Máster Universitario en Sistemas Inteligentes

Trabajo Final de Máster

Super-Resolución inter-sensor para el realce espacial de imágenes Sentinel

Autor:
Adrian Puertas Cabedo

Tutor:
Ramón Alberto Mollineda Cardenas
Co-Tutor:
Rubén Fernández Beltrán

Fecha de lectura: 29 de octubre de 2018

Curso académico 2017/2018

Índice de contenidos

Capítulo 1. Introducción

Este trabajo es un estudio detallado de las posibilidades del método SRCNN (Dong et al., 2016) [1], una red neuronal convolucional profunda, en la tarea de superresolución de imágenes inter-sensor. Se quiere experimentar sobre la implementación del método realizada en TensorFlow durante el trabajo de iniciación a la investigación con el fin de optimizarla aplicando ciertas modificaciones. Además, se realiza una comparativa de este método frente a otros más tradicionales.

Este capítulo introductorio tiene como finalidad explicar el contexto del problema objeto de estudio y la motivación que nos ha llevado a elegirlo, así como definir los objetivos que se esperan de este trabajo.

1.1 Contexto y motivación

El programa Copernicus es una iniciativa conjunta de la Comisión Europea, la Agencia Espacial Europea y la Agencia Europea del Medio Ambiente con el fin de proporcionar información operacional de nuestro planeta capturada desde el espacio y que resulte de utilidad para múltiples aplicaciones de seguridad y medio ambiente. En este contexto, se han planificado cinco diferentes misiones de observación de la Tierra, llamadas Sentinel, para garantizar dicha provisión operacional de datos. Entre todos los recursos del programa Copernicus, las misiones Sentinel-2 (S2) y Sentinel-3 (S3) comparten una sinergia especial porque ambas se centran en servicios de monitoreo global sobre superficies terrestres utilizando para este fin imágenes multiespectrales de media y alta resolución. Más específicamente, S2 es una misión en órbita polar que comprende dos satélites idénticos, S2A que se lanzó el 23 de junio de 2015 y S2B que lo siguió el 7 de marzo de 2017. Cada satélite incorpora un instrumento multiespectral (MSI) que proporciona un conjunto versátil de 13 bandas espectrales que van desde el infrarrojo visible y cercano (VNIR) hasta el infrarrojo de onda corta (SWIR). En particular, cuatro de estas bandas (B02-B04, B08) se adquieren a una resolución espacial de 10 mpp (metros por píxel), seis bandas (B05-B07, B08A, B11, B12) a 20 mpp y las tres bandas restantes (B01, B09, B10) a 60 mpp. De forma análoga, la misión S3 incluye un par de satélites, llamados S3A y S3B, donde el primero se lanzó el 16 de febrero de 2016 y el segundo se lanzó con éxito el 25 de abril de 2018. Ambos satélites transportan el instrumento “Ocean and Land Color Instrument” (OLCI), que proporciona 21 bandas (Oa01-Oa21) que abarcan desde los 403 a los 1040 nanómetros en el rango espectral VNIR con anchos de banda de 3,75 a 40 nanómetros. En cuanto a la resolución espacial del sensor, OLCI tiene un requisito de resolución global de 300 mpp para las zonas costeras y los productos terrestres.

Aunque las misiones S2 y S3 se diseñaron para proporcionar productos de datos globales de vegetación, suelo y agua (vías navegables interiores y áreas costeras), las diferencias espectrales y espaciales entre los sensores MSI y OLCI hacen que cada satélite sea más adecuado para un campo de aplicación particular. Mientras que la mayor resolución espacial de S2 permite el uso de sus productos para tareas de caracterización con el requisito de un alto nivel de detalle espacial, como el mapeo del suelo o la clasificación del

