores resultados que las siamesas.

Durante el entrenamiento, la entrada de estas redes es un triplete de imágenes. En este trabajo y en el TFG, los tripletes se forman emparejando dos imágenes del mismo lugar en estaciones del año distintas (p.ej., invierno-primavera) y una imagen de otro lugar cualquiera. Con el conjunto de datos de entrenamiento desarrollado se dispone aproximadamente de 840,000 tripletes.

Si bien la fase de entrenamiento de las redes siamesas y triplets utilizan dos y tres copias de la misma red, en la fase de ejecución no es necesario. Como las redes son copias con los mismos pesos y parámetros, cuando termina el entrenamiento basta con quedarse con una de las copias para procesar las imágenes y obtener los descriptores.

De todas las arquitecturas evaluadas en el TFG [13], la arquitectura que mejores resultados ofreció fue una red triplet. Se muestra la estructura en la Figura 3.5. La primera parte de la red

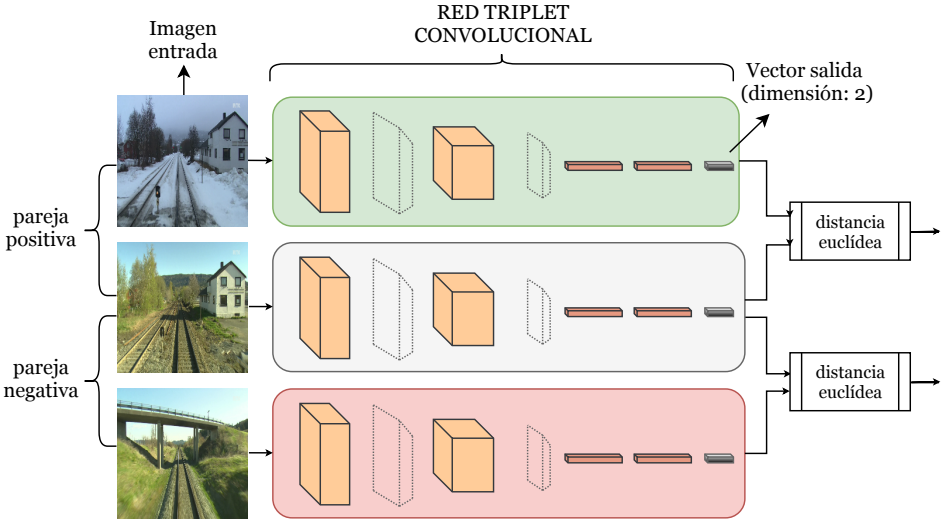

Figura 3.4: Diagrama de bloques del entrenamiento de una red triplet. Se aprecia la pareja positiva formada por dos imágenes del mismo lugar en primavera e invierno. La pareja negativa la forman dos lugares distintos en primavera.

está formada por las diez primeras capas de la red VGG-16 pre-entrenada. A la salida de la red VGG-16 se añadió una capa totalmente conectada de 128 neuronas. La salida final de la red es un descriptor de 128 dimensiones que se utiliza en el algoritmo de reconocimiento de lugares.

Cuando se parte de una red pre-entrenada y se añaden nuevas capas existen dos posibilidades. Entrenar únicamente la capa añadida y mantener los pesos de la red pre-entrenada intactos o entrenar también la red pre-entrenada para el nuevo problema (lo que se conoce como ajuste fino). En el TFG se estudiaron las dos posibilidades, la red sin ajuste fino (VGG-16-Triplet) y con ajuste fino (VGG-16-Triplet-Fine-Tuned). Se comprobó que la estructura con ajuste fino ofrecía los mejores resultados de todas las pruebas realizadas. Los resultados se mostrarán en el Capítulo 5 para poder compararlos con los alcanzados en el TFM.

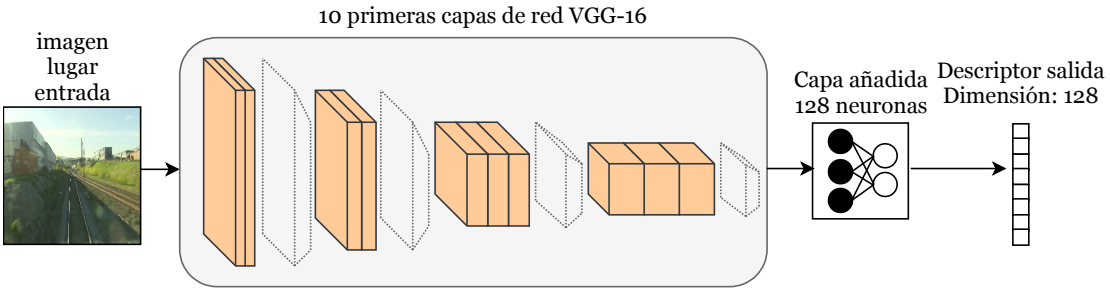

Figura 3.5: Arquitectura final utilizada en el TFG.

4. Reconocimiento de lugares a partir de múltiples vistas

En este capítulo se explican las arquitecturas neuronales que se proponen en este trabajo para resolver el problema del reconocimiento de lugares.

En la Figura 4.1 se muestra una pareja de lugares que el reconocedor basado en una única vista considera que son el mismo, a pesar de que en realidad son lugares muy alejados. Se puede apreciar que los lugares se parecen porque las vías giran a la izquierda y los árboles son parecidos. En la Figura 4.1 se muestra una secuencia de imágenes de lugares que se encuentran a varios cientos de metros. La pareja de lugares erróneamente emparejados se encuentra en vertical en el centro de las secuencias. A izquierda y derecha se muestran los lugares que se encuentran varios cientos de metros antes y después del lugar central, respectivamente. Se puede apreciar que, aunque la imagen central se parece mucho, los lugares anteriores y posteriores son muy diferentes. Utilizando la información temporal se puede lograr un reconocedor de lugares más robusto.

Figura 4.1: Lugar reconocido incorrectamente. El reconocedor estima que las dos imágenes son el mismo lugar incorrectamente. Las dos imágenes son de lugares distintos a pesar de tener una apariencia similar.

Las arquitecturas neuronales propuestas en este trabajo son las redes triplets convolucionales y las redes triplets LSTM. A continuación se va a desarrollar cada una de ellas.

Figura 4.2: Comparación de secuencias de distintos lugares. La secuencia superior está formada por lugares en verano. La secuencia inferior no tiene relación con la superior y está formada por lugares en invierno.

4.1. Redes triplets convolucionales

Estas redes se introducen en el Capítulo 3. Las redes triplets tienen como salida un descriptor donde lo importante es la distancia entre un descriptor y otro. Durante su entrenamiento, se utilizan tres copias de la misma red que procesan al mismo tiempo una pareja de elementos de la misma clase y un elemento de la clase contraria. El objetivo del entrenamiento es acercar los descriptores de la misma clase y alejar los de clases contrarias. Una vez entrenadas, basta con tomar una de las redes y procesar los datos de entrada. Midiendo las distancias entre los descriptores se pueden clasificar los datos.

En el reconocimiento de lugares, el objetivo es que los descriptores de imágenes del mismo lugar sean cercanos aunque haya cambios de apariencia. Al mismo tiempo, los descriptores de lugares distintos deben ser lejanos aunque las imágenes se parezcan. Es posible utilizar información temporal con las redes triplets convolucionales, sin ninguna capa neuronal recurrente. En esta sección se van a comentar varias técnicas para ello.

4.1.1. Redes neuronales pre-entrenadas (ResNet-50)

Como se ha visto en el Capítulo 3, las redes implementadas previamente parten de la arquitectura pre-entrenada de la red VGG-16. El campo de las redes neuronales avanza continuamente y a día de hoy existen redes alternativas que pueden llegar a funcionar mejor.

Las primeras pruebas de este trabajo consistieron en estudiar el uso de una red pre-entrenada distinta, la red ResNet-50 desarrollada por [30]. Esta red tiene cincuenta capas, frente a las dieciséis de la red VGG-16. En [31] se demuestra que esta red puede superar a la red VGG-16 en diversos problemas.

Para poder utilizar esta nueva red, se ha estudiado el funcionamiento del algoritmo reconocedor de lugares basado en una única vista utilizando el descriptor de salida de varias capas de la red ResNet-50. Dado que el número de capas es muy elevado, se comenzó descartando las primeras capas, puesto que el descriptor de salida es demasiado grande. Posteriormente, se probaron las capas en saltos de diez. Se estudiaron las capas 10, 20, 30, 40 y 50. Una vez elegida la mejor de las cinco, se estudiaron las capas próximas a ella. Finalmente, se comparó el resultado de la mejor capa de la red ResNet-50 frente al de la red VGG-16.

4.1.2. Concatenación de descriptores sin entrenamiento (ResNet-50-Concat)

El reconocedor de lugares basado en una única vista parte de una imagen de un lugar a reconocer y de un conjunto de imágenes de lugares conocidos. Para reconocer el lugar, se comienza extrayendo los descriptores usando la red neuronal. El descriptor del lugar a reconocer se compara con los descriptores de los lugares conocidos. El descriptor más cercano según la distancia euclídea es el lugar reconocido.

La aproximación SeqSLAM [11] propuso comparar secuencias en lugar de comparar descriptores uno a uno. Basándose en esta técnica, sería posible utilizar los descriptores extraídos de una red triplet convolucional. Para ello se agruparían los descriptores obtenidos de varias imágenes consecutivas y se compararían con descriptores agrupados en secuencias del mismo tamaño.

Para entender esto mejor, en la Figura 4.3 se muestra la forma de extraer los descriptores de una secuencia de tres imágenes. Se puede apreciar que la entrada a la red son tres imágenes consecutivas del trayecto. Aunque las flechas son paralelas, en la realidad se procesarían las imágenes una a una con la red neuronal y después se combinarían los resultados. Se puede apreciar que la red está formada por la primera parte pre-entrenada y una capa neuronal posterior. Al procesar tres imágenes, se obtienen tres descriptores distintos a la salida de la red pre-entrenada. Estos tres descriptores se vuelven a procesar uno por uno con la capa de neuronas añadida para obtener los tres descriptores finales de salida. En lugar de utilizar los tres por separado, se unen mediante concatenación en un único vector, tres veces más grande. Comparando ese descriptor concatenado con otros descriptores concatenados, se determina el lugar reconocido.

Esta aproximación basada en múltiples vistas es teóricamente más robusta que un reconocedor con una única vista. Aunque una de las imágenes de la secuencia presente un cambio de apariencia muy drástico, los descriptores del resto de imágenes de la secuencia pueden ayudar a que el lugar se reconozca correctamente. Una de las desventajas de este método es que el descriptor es tres veces más grande que el original. Además, la red neuronal no utiliza realmente la información temporal de la secuencia, ya que cada descriptor es independiente de la entrada procesada previamente.

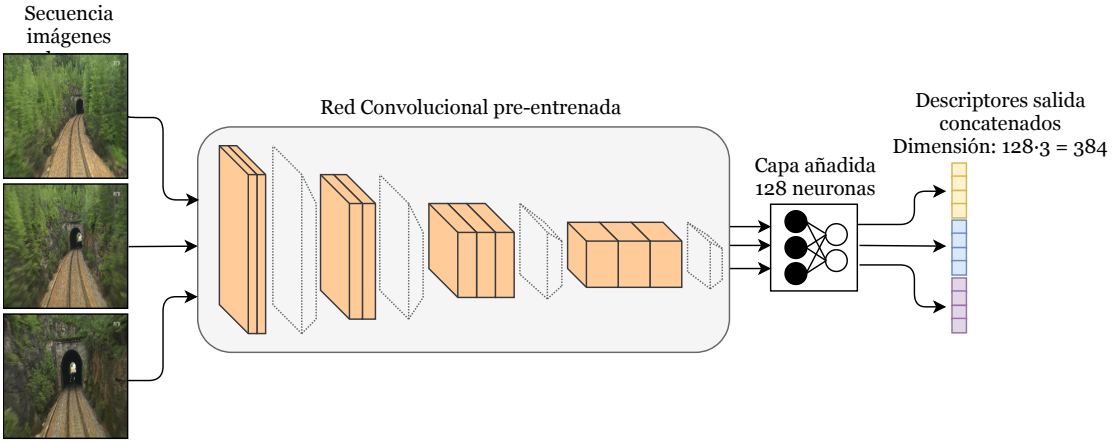

Figura 4.3: Concatenación de descriptores sin entrenamiento. Se muestra la estructura de la estrategia propuesta.