uso del suelo, S3 puede proporcionar una estimación más precisa de parámetros biofísicos mediante su mayor sensibilidad espectral en la longitud de onda VNIR. Específicamente, uno de los descriptores biofísicos más importantes que pueden ser estimados a partir de datos de teledetección es la cobertura vegetal la que nos proporciona información valiosa sobre el estado de salud de la Tierra. En general, los índices de vegetación más populares, como NDVI (índice de vegetación diferencial normalizado) o SAVI (índice de vegetación ajustado al suelo), explotan la correlación entre la longitud de onda de absorción máxima de clorofila y el infrarrojo cercano, ya que la vegetación tiende a reflejar de forma natural longitudes de onda más largas a medida que el contenido de clorofila aumenta. Como resultado, la mayor resolución espectral VNIR del sensor OLCI aumenta la sensibilidad del instrumento más allá del nivel máximo de la clorofila, lo que finalmente conduce a una mejor estimación de la vegetación cuando se usan datos S3. No obstante, la principal desventaja de las imágenes capturadas mediante el sensor OLCI de S3 consiste en su limitada resolución espacial, puesto que la mayor sensibilidad espectral del instrumento genera una serie de limitaciones físicas en el proceso de captura que condicionan la capacidad del sensor para discernir diferentes elementos estructurales sobre la superficie terrestre. De forma opuesta, el sensor MSI de S2 es capaz de alcanzar una resolución espacial sensiblemente mayor pero utilizando una sensibilidad espectral bastante menos precisa.

Teniendo en cuenta todas estas consideraciones, el presente trabajo está enfocado en el realce espacial de datos S3, capturados mediante el instrumento OLCI, tomando como referencia la información de alta resolución proporcionada por el sensor MSI de S2. Es decir, el objetivo principal del trabajo se basa en explotar la sinergia existente entre las misiones S2 y S3 para mejorar la resolución espacial de productos S3 mediante el uso de técnicas de realce espacial o superresolución. Típicamente este tipo de técnicas han sido usadas de forma específica para un sensor concreto [5] o incluso utilizando múltiples instrumentos de captura simultánea [6]. No obstante, en este trabajo perseguimos un objetivo más general donde se busca superresolver imágenes del instrumento OLCI de S3 utilizando como información auxiliar de referencia los datos capturados por el sensor de otro satélite diferente, es decir, MSI de S2.

Para ello, primero definimos un arquitectura de superresolución basada en redes neuronales convolucionales capaz de efectuar el proceso de realce espacial tomando en consideración datos imagen intersensor. Posteriormente, consideramos una base de datos de imágenes multi-espectrales que consta de diez productos S2/S3 procesados que pertenecen a cinco regiones europeas diferentes: Sierra de Andújar (España), Madrid (España), Burdeos (Francia), Milán (Italia) y Utrech (Holanda). Finalmente, llevamos a cabo una comparativa experimental utilizando varios algoritmos de superresolución disponibles en la literatura con objeto de estudiar la efectividad de la propuesta así como analizar los límites de superresolución existentes en S3.

1.2 Objetivos

Los principales objetivos que este trabajo persigue son:

- Aplicar los conocimientos aprendidos y el método implementado en el trabajo de iniciación a la investigación en un problema diferente.

- Estudiar el potencial de la superresolución en imágenes multiespectrales.

- Explotar la sinergia existente entre las diferentes misiones Sentinel, dentro del contexto del programa Copernicus, para refinar los productos actualmente ofrecidos por la ESA.

- Mejorar la resolución espacial de las imágenes OLCI de S3 mediante el uso de información auxiliar de referencia del sensor MSI de S2.

- Analizar los límites de las técnicas de superresolución para el realce espacial en S3.

- Contrastar los resultados obtenidos con métodos convencionales de superresolución.

Además se persiguen los siguientes objetivos formativos:

- Adquirir nociones sobre metodologías de investigación básica y aplicada.

- Ser capaz de integrarse en una dinámica de trabajo en equipo.

- Estudiar autónomamente una fuente de información, y explicar su contenido.

- Ser capaz de redactar documentos científico-técnicos.

1.3 Estructura de la memoria

Esta memoria se estructura en cinco capítulos, de la siguiente manera:

- Capítulo 1. Se explica el contexto del problema y la motivación, así como los objetivos, del trabajo llevado a cabo.

- Capítulo 2. Se realiza una introducción general de las redes neuronales convolucionales.

- Capítulo 3. Se analiza el método objeto de estudio, sus fundamentos, su arquitectura, los hiperparámetros, y las fases de entrenamiento y test.

- Capítulo 4. Se detallan los conjuntos de datos utilizados, el preprocesamiento llevado a cabo y los experimentos realizados. También se realiza una comparación con otros métodos.

- Capítulo 5. Se aportan las conclusiones obtenidas tras la realización de todo el trabajo.

Capítulo 2. Redes Neuronales Convolucionales

Este capítulo tiene como objetivo introducir de manera general las redes neuronales convolucionales dado que estas son la base del método objeto de estudio que se presenta en el próximo capítulo.

2.1 Introducción

Las redes neuronales convolucionales son un tipo de redes neuronales artificiales profundas que se utilizan principalmente para clasificar imágenes, agruparlas por similitud e incluso reconocer objetos dentro de escenas. Como se muestra en la figura 2.1, estas redes pueden llegar a reconocer todo tipo de objetos como, por ejemplo, rostros, personas, animales, vehículos, señales de tráfico, tumores, etc. Esto es una de la principales razones por las que las redes convolucionales y el aprendizaje profundo han captado la atención de tanta gente, ya que están impulsando importantes avances en aplicaciones de visión por computador en coches autónomos, robótica, drones, diagnósticos médicos, etc.

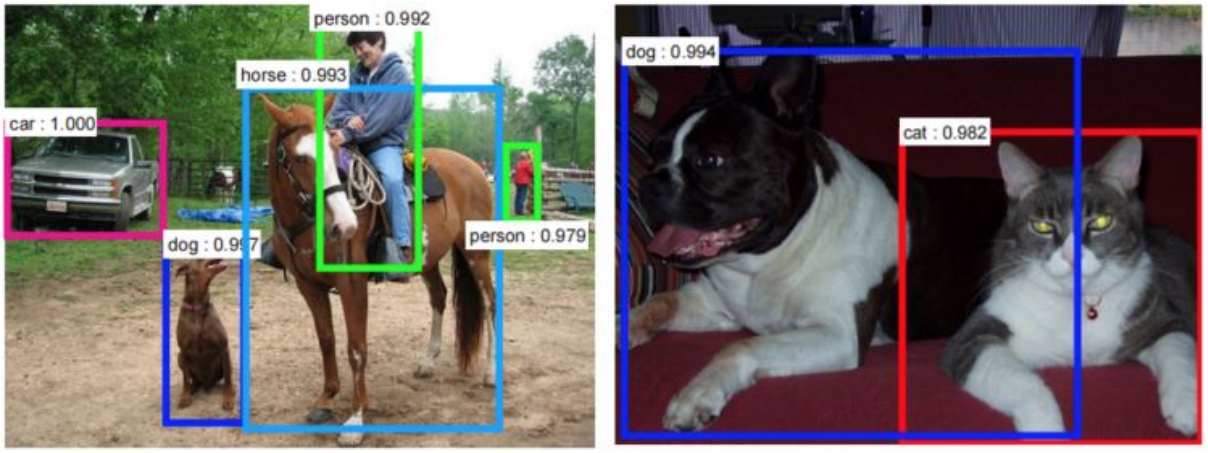

Figura 2.1. Reconocimiento de objetos en imágenes. Fuente[1]

Otra aplicación muy popular es el reconocimiento óptico de caracteres (OCR), con el fin de digitalizar textos y hacer posible el procesamiento del lenguaje natural en documentos analógicos y escritos a mano. E incluso, sería posible aplicar las redes convolucionales al análisis de sonido cuando este se representa visualmente como un espectrograma. En nuestro caso, se va a utilizar una red neuronal convolucional para *superresolver* imágenes, es decir, conseguir imágenes de alta resolución a partir de imágenes de baja resolución.