Los experimentos realizados consistieron en entrenar una estructura triplet convolucional basada en la nueva red pre-entrenada (ResNet-50-Triplet). Para ello, se añadió una capa totalmente conectada con 128 neuronas a partir de la mejor capa de la red ResNet-50. Una vez comparado el funcionamiento de la red VGG-16-Triplet con la nueva red ResNet-50-Triplet, se modificó el algoritmo del reconocedor para concatenar los descriptores y comparar las secuencias (ResNet-50-Concat).

4.1.3. Concatenación de descriptores con entrenamiento (ResNet-50-Concat-Fused)

Uno de los problemas de la aproximación previa, era que la red realmente no utiliza la información temporal para extraer los descriptores. Antes de probar las redes recurrentes, se estudió la posibilidad de entrenar las redes convolucionales con información secuencial.

En la Figura 4.3 se muestra la aproximación anterior y se puede apreciar que los descriptores concatenados son los de salida de la capa añadida. En lugar de esto, es posible entrenar la capa añadida directamente con los descriptores concatenados de la red pre-entrenada como nueva entrada. De esta forma, sin necesidad de utilizar una red recurrente, la capa añadida extraería un único descriptor que contendría información de la secuencia de imágenes completa.

En la Figura 4.4 se muestra el procedimiento para extraer el descriptor con esta nueva aproximación. Se procesarían primero las imágenes una a una con la red pre-entrenada. A continuación, se concatenarían los descriptores y se procesaría el vector obtenido con la capa añadida para obtener el descriptor final. Es importante tener en cuenta que ahora la entrada de la capa añadida es tres veces más grande. Como la capa es totalmente conectada y el número de parámetros depende del tamaño de la entrada, la capa añadida tiene el triple de parámetros aproximadamente que la capa añadida en las aproximaciones previas.

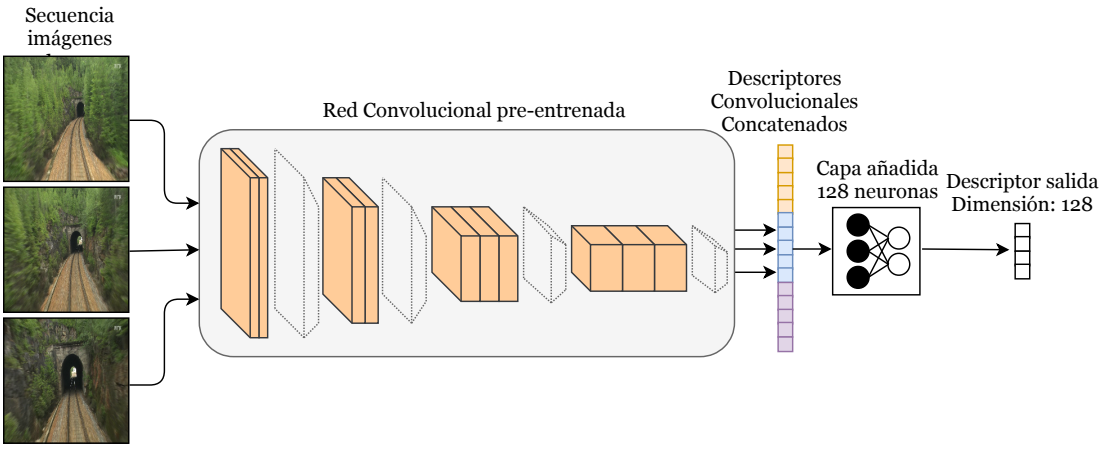

Figura 4.4: Concatenación de descriptores con entrenamiento. Se muestra la estructura de la estrategia propuesta.

Esta aproximación también debería ser más robusta que los métodos con una única vista. La ventaja principal frente al método anterior es que ahora la red es capaz de obtener un único descriptor que fusiona la información temporal de las tres imágenes de entrada. Al conectarse a los descriptores de la secuencia concatenados, la red puede decidir las partes de la imagen que presentan cambios de apariencia más drásticos y fijarse en otras zonas.

Los experimentos realizados consistieron en entrenar una estructura triplet, concatenando los vectores de salida de la red pre-entrenada ResNet-50 y añadiendo una nueva capa de 128 neuronas que aprende a utilizar todos los descriptores de la secuencia (ResNet-50-Concat-Fused).

4.2. Redes triplets LSTM (ResNet-50-Triplet-LSTM)

Las redes LSTM que se introducen en el Capítulo 2 son la opción más utilizada para trabajar con secuencias. En la Figura 4.5 se muestra la estructura propuesta para utilizar las redes LSTM en este trabajo. Se puede apreciar que la primera parte coincide con la aproximación previa. Se procesan las imágenes de la secuencia y se extraen los tres descriptores de salida de la red pre-entrenada. La diferencia se encuentra en que la capa añadida es una capa LSTM. En la figura se ha utilizado la representación desplegada. Se puede observar que el descriptor de la primera imagen es el primero en introducirse a la capa. El estado interno del primer instante se introduce, junto al descriptor de la segunda imagen, en el segundo instante. El tercer instante tiene como entradas el estado previo y el descriptor de la última imagen. La salida final de la red es la salida del último instante.

De nuevo, esta aproximación debería ser más robusta que los métodos basados en una vista. Frente a la aproximación previa, esta arquitectura también tiene como salida un único descriptor que fusiona la información temporal para ser más robusto. La diferencia principal

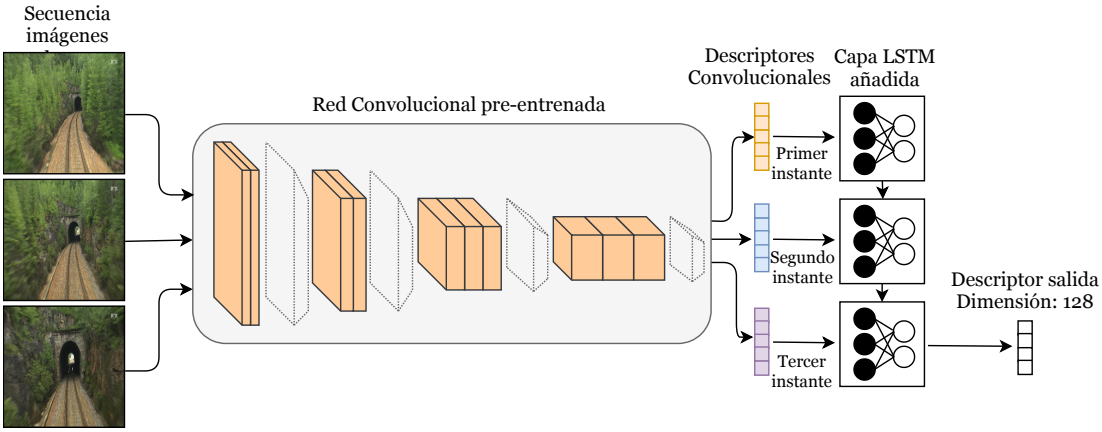

Figura 4.5: Red triplet LSTM. Se muestra la estructura de la estrategia propuesta.

es que las redes LSTM, al estar pensadas para trabajar con secuencias, son más flexibles y eficientes. Aunque en los ejemplos previos se han mostrado secuencias de tres imágenes, sería posible utilizar secuencias más largas y las redes LSTM se adaptarían mejor a ello. La desventaja principal es que las redes LSTM son más difíciles de entrenar.

Los experimentos realizados consistieron en entrenar una estructura triplet, utilizando los descriptores de salida de la red pre-entrenada ResNet-50 como entrada secuencial a una capa LSTM con un descriptor de salida con una dimensión de 128 (ResNet-50-Triplet-LSTM). Para entender mejor cómo se entrena esta arquitectura, se ha mostrado el entrenamiento en la Figura 4.6. Al ser una estructura triplet, se utilizan tres copias de la misma red. Al utilizar capas LSTM, la entrada son secuencias de imágenes. Por ello, la entrada de la red son tripletes de secuencias de imágenes. En la figura se puede apreciar la pareja positiva (secuencia del mismo lugar en primavera y verano) y la pareja negativa (secuencas de lugares distintos). El descriptor obtenido en el último instante con las redes LSTM de la pareja positiva se acerca y el de la pareja negativa se aleja.

4.2.1. Prueba de concepto

Para demostrar la capacidad de las redes LSTM de mejorar los resultados en el reconocimiento de lugares, se va a realizar una prueba con un problema más simple, el reconocimiento de cifras escritas a mano.

Para la prueba se utiliza el conjunto de datos MNIST [32]. Es un conjunto de 70,000 imágenes de cifras escritas a mano del cero al nueve. Aunque las imágenes de las cifras no tienen a priori ninguna relación temporal, es posible modificar las imágenes para transformarlas en secuencias. Se puede apreciar el proceso de transformación en la Figura 4.7. Se comienza dividiendo la imagen y posteriormente se agrupan las partes en orden. Con esta técnica se puede transformar cualquier dato en una secuencia a ser procesada por una red recurrente.

Cuando se dispone de la imagen de una cifra completa resulta sencillo determinar la cifra que aparece. Cuando se dispone de un pequeño recorte de la imagen original resulta más complicado. Es posible que haya trozos de cifras que se parezcan, como los tramos rectos del 9, el 1 o el 7. Aunque la información de cada uno de los trozos es insuficiente, la red LSTM es capaz de integrar la información de todos ellos para reconocer la cifra que aparece. Para demostrar esto se entrenan dos arquitecturas, una red triplet LSTM y una red triplet convolucional. Se muestran las dos en la Figura 4.8. La red LSTM (Figura 4.8a) se entrena con secuencias de cifras y la red convolucional (Figura 4.8b) se entrena con trozos individuales.

Para evaluar los resultados se utilizan 1,000 imágenes de cada cifra del 0 al 9. Se dividen todas ellas en cuatro partes como las de la Figura 4.7. Con la red triplet LSTM se procesan en secuencias y se extraen todos los descriptores. Con la red triplet normal se procesan los trozos individualmente. Para cada uno de los descriptores extraídos, se busca el más cercano entre el resto de descriptores según la distancia euclídea. Cuando el más cercano pertenece a otra secuencia o a otro trozo de la misma cifra, se considera como correcto. El número total de emparejamientos correctos es la métrica utilizada.

La red LSTM obtiene un 97% de emparejamientos correctos. La red normal un 60%. Estos resultados demuestran que una red LSTM es capaz de fusionar información temporal de forma efectiva en un único descriptor. En el reconocimiento de lugares se espera ver una mejora de los resultados similar.

Además de estos resultados, también se puede utilizar el ejemplo para entender mejor el funcionamiento de las redes triplets LSTM. La salida de la red es un descriptor de dimensión 2. Como se trata de redes triplets, representando los descriptores de salida de todas las secuencias en un eje de coordenadas se pueden apreciar las agrupaciones de los descriptores en dos dimensiones. En la Figura 4.9 se muestra la red LSTM desplegada procesando una secuencia y las agrupaciones de todos los descriptores extraídos. Aunque el único descriptor realmente utilizado es el del instante final, la red permite extraer los descriptores de los instantes anteriores. Las agrupaciones de descriptores tienen un color distinto para cada cifra (p. ej., ceros en rojo y cuatros en azul oscuro.). El descriptor extraído del primer instante, solo contiene información de un pequeño trozo de la imagen, por lo que los descriptores están mezclados y apenas se puede distinguir una cifra de otra. El descriptor final contiene información de toda la secuencia. Se puede apreciar que los descriptores finales forman agrupaciones donde se puede distinguir una cifra de otra claramente.