[1] https://arxiv.org/pdf/1506.01497v3.pdf

2.2 Convolución

Las redes neuronales convolucionales utilizan **filtros**, también conocidos como núcleos o *kernels*, con los que realizan la operación de convolución sobre la imagen para detectar **características** como, por ejemplo, bordes, texturas, etc. La operación de convolución se realiza moviendo el filtro sobre regiones de la imagen del mismo tamaño que el filtro, y consiste en calcular la suma del producto elemento a elemento entre ambas estructuras, como se muestra en la figura 2.2. Como resultado, se obtiene un valor numérico que indica la probabilidad de que la característica que representa dicho filtro esté presente en la región de la imagen que está siendo analizada.

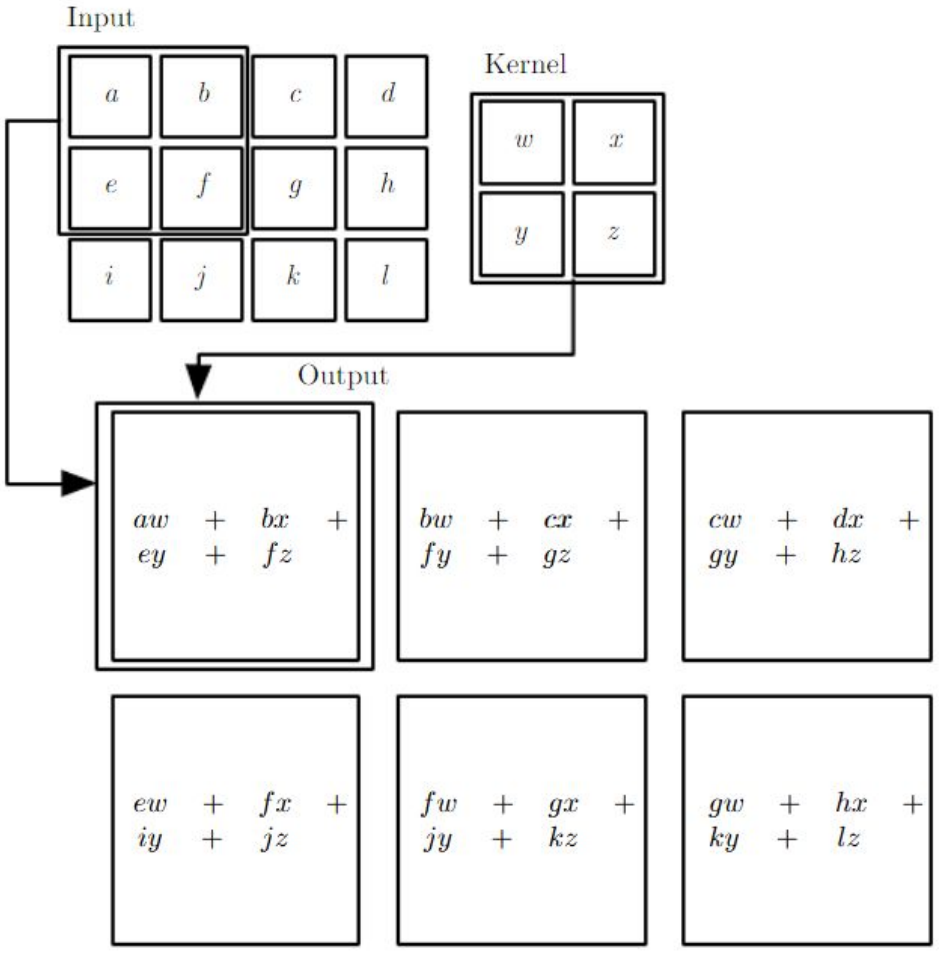

Figura 2.2. Ejemplo de convolución 2D. La salida está restringida a las posiciones donde el filtro puede superponerse completamente. La estrategia mostrada en la figura (*válida*) no añade elementos en los bordes para mantener el tamaño original en la salida. Fuente[2]

El tamaño de la salida tras la operación de convolución puede variar según el parámetro de relleno o *padding*, que puede optar por dos estrategias: *válida* o *equivalente*. La estrategia *válida* restringe la salida a regiones donde el filtro puede superponerse completamente, lo que provoca que la salida sea más pequeña al perder los bordes según el tamaño del filtro.