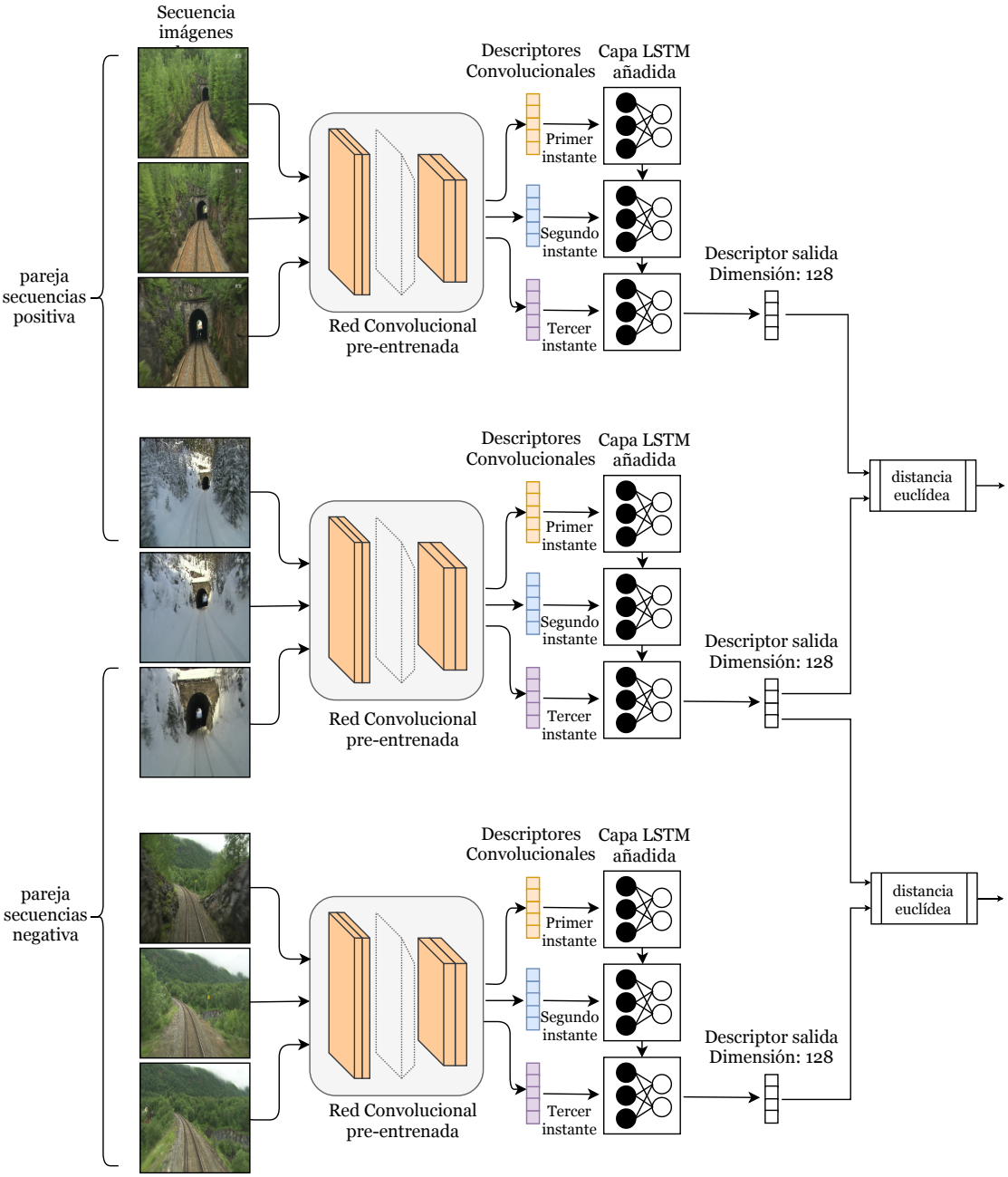

Figura 4.6: Arquitectura de la red triplet LSTM durante el entrenamiento. Se muestra la estructura de la estrategia propuesta. La imagen se aprecia mejor en color y formato electrónico.

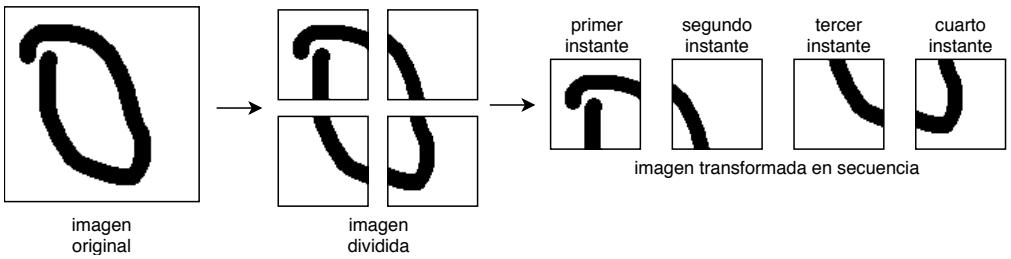

Figura 4.7: Transformación de la imagen de una cifra en una secuencia.

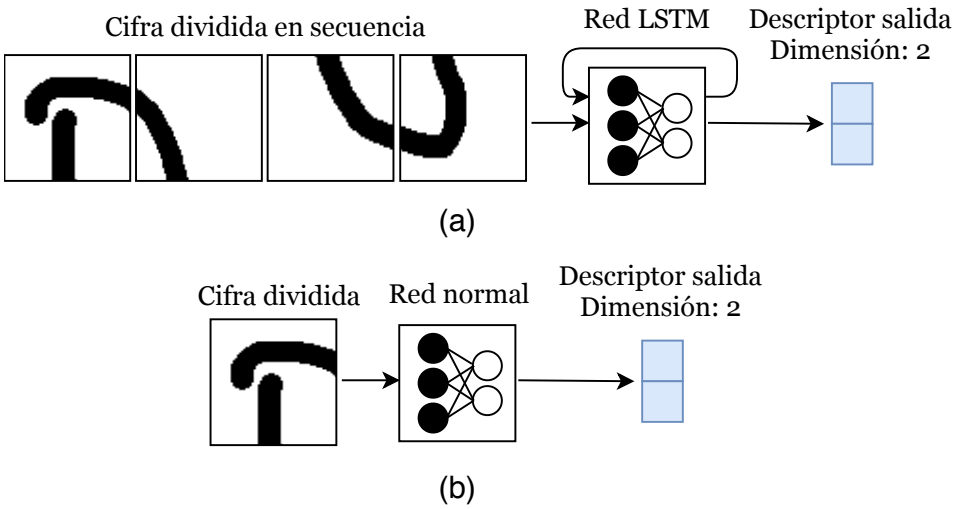

Figura 4.8: Red LSTM frente a red normal. **a:** Red LSTM procesando una secuencia de cuatro imágenes. **b:** Red normal procesando una imagen individual. El descriptor de salida en ambas redes tiene dimensión dos.

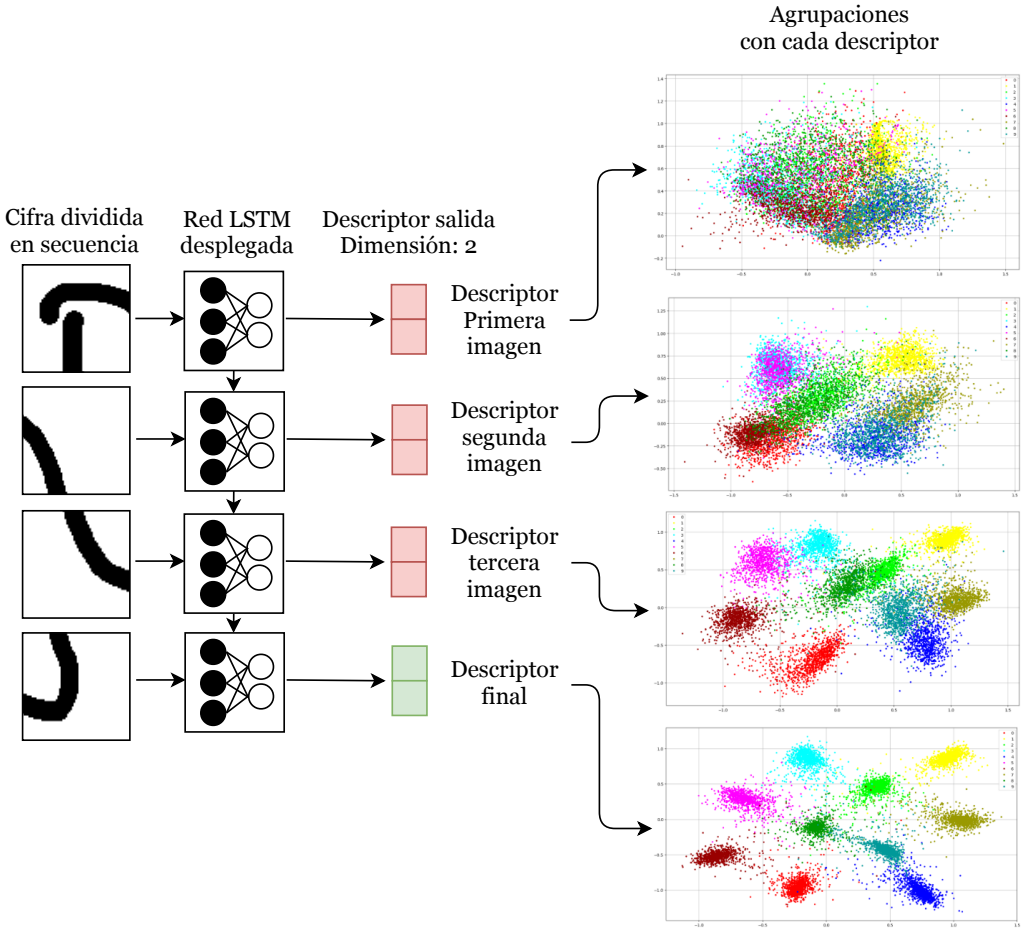

Figura 4.9: Red LSTM desplegada procesando una secuencia y agrupaciones de descriptores.

5. Resultados

En este capítulo se presentan los resultados obtenidos en este trabajo utilizando las arquitecturas mostradas en el Capítulo 4. En la Sección 5.6 se comparan todos los métodos y se dan los resultados finales.

5.1. Evaluación y métricas

El conjunto de datos utilizado para evaluar el algoritmo ha sido la división de test del conjunto Nordland presentado en la Sección 3.1 del Capítulo 3. El conjunto de test está formado por 3,450 imágenes secuenciales del mismo trayecto de tren en otoño, invierno, primavera y verano.

El proceso de evaluación del algoritmo se muestra en la Figura 5.1. La primera parte consiste en elegir la estación del año de referencia y la de entrada. La base de referencia es el conjunto de imágenes de las cuales se conoce el lugar en el que han sido tomadas. La base de entrada es el conjunto de imágenes de los lugares a reconocer. En este trabajo la mayoría de resultados se presentan usando invierno y verano como entrada o referencia. De esta forma, la nieve en invierno y los cambios de iluminación en verano suponen un reto para el reconocedor.

Posteriormente se procesan las imágenes para obtener los descriptores. En la Figura 5.1 se muestra el procesamiento de imágenes sueltas, aunque en el Capítulo 4 se ha visto que cada arquitectura propuesta tiene sus propias características. Algunas procesan imágenes sueltas, otras procesan secuencias, algunas utilizan redes normales y otras redes recurrentes. A pesar de ello, el resto del procedimiento es idéntico.

Finalmente, se compara el descriptor del lugar de entrada, con todos los descriptores de referencia. El lugar del que se ha extraído descriptor de la base de referencia a menor distancia euclídea es el lugar que el algoritmo cree haber reconocido. En este trabajo se cuenta como acierto si el lugar predicho está entre dos imágenes anteriores y dos posteriores con respecto a la imagen tomada en el mismo lugar. Contando el número total de veces que el lugar reconocido es correcto, se obtiene la métrica denominada fracción de lugares correctos (fc). Su fórmula es la siguiente:

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

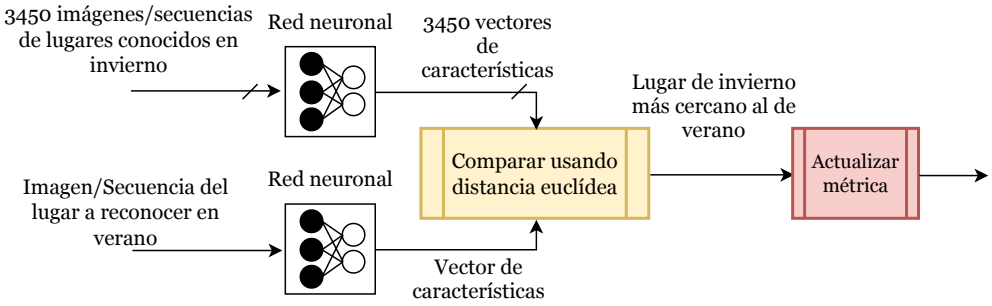

Figura 5.1: Ejemplo del proceso de evaluación del reconocedor. La base de referencia son las imágenes de los lugares en invierno. La entrada son los mismos lugares en verano.

En el caso de las estrategias que utilizan secuencias, en la Figura 5.2 se muestra el proceso de agrupación de varias imágenes consecutivas en un tramo de primavera. Se puede ver que la primera secuencia se forma con las tres primeras imágenes. Para formar la siguiente secuencia, se agrupan la segunda, tercera y cuarta imagen. Repitiendo este proceso, se transforma el conjunto de test original en un conjunto de secuencias. Los lugares que aparecen en cada instante de cada secuencia son los mismos en la misma secuencia del resto de estaciones del año (p.ej., primera secuencia de verano tiene los mismos lugares que primera secuencia de invierno, primavera y otoño.).

5.2. Redes pre-entrenadas

De las capas 10, 20, 30, 40 y 50 de la red ResNet-50, se comprobó que los mejores resultados se obtienen utilizando los descriptores de la capa 20 y la capa 30. Posteriormente, se estudiaron el resto de capas que se encuentran entre estas dos. Se pudo comprobar que el mejor resultado lo ofrece la capa 21 y el segundo mejor resultado lo ofrece la capa 23. Con los descriptores de la capa 23 se obtiene aproximadamente un 4% menos de lugares correctos que con la capa 21. Por otro lado, el tamaño del descriptor de salida de la capa 23 tiene 50,176 dimensiones, frente al de la capa 21 que tiene el doble, 100,352. Un descriptor de menor tamaño implica menor coste computacional, por ello se decide que la capa a comparar con la red VGG-16 es la capa 23 de la red ResNet-50.

En la Figura 5.3 se muestra la comparativa de los resultados utilizando la capa 23 de la red ResNet-50 frente a la capa 10 (también llamada capa POOL4) de la red VGG-16. A la izquierda se muestran los resultados con el invierno como entrada y el verano como referencia. A la derecha se muestran los resultados con el verano como entrada y el invierno como referencia. Se puede apreciar que la red ResNet-50 ofrece mejores resultados, superando a la red VGG-16 por un 12% y un 20% en cada caso.

Además de ser más robusto y funcionar mejor, el descriptor de la capa utilizada de la red

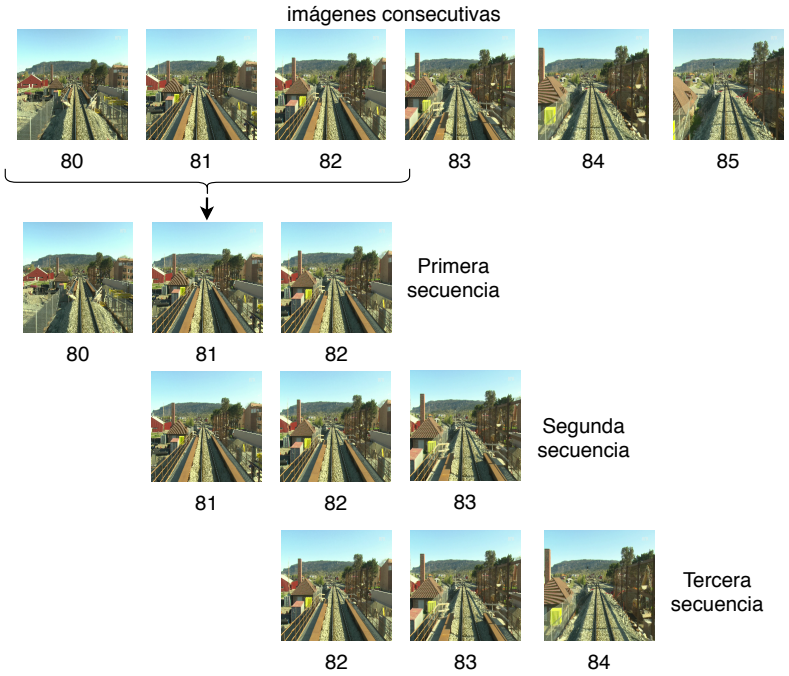

Figura 5.2: Formación de secuencias con imágenes de lugares. Se parte de varias imágenes consecutivas de lugares y se agrupan en función del tamaño de la secuencia.

ResNet-50 es más compacto (dimensión 50,176) que el de la red VGG-16 (dimensión 100,352). Por todo ello, se concluye que el punto de partida para el resto de arquitecturas es la capa 23 de la red ResNet-50.

También es importante apreciar que cuando el invierno se usa como referencia, los resultados son peores. Esto se debe a que en invierno, la nieve hace que todos los lugares se parezcan más entre sí. Cuando se compara el descriptor de entrada en verano con todos los de invierno para reconocer el lugar, hay más posibilidades de que el más parecido no sea el del lugar correcto.

5.3. Concatenación de descriptores sin entrenamiento

Aunque la concatenación de vectores no se entrena, si que es necesario obtener una red capaz de extraer los descriptores individuales. El primer paso fue entrenar una red triplet convolucional partiendo de la capa 23 de la red ResNet-50 (ResNet-50-Triplet). Se añadió una capa totalmente conectada de 128 neuronas con un descriptor de salida de dimensión 128. Entre la salida de la capa pre-entrenada y la capa totalmente conectada se utilizó una capa Dropout con el objetivo de mejorar el entrenamiento como se menciona en la Sección 2.2 del Capítulo 2.

En las pruebas realizadas tanto en este apartado como en los siguientes, a no ser que se indique lo contrario, no se realiza el ajuste fino de la red pre-entrenada y únicamente se entrena

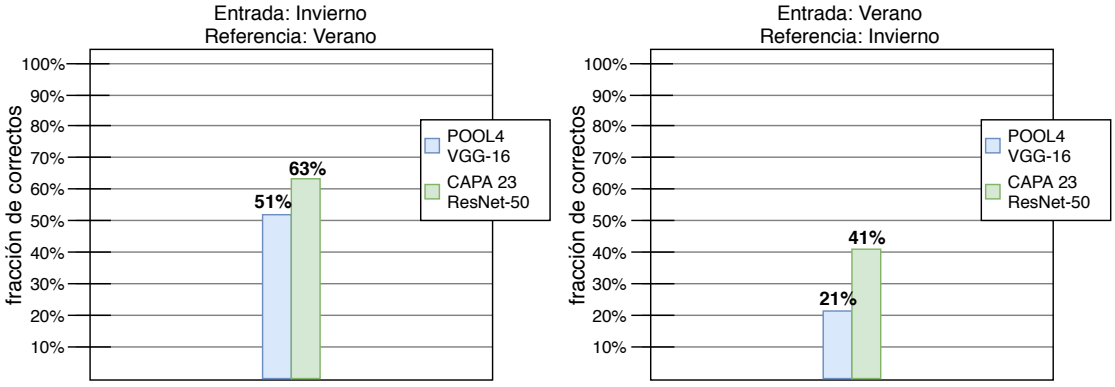

Figura 5.3: Comparativa de los resultados utilizando la capa POOL4 de la red VGG-16 (POOL4 VGG-16) frente a la capa 23 de la red ResNet-50 (CAPA 23 ResNet-50). La imagen se aprecia mejor en color y formato electrónico. **Izquierda:** Fracción de correctos usando el invierno como entrada y el verano como referencia. **Derecha:** Fracción de correctos usando el verano como entrada y el invierno como referencia.

la capa añadida. Por ello, los resultados se comparan con la red VGG-16-Triplet sin ajuste fino. También es relevante aclarar que, para poder entrenar las redes recurrentes de estrategias posteriores, se hizo necesario cambiar las librerías de redes neuronales que se utilizaron en el TFG. Por ello se volvieron a repetir los experimentos con las nuevas librerías para evaluar las distintas estrategias en igualdad de condiciones. Los resultados que se muestran a continuación, a no ser que se indique lo contrario, son los resultados obtenidos con las nuevas librerías y pueden diferir ligeramente de los obtenidos en el TFG.

Como se indica en la Sección 3.2.2 del Capítulo 3, el entrenamiento se realiza formando 840,000 tripletes de entrada de imágenes del conjunto Nordland. Se realizan diversas pruebas entrenando la red de 5 a 8 épocas[1].

En la Figura 5.4 se muestra la comparativa entre los resultados de la red VGG-16-Triplet y la red ResNet-50-Triplet. También se muestran los resultados obtenidos previamente con los descriptores de las capas pre-entrenadas para apreciar la mejora obtenida con la capa añadida. Ambas redes tienen como salida un descriptor de 128 dimensiones y fueron entrenadas durante un número similar de épocas. Se observa que la red triplet basada en la red ResNet-50 tiene mejor desempeño que la basada en la red triplet VGG-16 en todas las combinaciones (77% vs 75% en invierno-verano y 75% vs 72% en verano-invierno). Por ello se decide utilizar la red ResNet-50 con una capa añadida y un descriptor de salida de 128 dimensiones como punto de partida para la concatenación de descriptores.

También es importante apreciar que las redes triplets mejoran drásticamente con respecto a la utilización del descriptor de las redes pre-entrenadas. Esto se aprecia especialmente usando invierno como referencia, donde se pasa de obtener un 41% con la red ResNet-50 a un 75%

[1]Una época es el intervalo del entrenamiento en el que se procesan todos los datos de entrada una vez. En este caso los 840,000 tripletes.

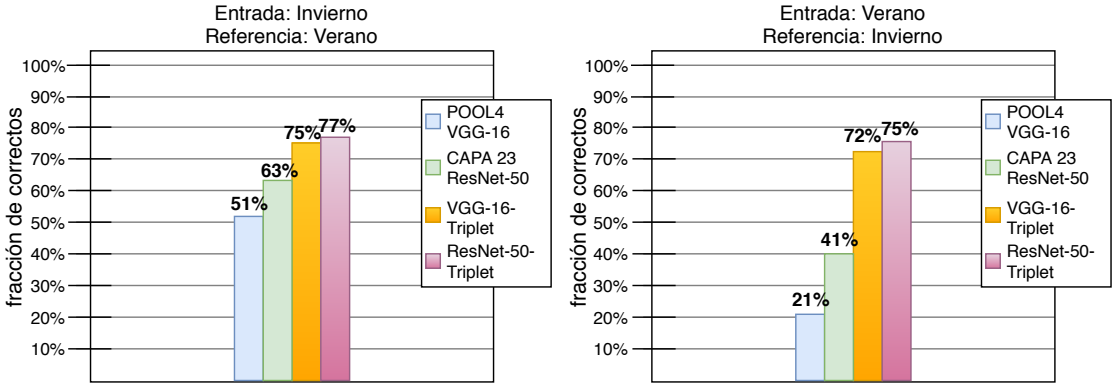

Figura 5.4: Comparativa de los resultados utilizando la capa POOL4 de la red VGG-16 (POOL4 VGG-16) y la estructura triplet entrenada desde esa red (VGG-16-Triplet) frente a la capa 23 de la red ResNet-50 (CAPA 23 ResNet-50) y la estructura triplet entrenada desde esa red (ResNet-50-Triplet). La imagen se aprecia mejor en color y formato electrónico. **Izquierda:** Fracción de correctos usando el invierno como entrada y el verano como referencia. **Derecha:** Fracción de correctos usando el verano como entrada y el invierno como referencia.

con la red ResNet-50-Triplet. Esto demuestra que el entrenamiento es capaz de mejorar la invarianza de los descriptores obtenidos a los cambios de apariencia.