[2] https://www.deeplearningbook.org/contents/convnets.html

La estrategia *equivalente* hace que la salida sea de idéntico tamaño a la entrada. Para evitar la pérdida de los bordes, añade bordes artificiales a la estructura de entrada para que el filtro pueda superponerse completamente en todas las posiciones de la entrada original.

El desplazamiento del filtro sobre la imágen se realiza ordenadamente, en ambos ejes, con un parámetro de paso o *stride*. Normalmente el paso suele ser de una posición, por lo que el filtro operaría con la región determinada por cada elemento (según la estrategia de relleno). En caso de usar un paso más grande, el filtro no pasaría por todos los elementos, por lo que el tamaño de salida se vería reducido.

2.3 Función de activación

La función de activación produce una transformación no lineal al resultado de aplicar un filtro a una región de la imagen. Más formalmente, la salida de la operación de convolución se suma con un término de polarización (*bias*), y este resultado se convierte en argumento de la función de activación no lineal.

El propósito de la función de activación es introducir no linealidad en el funcionamiento de la red, ampliando su capacidad para resolver problemas. Existen diferentes funciones de activación tradicionales como la sigmoide o la tangente hiperbólica, aunque una de las más populares es la función de activación ReLU (unidad lineal rectificada), cuyo comportamiento se muestra en la figura 2.3. Esta función es usada en este trabajo.

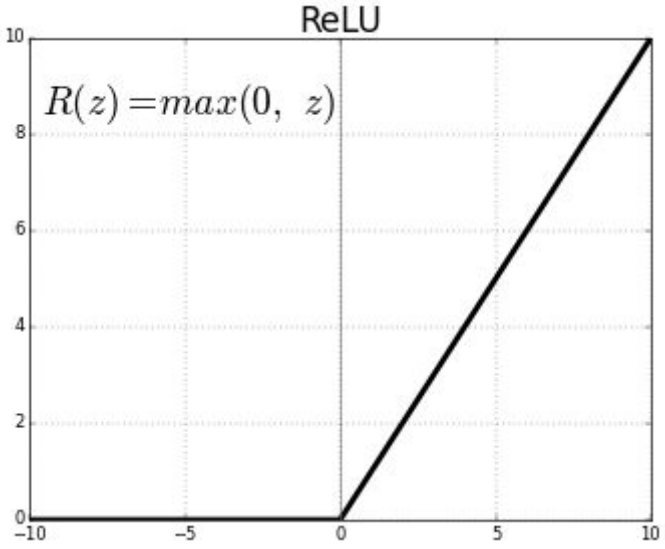

Figura 2.3. La función ReLU consiste simplemente en transformar a 0 los valores de entrada negativos, y mantener los valores positivos. Fuente[3]

[3] https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec

2.4 Submuestreo

Para acelerar el proceso de aprendizaje, reducir el consumo de memoria, e incrementar la capacidad de generalización del modelo, es común utilizar un operador de submuestreo, o *downsampling*, que contribuirá a reducir la redundancia presente en los datos y a hacer el modelo invariante a traslaciones locales. La operación más común para este menester es la elección del valor máximo como representante de una región, o *max pooling*, aunque existen otros como la media, mediana, o mínimo. En esta técnica, tal y como se observa en la figura 2.4, se recorren regiones de la imagen de acuerdo con un paso establecido y, en cada paso, el valor máximo dentro de la ventana se agrupa en una matriz de salida. Aunque esta es común en las redes convolucionales, este operador no se usa en la SRCNN.

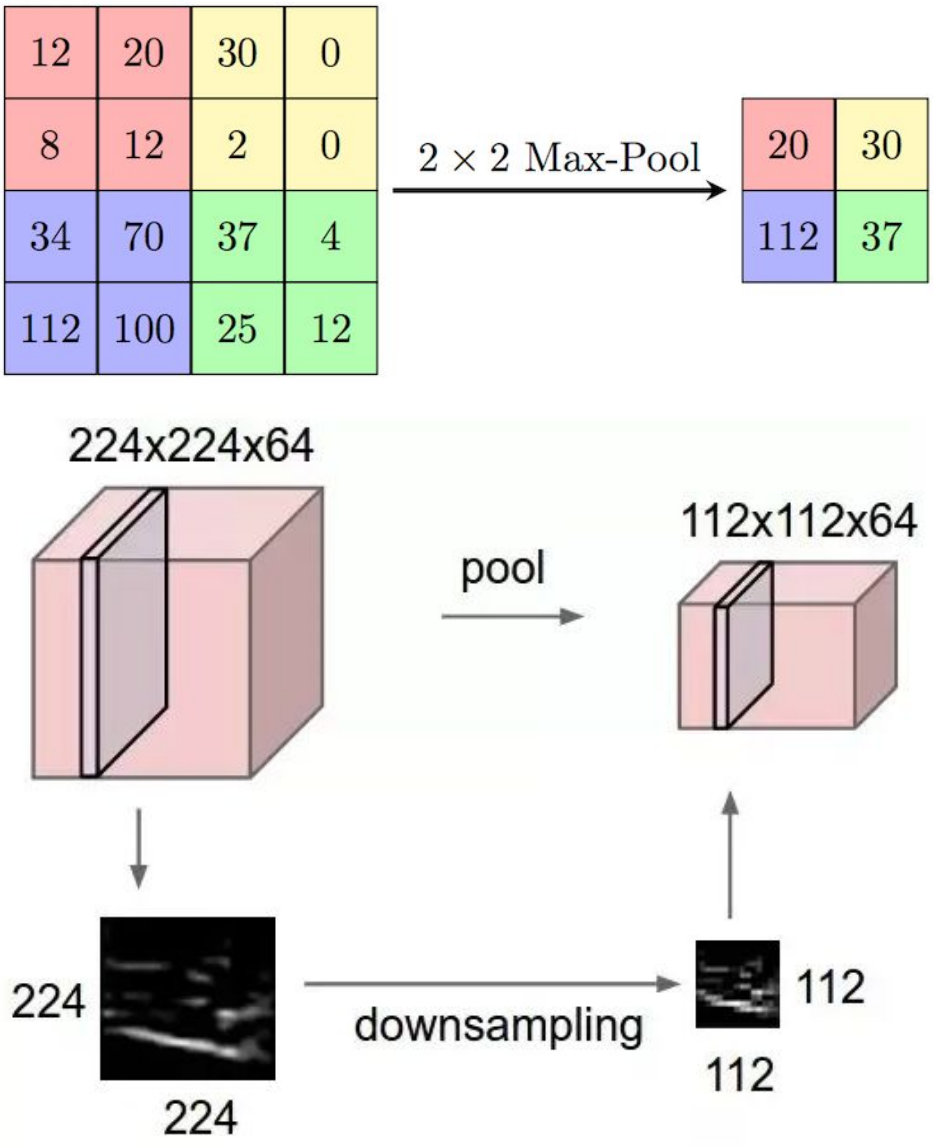

Figura 2.4. Representación visual y real del operador *max pooling*. Fuente[4]

[4] https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

2.5 Arquitectura típica

Una arquitectura típica de red neuronal convolucional suele alternar capas de convolución (con función de activación en su salida) con capas de submuestreo, con tantos niveles de profundidad como sea necesario. Finalmente, se puede terminar con una o más capas completamente conectadas si lo que se busca es clasificar las imágenes. En la figura 2.5 se muestra un representación de arquitectura típica para problemas de clasificación.