El siguiente paso es modificar el algoritmo para concatenar los descriptores de salida de la red obtenida y utilizar la información temporal. En la Figura 5.5 se muestra la comparación de los resultados previos utilizando un único descriptor (ResNet-50-Triplet) frente a la concatenación de tres descriptores (ResNet-50-Concat). Se puede apreciar que la mejora de los resultados concatenando descriptores es drástica en las dos combinaciones. Se obtiene un incremento de al menos el 15% en los lugares reconocidos correctamente. En el caso de invierno como entrada, el descriptor individual obtiene un 72%, frente al 92% que se obtiene concatenando los descriptores. Esto confirma la hipótesis de partida del trabajo, al usar la información temporal se mejora la robustez del algoritmo aunque los descriptores individuales no sean suficientemente invariantes a los cambios de apariencia.

También se estudió la posibilidad de incrementar el tamaño de la secuencia. Concatenando diez descriptores en lugar de tres, se mejoran los resultados un 2%. El problema es que a mayor tamaño de secuencia, mayor coste computacional y la mejora no merece la pena. Por eso se decidió que el resto de pruebas se realizarían con secuencias de tres imágenes.

5.4. Concatenación de descriptores con entrenamiento

Esta estrategia es la primera que se entrena específicamente con secuencias siguiendo la arquitectura desarrollada en la Sección 4.1.3 del Capítulo 4. Se utiliza una estructura triplet, concatenando los vectores de salida de la red pre-entrenada ResNet-50 y se añade una nueva

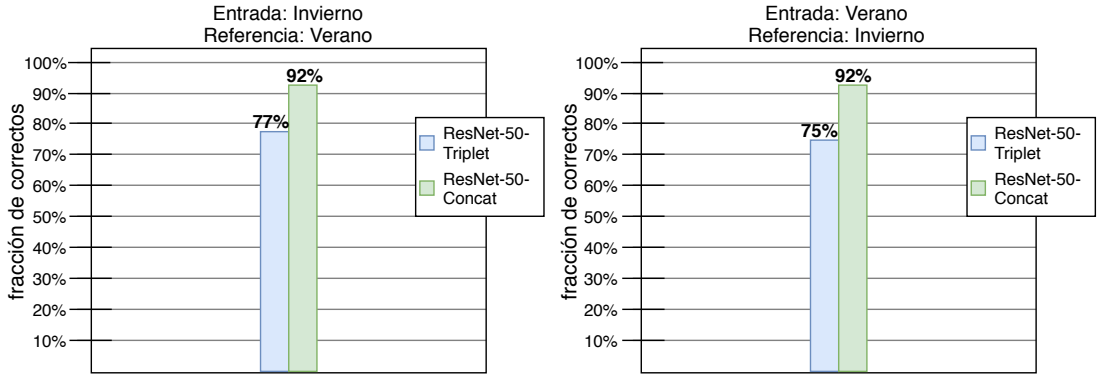

Figura 5.5: Comparativa de los resultados obtenidos utilizando un único descriptor de salida (ResNet-50-Triplet) y los resultados concatenando descriptores (ResNet-50-Concat). La imagen se aprecia mejor en color y formato electrónico. **Izquierda:** Fracción de correctos usando el invierno como entrada y el verano como referencia. **Derecha:** Fracción de correctos usando el verano como entrada y el invierno como referencia.

capa de 128 neuronas que aprende a utilizar todos los descriptores de la secuencia.

La red se entrena con tripletes de secuencias de tres imágenes. Es decir, dos secuencias del mismo lugar con distinta apariencia y una secuencia de otro lugar. Se realizan varias pruebas entrenando la red de 5 a 8 épocas con 840,000 tripletes de secuencias de entrada.

En la Figura 5.6 se muestra la comparación de los resultados de la sección previa frente a la red entrenada en este apartado para utilizar las secuencias (ResNet-50-Concat-Fused). Se puede observar que la red entrenada con secuencias vuelve a superar a la red que utiliza un único descriptor (ResNet-50-Triplet) por más de un 10%. En el caso del invierno como entrada, la red ResNet-50-Triplet obtiene un 77% frente al 87% de la red entrenada con secuencias. A pesar de ello, la concatenación de descriptores (ResNet-50-Concat) supera a la red entrenada con secuencias (ResNet-50-Concat-Fused) por más de un 5% en todos los casos. En la Sección 5.6 se analizan las diferencias de las estrategias en profundidad.

5.5. Redes triplets LSTM

Esta estrategia, propuesta en la Sección 4.1.3 del Capítulo 4, consiste en utilizar redes recurrentes de tipo LSTM. Se entrena una estructura triplet, concatenando los vectores de salida de la red pre-entrenada ResNet-50 y se añade una nueva capa LSTM que aprende a utilizar todos los descriptores de forma secuencial. El descriptor utilizado para realizar el reconocimiento de lugares es el de salida de la capa LSTM en el último instante temporal. Se utilizan secuencias de tres imágenes y el descriptor de salida tiene una dimensión de 128.

La red se entrena con tripletes de secuencias de tres imágenes. Se realizan varias pruebas entrenando la red de 5 a 8 épocas con 840,000 tripletes de secuencias de entrada. Una vez se

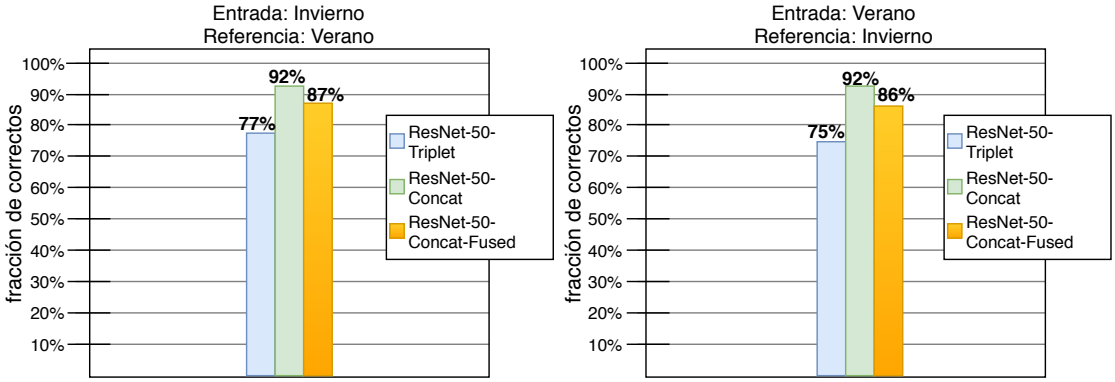

Figura 5.6: Comparativa de los resultados utilizando una red triplet entrenada con secuencias (ResNet-50-Concat-Fused) frente a la red triplet usando un único descriptor (ResNet-50-Triplet) y la concatenación de descriptores sin entrenamiento (ResNet-50-Concat). La imagen se aprecia mejor en color y formato electrónico. **Izquierda:** Fracción de correctos usando el invierno como entrada y el verano como referencia. **Derecha:** Fracción de correctos usando el verano como entrada y el invierno como referencia.

obtienen los primeros resultados, se realizan varias pruebas alternativas para tratar de mejorar los resultados. Se estudian las siguientes modificaciones:

- **Modificar el tamaño de la secuencia:** Se estudia el funcionamiento de la red LSTM si se incrementa la secuencia de 3 a 6 y 10 imágenes.
- **Introducir Dropout recurrente:** Se estudia la posibilidad de introducir capas Dropout en la parte recurrente de una red LSTM. En lugar de eliminar conexiones con la capa previa durante el entrenamiento, el Dropout recurrente elimina conexiones con la información de salida que se vuelve a introducir a la capa en cada instante.
- **Modificar las secuencias con lugares aleatorios:** Una secuencia normal se compone de los lugares en orden: [Lugar 1, Lugar 2, Lugar 3]. Se estudia la posibilidad de introducir lugares aleatorios en las secuencias de entrenamiento de tipo: [Lugar 45, Lugar 2, Lugar 3]. En ese caso, la red debería ignorar la información del primer lugar de la secuencia. Es posible que esto ayude al entrenamiento de la red.

En la Tabla 5.1 se muestran los resultados de las diversas pruebas con redes LSTM. En cuanto a cambiar el tamaño de las secuencias, se aprecia que la red LSTM básica (LSTM Secuencia 3) es la que mejor funciona de todas. Aumentar el tamaño de las secuencias (LSTM Secuencia 6 y 10) empeora el resultado. Esto puede deberse al hecho de que las redes LSTM son complicadas de entrenar y a mayor tamaño de secuencia, los problemas se agravan. El resto de pruebas se realizan con una secuencia de tres imágenes.

El Dropout recurrente (LSTM Dropout Recurrente) consigue mejorar un 3% la fracción de correctos y alcanza el 84,13%. Esto tiene sentido ya que el Dropout ayuda a mejorar el

Tabla 5.1: Fracción de correctos de las pruebas LSTM usando invierno como referencia y verano como entrada.

Prueba	Fracción de correctos (Entrada: Verano. Referencia: Invierno.)
LSTM Secuencia 3	81,78%
LSTM Secuencia 6	81,24%
LSTM Secuencia 10	73,67%
LSTM Dropout Recurrente	84,13%
LSTM Dropout + Primera Aleatoria	85,79%

entrenamiento. Las pruebas modificando las secuencias con lugares aleatorios se realizaron incluyendo también el Dropout recurrente con el objetivo de incrementar todavía más la mejora.

Modificando las secuencias con lugares aleatorios y añadiendo Dropout recurrente (LSTM Dropout + Primera Aleatoria) se consigue alcanzar un 85,79%. Introducir lugares aleatorios en las secuencias durante el entrenamiento fuerza a la red LSTM a considerar los tres instantes temporales con la misma importancia. Cuando no se introducen lugares aleatorios, la red puede tender a utilizar únicamente información de uno de los lugares. Se considera que la mejor red para el resto de comparaciones es la red LSTM utilizando Dropout recurrente y modificando las secuencias. En el resto del capítulo esta red aparecerá referida como ResNet-50-Triplet-LSTM.

En la Figura 5.7 se muestra la comparación de los resultados de las secciones previas frente a la red LSTM (ResNet-50-Triplet-LSTM). La red LSTM solo consigue superar a la red triplet utilizando un único descriptor (ResNet-50-Triplet). En el caso del invierno como entrada, la red LSTM obtiene un 85% de correctos frente al 77% de la red ResNet-50-Triplet. A pesar de ello, la red LSTM obtiene peores resultados que la concatenación simple de descriptores (ResNet-50-Concat) y la red entrenada con secuencias (ResNet-50-Concat-Fused). En la Sección 5.6 se analizan las diferencias de las estrategias en profundidad.

5.6. Comparativa

En la Figura 5.8 se muestran los resultados obtenidos con las estrategias utilizadas en este trabajo junto a la mejor del TFG (VGG-16-Triplet-Fine-Tuned). En esta sección, se muestran los resultados del TFG originales y no las pruebas realizadas con las nuevas librerías, para poder evaluar si se mejoran los resultados realmente. La mejor fracción de correctos se obtiene con la concatenación simple de descriptores sin entrenamiento (ResNet-50-Concat) que alcanza un 92% en todas las combinaciones. Se observa que la peor de las tres estrategias de este trabajo es la red triplet utilizando un único descriptor (ResNet-50-Triplet) y la peor de las que utiliza secuencias es la red LSTM (ResNet-50-Triplet-LSTM). A pesar de ello, la red LSTM consigue empatar con la red VGG-16-Triplet-Fine-Tuned, consiguiendo ambas un 86% con el invierno como referencia.