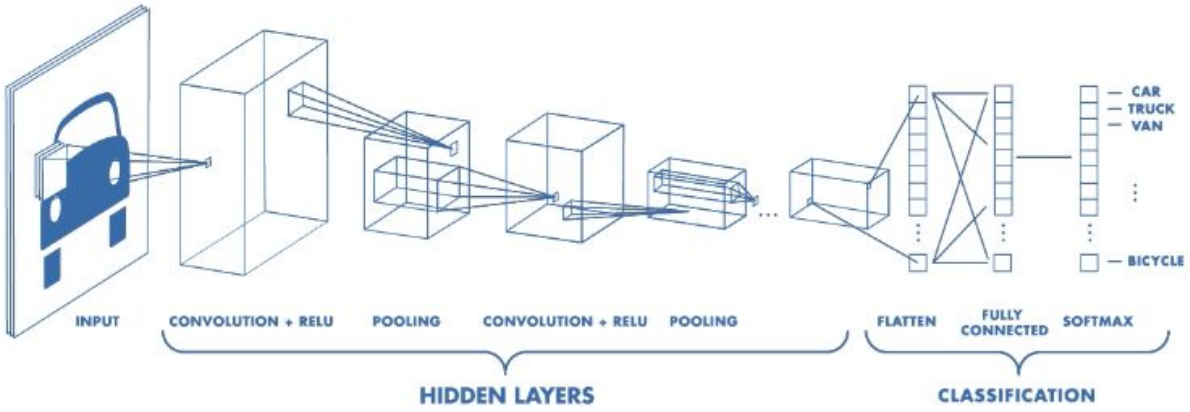

Figura 2.5. Arquitectura típica de red neuronal convolucional para clasificación de imágenes. Fuente[5]

[5] https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050

Capítulo 3. Red neuronal convolucional para superresolución (SRCNN)

Este capítulo tiene como objetivo presentar y analizar el método SRCNN. Comenzaremos presentando la arquitectura de la red a implementar y el objetivo de cada una de sus capas. Tras esto explicaremos los hiperparámetros así como el proceso de entrenamiento y test propuestos en el trabajo original.

3.1 Arquitectura de la SRCNN

En esta sección se introduce la arquitectura básica de la SRCNN, tal y como aparece en el trabajo de Dong et al. No obstante, antes de comenzar con los detalles de la arquitectura, se describe el protocolo típico de evaluación de soluciones de superresolución.

La red toma como entrada una imagen de baja resolución, y genera una versión de alta resolución de idéntico tamaño. La imagen de entrada la denotaremos como Y, mientras que la imagen salida de la red, como $F(Y)$, donde F representa la función de transformación implementada por la red. El objetivo del proceso de aprendizaje (optimización de parámetros) es lograr imágenes estimadas $F(Y)$ tan parecidas como sea posible a las imágenes *ground truth* de alta resolución X.

Se quiere aprender la función de mapeado F que conceptualmente debe recrear tres operaciones que suelen realizar los métodos convencionales:

1. **Extracción y representación de parches.** Esta operación extrae parches superpuestos de la imagen de baja resolución Y, y representa cada parche como un vector de alta resolución. Estos vectores comprenden un conjunto de mapas de características, cuyo número es igual al número de operadores de representación (por ejemplo, filtros).

2. **Mapeado no lineal.** Esta operación mapea de forma no lineal cada vector de alta resolución en otro vector de alta resolución. Cada vector mapeado es conceptualmente la representación de un parche de alta resolución.

3. **Reconstrucción.** Esta operación agrega las representaciones de los parches de alta resolución anteriores para generar la imagen final de alta resolución. Se espera que esta imagen sea similar a la imágenes *ground truth* X.

Estas operaciones pueden ser implementadas en forma de red neuronal convolucional, como se describe en los siguientes subapartados. Una representación conceptual de la red se muestra en la figura 3.1.