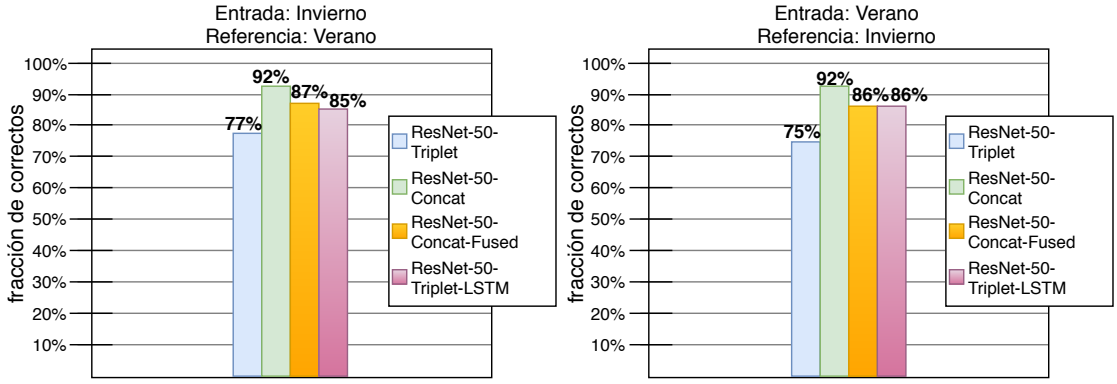

Figura 5.7: Comparativa de los resultados obtenidos utilizando la red LSTM (ResNet-50-Triplet-LSTM), la red triplet normal (Triplet ResNet-50), la concatenación de descriptores sin entrenamiento (Triplet Concat ResNet-50) y la red triplet entrenada con secuencias (Triplet Train Concat ResNet-50). La imagen se aprecia mejor en color y formato electrónico. **Izquierda:** Fracción de correctos usando el invierno como entrada y el verano como referencia. **Derecha:** Fracción de correctos usando el verano como entrada y el invierno como referencia.

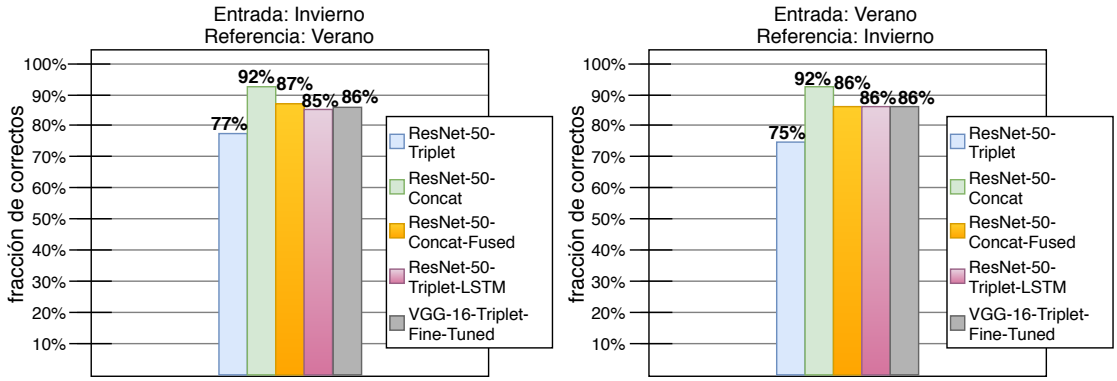

Figura 5.8: Comparativa de los resultados obtenidos en este trabajo utilizando un único descriptor de la red triplet (ResNet-50-Triplet), la concatenación de descriptores sin entrenamiento (ResNet-50-Concat), la red triplet entrenada con secuencias (ResNet-50-Concat-Fused) y la red LSTM (ResNet-50-Triplet-LSTM) frente a la red triplet con ajuste fino VGG-16-Triplet-Fine-Tuned. La imagen se aprecia mejor en color y formato electrónico. **Izquierda:** Fracción de correctos usando el invierno como entrada y el verano como referencia. **Derecha:** Fracción de correctos usando el verano como entrada y el invierno como referencia.

Cabe destacar que los resultados de la red VGG-16-Triplet-Fine-Tuned se consiguieron con el ajuste fino de la red pre-entrenada. El problema del ajuste fino es que hace que la red funcione mejor con datos como los del conjunto Nordland, pero puede empeorar su funcionamiento con datos distintos a los del entrenamiento. Este problema se conoce como sobreajuste ("*overfitting*") y se demostrará en la Sección 5.8. Por ello, se decidió que en este trabajo no se realizaría el ajuste fino. En su lugar, se estudia si las nuevas estrategias pueden superar los

resultados gracias a la información temporal, sin las desventajas que el ajuste fino conlleva.

Antes de poder extraer las conclusiones finales es necesario comprobar el funcionamiento de las distintas estrategias en todas las circunstancias posibles. En las pruebas anteriores, se ha supuesto que las secuencias de entrada y las de referencia se han tomado en las mismas condiciones (secuencias sincronizadas). Cuando se usan secuencias hay que tener en cuenta aspectos como la posibilidad de que se tomen a distinta velocidad o incluso en sentidos contrarios. Se han realizado varias pruebas con las siguientes condiciones:

- **Test con una secuencia invertida:** Esta prueba estudia la posibilidad de que una de las secuencias se tome en sentido inverso. Se muestra un ejemplo de ello en la Figura 5.9a. Los índices de los lugares en cada secuencia serían: Entrada invertida: [Lugar 3, Lugar 2, Lugar 1] y Referencia normal: [Lugar 1, Lugar 2, Lugar 3].
- **Test con salto par/impar:** Esta prueba estudia la posibilidad de que las imágenes de las secuencias no contengan exactamente el mismo lugar. Se muestra un ejemplo de ello en la Figura 5.9b. Los índices de los lugares serían: Entrada: [Lugar 1, Lugar 3, Lugar 5] y Referencia: [Lugar 2, Lugar 4, Lugar 6].
- **Test con velocidad aleatoria:** Esta prueba estudia la posibilidad de que las secuencias se tomen a velocidades aleatorias y distintas. Los índices de varias secuencias serían Input: [1 2 4...5 6 7...9 11 13...] y Referencia: [1 3 5...6 7 8...10 11 13...].

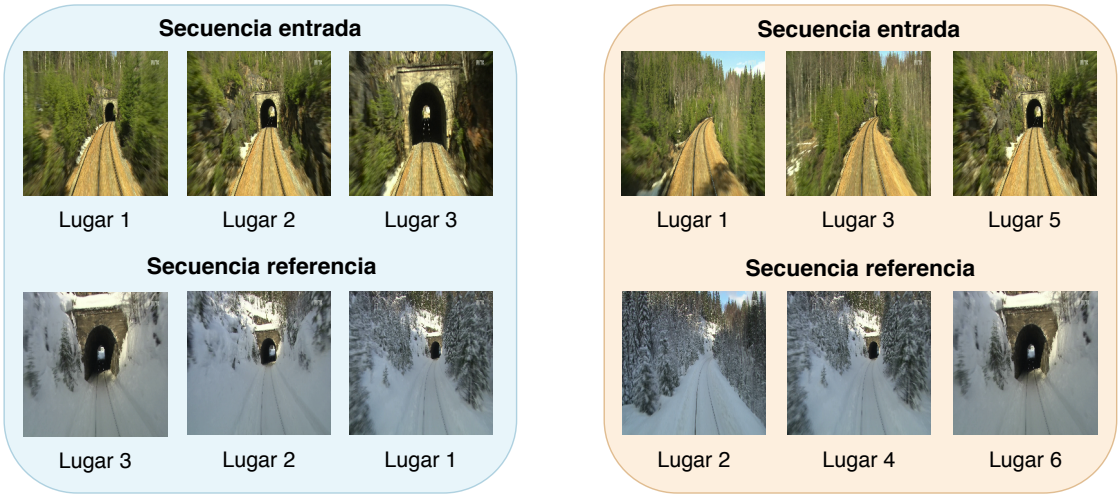

Figura 5.9: Ejemplos de dos tipos de secuencias complicadas. **a:** Ejemplo de secuencia invertida en el que la secuencia de referencia ha sido tomada en sentido contrario a la de entrada. **b:** Ejemplo de secuencia con salto par/impar en el que las imágenes de las secuencias no coinciden exactamente.

En la Figura 5.10 se muestran los resultados de las tres estrategias que usan secuencias en todas las pruebas con secuencias complejas realizadas. Se observa que la red LSTM (Triplet

LSTM Drop + Aleat) es la que menos empeora cuando una de las secuencias se encuentra invertida. En el resto de casos, la concatenación de descriptores sin entrenamiento (Triplet Concat ResNet-50) es la mejor de todas.

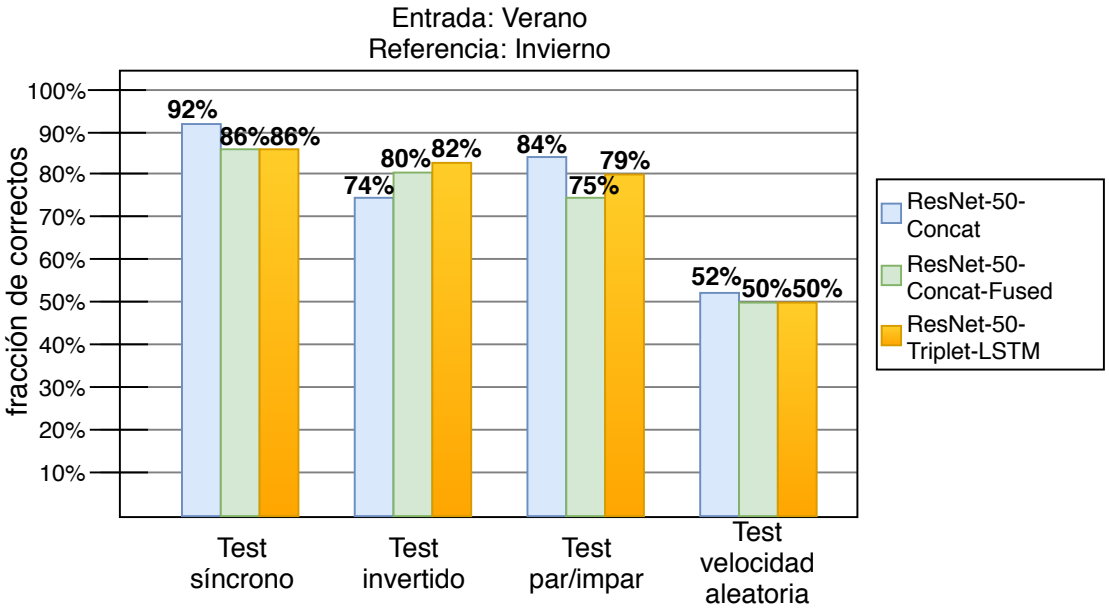

Figura 5.10: Comparativa de los resultados obtenidos utilizando la concatenación de descriptores sin entrenamiento (ResNet-50-Concat), la red triplet entrenada con secuencias (ResNet-50-Concat-Fused) y la red LSTM (ResNet-50-Triplet-LSTM) ante pruebas complejas. Se muestran los resultados del test sincronizado, el invertido, usando salto par/impar y el test de velocidad aleatoria usando el verano como entrada y el invierno como referencia. La imagen se aprecia mejor en color y formato electrónico.

Considerando todos los resultados obtenidos se puede afirmar que la mejor estrategia de todas las utilizadas consiste en entrenar una red triplet convolucional, sin considerar información temporal. Partiendo de la red pre-entrenada ResNet-50 y concatenando posteriormente los descriptores obtenidos de varias imágenes se mejoran drásticamente los resultados.

Cuando se entrena una red neuronal para combinar la información temporal en un único descriptor, los resultados empeoran. Esto se debe a varios motivos. Por un lado, los descriptores individuales son más discriminativos que un descriptor que combina la información de varios. Además, las redes LSTM son más complicadas de entrenar y es posible que no alcancen los resultados óptimos. A pesar de ello, la información temporal siempre mejora los resultados frente a las aproximaciones que no la utilizan.

Por otro lado, las redes LSTM y las redes entrenadas para procesar descriptores concatenados son capaces de combinar información en un único descriptor, lo cual conlleva ventajas en cuanto al coste computacional. También es importante mencionar que las redes LSTM son más robustas cuando las secuencias contienen cambios drásticos como el hecho de que el

sentido del recorrido cambie. Esto se debe a que las redes LSTM están diseñadas para tener en cuenta la información temporal durante su entrenamiento. Cuando las secuencias contienen cambios más simples, la estrategia ganadora sigue siendo la concatenación de descriptores. Esto se debe a que varios descriptores concatenados contienen información más discriminativa que un único descriptor que combina información temporal.

5.7. Estado del arte

Este trabajo se puede comparar con tres técnicas que se consideran estado del arte en reconocimiento de lugares: el análisis de componentes principales (PCA) [33], el entrenamiento de dos redes neuronales para clasificar lugares con reducción multiescala *"multiscale pooling"* [8] y el entrenamiento de redes triplets para reconocimiento de lugares invariante a la apariencia [9]. El artículo de la técnica PCA [33] utiliza la misma métrica que este trabajo. Los desarrolladores de las redes con reducción multiescala [8] publicaron sus dos redes neuronales y se pueden comprobar sus resultados. El artículo de las redes triplets invariantes a la apariencia [9] utiliza métricas distintas que no se pueden comparar de forma rigurosa y la comparación solo es teórica en cuanto a técnicas utilizadas.

En la Figura 5.11 se muestran los resultados obtenidos utilizando la mejor estrategia de este trabajo, la concatenación de descriptores sin entrenamiento (ours), frente a las redes neuronales con reducción multiescala (Hybridnet y Amosnet) y la aproximación PCA. Se puede observar que la aproximación de este trabajo consigue una mejora de más del 15% con respecto a la segunda más cercana (Hybridnet) usando como referencia el verano. Los resultados son especialmente llamativos usando como referencia el invierno, donde la técnica PCA llega al 66% y la aproximación de este trabajo al 92%.

Este trabajo también se puede comparar con el algoritmo de reconocimiento de lugares que mejores resultados ofrece utilizando información secuencial, SeqSLAM [11]. Esta aproximación se basa en utilizar algoritmos de visión por computador para extraer descriptores de las imágenes. Comparando los descriptores concatenados en secuencias, de forma similar a como se hace en este trabajo, se reconocen los lugares. Este algoritmo tiene una serie de parámetros que se pueden configurar, entre ellos el tamaño de la secuencia. Para obtener los resultados que se muestran en la Figura 5.12 se utilizó la implementación en Matlab del algoritmo [34].

En la Figura 5.12 se muestran los resultados obtenidos utilizando la mejor estrategia de este trabajo, la concatenación de descriptores sin entrenamiento (ours), y la aproximación SeqSLAM (SeqSLAM s = 3, 10 y 100) con varios tamaños de secuencia (tres, diez y cien). Se puede apreciar que el algoritmo SeqSLAM mejora conforme se incrementa el tamaño de las secuencias. Con secuencias de tamaño tres (mismo tamaño que las de la aproximación de este trabajo), SeqSLAM apenas llega al 33%. Solo con secuencias de tamaño cien se supera el resultado de este trabajo. A pesar de ello, utilizar secuencias de tamaño cien tiene un coste computacional superior a utilizar secuencias de tres imágenes.

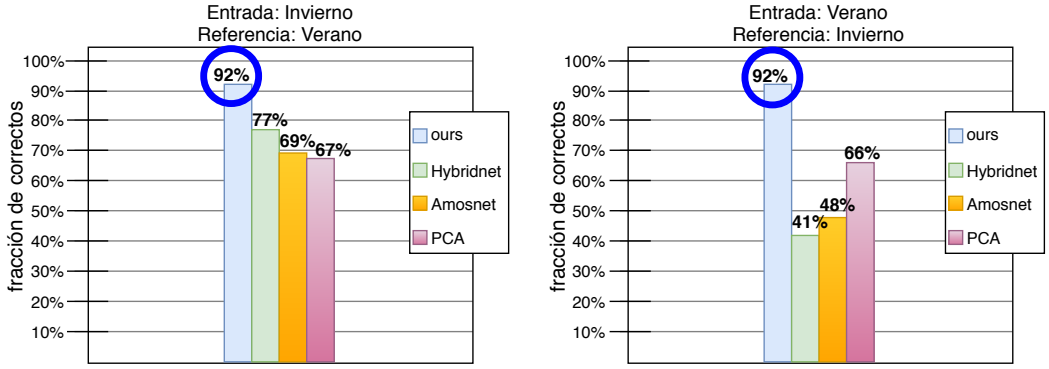

Figura 5.11: Comparativa de los resultados obtenidos utilizando la mejor estrategia de este trabajo, la concatenación de descriptores sin entrenamiento (ours), frente a las redes neuronales con reducción multiescala (Hybridnet y Amosnet) y la aproximación PCA. La imagen se aprecia mejor en color y formato electrónico. **Izquierda:** Fracción de correctos usando el invierno como entrada y el verano como referencia. **Derecha:** Fracción de correctos usando el verano como entrada y el invierno como referencia.

La implementación de SeqSLAM [34] con la que se han obtenido los resultados utiliza como descriptores las propias imágenes, reduciendo su tamaño a 64×32 píxeles y en escala de grises. El valor de los píxeles se almacena con un byte por lo que cada descriptor de SeqSLAM ocupa $64 \times 32 \times 1 = 2048$ bytes. Las secuencias de cien descriptores ocupan por tanto 204,800 bytes. El descriptor individual de las distintas estrategias de este trabajo tiene una dimensión de 128 y cada valor se codifica con 4 bytes. Cada descriptor ocupa por tanto $128 \times 4 = 512$ bytes. En el caso de los descriptores concatenados de tres en tres, cada secuencia ocupa 1,536 bytes. Las secuencias utilizadas en este trabajo ocupan aproximadamente 130 veces menos espacio en memoria que las secuencias a utilizar con SeqSLAM para obtener resultados competitivos.

También es importante mencionar que, aunque depende de la implementación, si se comparan secuencias hay que esperar a tener todas las imágenes de la secuencia de entrada para comenzar el reconocimiento de lugares. Utilizar secuencias de tamaño cien requiere más tiempo de espera que cuando la secuencia tiene tamaño tres. En las aplicaciones de navegación autónoma tener que esperar para comenzar el reconocimiento puede tener consecuencias drásticas.

Se puede concluir que los descriptores obtenidos por redes neuronales como las de este trabajo también superan a estrategias del estado del arte en reconocimiento de lugares utilizando información temporal.

En la Tabla 5.2 se muestran los resultado obtenidos con la mejor estrategia de este trabajo, la concatenación de descriptores sin entrenamiento , en todas las combinaciones posibles entrada-referencia del conjunto Nordland.

En la Figura 5.13 se muestran varios ejemplos de los lugares reconocidos utilizando una única vista frente a los lugares reconocidos utilizando varias vistas. Se puede apreciar que cuando se utiliza una sola vista los lugares recuperados son incorrectos. Es importante

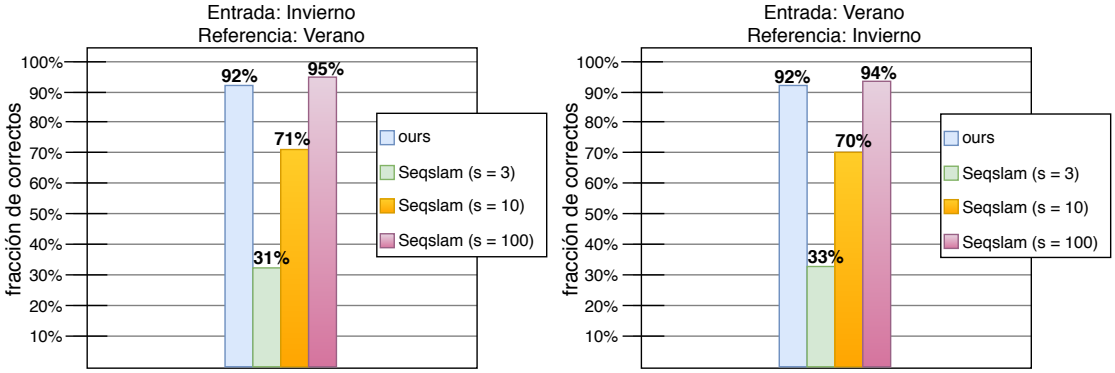

Figura 5.12: Comparativa de los resultados obtenidos utilizando la mejor estrategia de este trabajo, la concatenación de descriptores sin entrenamiento (ours), frente a la aproximación SeqSLAM con varios tamaños de secuencia (Seqslam s = 3, 10 y 100). La imagen se aprecia mejor en color y formato electrónico. **Izquierda:** Fracción de correctos usando el invierno como entrada y el verano como referencia. **Derecha:** Fracción de correctos usando el verano como entrada y el invierno como referencia.

Tabla 5.2: Fracción de correctos de la concatenación de descriptores sin entrenamiento en todas las combinaciones de estaciones.

entrada\referencia	verano	otoño	invierno	primavera
verano	—	97,85%	91,62%	98,54%
otoño	97,82%	—	91,64%	97,88%
invierno	91,76%	90,72%	—	92,27%
primavera	98,12%	97,82%	92,12%	—

mencionar que a pesar de ser erróneos, el lugar de entrada y el reconocido utilizando una vista presentan paisajes con características similares. Al utilizar varias vistas, el reconocedor funciona correctamente. Aunque alguna de las imágenes no contenga información suficiente, el reconocedor utiliza toda la secuencia para identificar el lugar.

5.8. Conjuntos de datos adicionales

Uno de los problemas del reconocimiento de lugares es que los cambios de apariencia son muy diversos. Los cambios que aparecen en el conjunto Nordland con el que se han hecho las pruebas vienen dados por fenómenos como la nieve, la lluvia o la iluminación. Para comprobar que el reconocedor de lugares se puede adaptar a otros cambios se hace necesario realizar pruebas en otros conjuntos de datos. En este trabajo se ha decidido comprobar el funcionamiento del reconocedor en los conjuntos Alderley y Santa Lucía.

Tabla 5.3: Fracción de correctos de las pruebas en el conjunto Alderley usando el día como entrada y la noche como referencia.

Prueba	Fracción de correctos (Entrada: Día. Referencia: Noche.)
VGG-16-Triplet-Fine-Tuned	0,15%
ResNet-50-Triplet	1,65%
ResNet-50-Concat	1,73%
ResNet-50-Triplet (Entrenando también en Alderley)	6,8%
ResNet-50-Concat (Entrenando también en Alderley)	7,82%

El conjunto Alderley fue desarrollado en [11]. Está formado por aproximadamente 15,000 imágenes del mismo recorrido tomadas de día y de noche. Es un conjunto muy complicado y presenta cambios de apariencia propios del cambio de iluminación del día a la noche como los que aparecen en la Figura 5.14izquierda. Se decide utilizar las últimas 4,600 imágenes para evaluar los resultados.

En la Tabla 5.3 se muestran los resultados de varias pruebas en el conjunto Alderley. En primer lugar se muestran los resultados de la red VGG-16-Triplet-Fine-Tuned. Se puede apreciar que los resultados son mucho peores que con el conjunto Nordland y apenas llegan al 0,2%. Es normal que los resultados empeoren puesto que se trata de un conjunto con cambios de apariencia muy diferentes de los vistos en el entrenamiento. La red VGG-16-Triplet-Fine-Tuned, debido al ajuste fino y al mencionado sobreajuste, funciona todavía peor que las demás estrategias. Además, se vuelve a demostrar que la red ResNet-50 es un mejor punto de partida que la red VGG-16.

Las dos pruebas siguientes de la tabla son los resultados de la red triplet entrenada desde la red ResNet-50 (ResNet-50-Triplet) y la mejor estrategia de este trabajo, la concatenación de descriptores sin entrenamiento (ResNet-50-Concat). Los resultados son peores que los del conjunto Nordland y la información temporal apenas ayuda a mejorar de un 1,65% a un 1,73%.

Las últimas dos pruebas consistieron en entrenar la red triplet convolucional partiendo de la red ResNet-50, esta vez utilizando datos del conjunto Nordland y el conjunto Alderley. Se aprecia que los resultados mejoran de un 1,65% para la red normal a un 6,8% cuando se entrena con los datos del conjunto Alderley también (ResNet-50-Triplet (Entrenando también en Alderley)). Finalmente, se aprecia que concatenar descriptores de esta nueva red (ResNet-50-Concat (Entrenando también en Alderley)) si que consigue una mejora mayor, del 6,8% de la nueva red normal al 7,82% de la concatenación de descriptores.

Es difícil comparar los resultados en este conjunto de datos con otras aproximaciones puesto que en el reconocimiento de lugares no hay un consenso en cuanto a las métricas utilizadas.

El conjunto Santa Lucía fue desarrollado por [35]. Se compone de diez vídeos de veinte minutos del mismo trayecto en coche por un barrio de la ciudad australiana de Brisbane. Cada trayecto se realizó a una hora distinta. Los lugares presentan cambios de iluminación, punto de vista y coches en movimiento. Se pueden ver los cambios de apariencia en la Figura 5.14derecha. En este trabajo se ha extraído una imagen por segundo de cada vídeo (unas 1,200 de cada vídeo) para evaluar los resultados.

El problema de este conjunto es que el reconocedor de lugares funciona comparando un conjunto de imágenes de referencia con uno de entrada. En el conjunto Nordland, existen diversas combinaciones como invierno frente a verano, primavera frente a otoño, etc. El conjunto Alderley solo tiene el día frente a la noche y la noche frente al día. El conjunto Santa Lucía tiene un número mayor de combinaciones a estudiar como el primer vídeo frente al último, el segundo frente al octavo, etc., lo que hace más complicado dar cifras exactas. En este trabajo se han estudiado 30 combinaciones elegidas de forma aleatoria y los resultados obtenidos se utilizan para calcular la media. Todas las estrategias de este trabajo alcanzan entre un 40% y un 50% de lugares correctos.

Los resultados con estos conjuntos demuestran que las redes son capaces de adaptarse a cambios de apariencia diferentes de los vistos durante su entrenamiento, siempre y cuando los cambios no sean demasiado drásticos. Es por ello relevante que el conjunto de datos de entrenamiento contenga cambios de apariencia muy diversos. A pesar de ello, a día de hoy no existen conjuntos de imágenes suficientemente grandes y es necesario que la comunidad dedicada al estudio del reconocimiento de lugares solucione este problema para mejorar los algoritmos.

5.9. Tiempos de ejecución y recursos computacionales

En la Tabla 5.4 se muestran los tiempos de ejecución de las estrategias propuestas en este trabajo. El sistema utilizado para las pruebas cuenta con un procesador Intel Core i5-6400 y las redes neuronales se han ejecutado en una tarjeta gráfica GeForce GTX 1060 (6GB). Es importante mencionar que optimizar el algoritmo no se encuentra dentro de los objetivos de este trabajo. Los tiempos dependen de la implementación y no se ha utilizado ningún tipo de computación paralela. Los tiempos de ejecución se dividen en dos categorías:

- **Tiempo de extracción de descriptores:** Es el tiempo que le cuesta a cada una de las redes neuronales procesar las imágenes y extraer los descriptores. En la Tabla 5.4 se muestra el tiempo de extracción de 1000 descriptores. Se puede apreciar que la estrategia ResNet-50-Triplet y ResNet-50-Concat son las más rápidas. Esto tiene sentido, pues en las dos se extraen los descriptores individualmente, lo cual cuesta menos que procesar varias imágenes. En el caso de la técnica ResNet-50-Concat se extraen y después se concatenan, por eso es algo más lenta. La más lenta de todas es la opción ResNet-50-Triplet-LSTM, ya que hay que procesar el descriptor de cada instante y evolucionar el estado interno de la red LSTM.

Tabla 5.4: Tiempos de ejecución de las distintas estrategias.

Prueba	Tiempo extracción 1000 descriptores (s)	Tiempo búsqueda del más cercano para 1000 descriptores (s)
ResNet-50-Triplet	14	2,9
ResNet-50-Concat	15	12
ResNet-50-Concat-Fused	17	2,9
ResNet-50-Triplet-LSTM	22	2,9

- **Tiempo de búsqueda del más cercano:** Es el tiempo que le cuesta al algoritmo comparar el descriptor de entrada con todos los de referencia y encontrar el más cercano según la distancia euclídea. En este caso se utiliza el conjunto de test Nordland y se compara cada uno de los descriptores de entrada con los 3,450 descriptores de referencia. Se muestra el tiempo de búsqueda de los mil primeros descriptores de entrada. Se aprecia que la técnica técnica ResNet-50-Concat es la más lenta y el resto tardan lo mismo. Esto tiene sentido, pues se concatenan tres descriptores de entrada y se comparan con los descriptores de referencia también concatenados. Al ser tres veces más grandes que en el resto de estrategias, la comparación tarda más. En el resto de estrategias los descriptores tienen el mismo tamaño, cambia la forma de extraerlos pero se tarda lo mismo en compararlos.

En cuanto a tiempos de entrenamiento de las redes neuronales, las arquitecturas de cada estrategia se han entrenado entre 20 y 30 horas.

Como se ha mencionado previamente, el descriptor individual de las distintas estrategias de este trabajo tiene una dimensión de 128 y cada valor se codifica con 4 bytes. Cada descriptor ocupa por tanto $128 \times 4 = 512$ bytes. Una base de datos formada por los descriptores extraídos ocuparía 500 Kibibytes por cada 1,000 imágenes. En el caso del conjunto de 3,450 imágenes de test usado en este trabajo la base de datos ocupa 1,68 Mebibytes.

(a)

(b)

(c)

Figura 5.13: Ejemplos de lugares reconocidos correctamente solo al usar secuencias. **a, b, y c:** Se indican en rojo las imágenes que el reconocedor basado en una vista considera que son el mismo lugar a pesar de ser lugares a cientos de kilómetros. Se indican en verde los lugares que se reconocen correctamente gracias a utilizar secuencias. Los lugares a la izquierda se muestran en verano y a la derecha en invierno.

Figura 5.14: Ejemplos de cambios de apariencia que aparecen en los conjuntos Alderley y Santa Lucía. **Izquierda:** Mismos dos lugares del conjunto Alderley de día y de noche. **Derecha:** Mismos dos lugares del conjunto Santa Lucía a dos horas distintas del día.

6. Conclusiones

En este trabajo de fin de Máster se ha planteado el uso de información temporal para mejorar la precisión del reconocimiento de lugares ante cambios drásticos de apariencia. Las estrategias estudiadas utilizan una red neuronal capaz de procesar las imágenes para extraer descriptores. La distancia euclídea entre descriptores mide la similitud de los lugares y sirve para reconocerlos. En lugar de comparar descriptores individualmente como hacen las aproximaciones típicas del estado del arte, en este trabajo se ha estudiado la posibilidad de comparar la información de secuencias de imágenes.

Se ha demostrado que redes pre-entrenadas más recientes, como la red ResNet-50, pueden funcionar mejor como punto de partida frente a redes más clásicas como la red VGG-16. En cuanto a las estrategias que utilizan información temporal, se estudió inicialmente la concatenación de varios descriptores. Posteriormente, se planteó el entrenamiento de redes neuronales para procesar varias imágenes a la vez. Finalmente, se entrenaron redes recurrentes para procesar las imágenes secuencialmente. Se ha comprobado que concatenar los descriptores proporciona los mejores resultados cuando las secuencias comparadas no presentan variaciones temporales significativas, como trayectorias en sentido inverso. Las redes recurrentes de tipo LSTM, no obtienen los mejores resultados. Esto demuestra la complejidad de las redes LSTM, especialmente en su entrenamiento. Como trabajo futuro es posible estudiar modificaciones de las redes recurrentes para mejorar su precisión. No obstante, la literatura más reciente no es concluyente acerca de la conveniencia de usar redes LSTM para datos secuenciales, por lo que la mejora no es segura. A pesar de todo, se ha demostrado que todas las estrategias estudiadas mejoran con respecto a los resultados obtenidos cuando no se tiene en cuenta la información temporal.

Finalmente, se ha demostrado que el reconocedor obtenido mejora los resultados del estado del arte en el conjunto Nordland. También se ha probado que los resultados se pueden replicar en conjuntos de datos distintos a los de entrenamiento, como el conjunto Santa Lucía. Cuando los cambios de apariencia son más drásticos, como en el conjunto Alderley, los resultados empeoran y se hace necesario introducir los cambios de apariencia en el entrenamiento de las redes. También es trabajo futuro la creación de un conjunto de datos con cambios de apariencia más diversos que los existentes actualmente. El método propuesto es capaz de reconocer correctamente más de un 92% de lugares en tramos de más de 80 kilómetros de recorrido cuando las condiciones son adversas.

Bibliografía

[1] E. Garcia-Fidalgo and A. Ortiz, "Hierarchical place recognition for topological mapping," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1061–1074, 2017.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[3] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. Furgale, "The gist of maps-summarizing experience for lifelong localization," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 2767–2773, IEEE, 2015.

[4] 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.

[5] 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.

[6] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5297–5307, 2016.

[7] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, and E. Romera, "Fusion and binarization of CNN features for robust topological localization across seasons," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 4656–4663, IEEE, 2016.

[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] 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.

[10] 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.

[11] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 1643–1649, IEEE, 2012.

[12] D. Olid Hernández, J. Civera Sancho, and J. M. Fácil Ledesma, "Reconocimiento de lugares con invarianza a cambios del entorno mediante redes neuronales," 2017.

[13] D. Olid, J. M. Fácil, and J. Civera, "Single-View Place Recognition under Seasonal Changes," *arXiv preprint arXiv:1808.06516*, 2018.

[14] P. Martinet, "10th Workshop on Planning, Perception and Navigation for Intelligent Vehicles at IROS 2018," 2018. [Online; web accedida el 11-Septiembre-2018].

[15] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[16] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. Software available from tensorflow.org.

[17] 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.

[18] 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.

[19] G. Alder and D. Benson, "Draw.io. Easy to use diagramming tool with extras for power users." [Online; web accedida el 11-Septiembre-2018].

[20] 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.

[21] B. M. del Brío y David Buldain Pérez, "Redes neuronales electrónicas: Tecnologías para machine learning. apuntes de la asignatura.," Sept 2017.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[23] 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.

[24] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] N. B. Corporation, "Nordlandsbanen: minute by minute, season by season," 2012. [Online; web accedida el 19-Junio-2017].

[27] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, vol. abs/1409.1556, 2014.

[28] 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.

[29] 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.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[31] A. Canziani, A. Paszke, and E. Culurciello, "An Analysis of Deep Neural Network Models for Practical Applications," *CoRR*, vol. abs/1605.07678, 2016.

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

[33] 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.

[34] N. Sünderhauf, "open source matlab implementation of the original seqslam algorithm published by milford and wyeth," 2013. [Online; web accedida el 11-Septiembre-2018].

[35] A. Glover, W. Maddern, M. Milford, and G. Wyeth, "FAB-MAP + RatSLAM: Appearance-based SLAM for Multiple Times of Day," in *ICRA*, (Anchorage, USA), 2010.