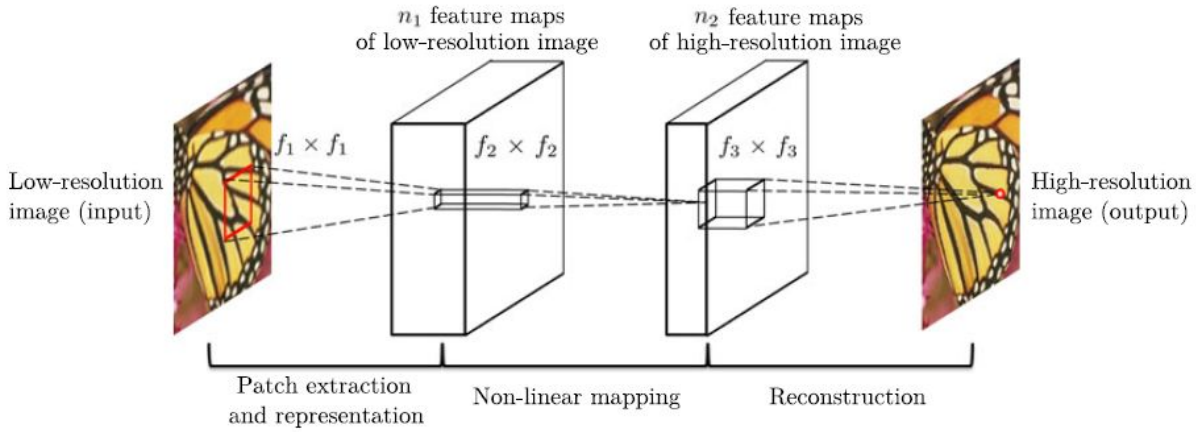

Figura 3.1. Dada una imagen Y de baja resolución, la primera capa convolucional de la red extrae un conjunto de mapas de características. La segunda capa aplica una transformación no lineal a estos mapas de características a un nuevo espacio creando las representaciones de alta resolución de los parches. La última capa combina las predicciones ubicadas en regiones locales para producir la imagen final de alta resolución $F(Y)$.

3.1.1 Extracción y representación de parches

Una estrategia popular en la restauración de imágenes es extraer parches y representarlos mediante un conjunto de bases preparadas previamente como PCA, DCT, Haar, etc. Esto equivale a convolucionar la imagen mediante un conjunto de filtros, cada uno de los cuales es una base.

Formalmente, la primera capa de la red es expresada como una operación F_1:

$$F_1(Y) = max(0,\ W_1 \ * Y + B_1)$$

donde W_1 y B_1 representan los filtros y los bias, respectivamente, y $*$ representa la operación de convolución. W_1 corresponde a n_1 filtros de tamaño $c \times f_1 \times f_1$, donde c es el número de canales de la imagen de entrada, y f_1 es el tamaño espacial del filtro. Intuitivamente W_1 aplica n_1 convoluciones a la imagen, y cada convolución tiene un kernel de tamaño $c \times f_1 \times f_1$. La salida se compone de n_1 mapas de características. B_1 es un vector n_1-dimensional, del que cada elemento está asociado a un filtro. Se aplica la función de activación ReLU ($max(0, x)$) para rectificar los resultados de los filtros.

3.1.2 Mapeado no lineal

La primera capa extrae un vector de características n_1-dimensional para cada parche. En la segunda operación, mapeamos cada uno de estos vectores n_1-dimensionales en un vector n_2-dimensional. Esto es equivalente a aplicar n_2 filtros con soporte espacial trivial de $n_1 \times 1 \times 1$, que en la práctica consiste en una transformación no lineal del espacio de representación n_1-dimensional a otro de dimensión n_2. Esta interpretación sólo es válida para filtros de 1×1, aunque la operación es fácil de generalizar a filtros más grandes como

3×3 o 5×5. En este caso, la transformación no lineal no se realiza sobre la imagen de entrada, sino sobre parches 3D del mapa de características generado por una capa anterior.

Formalmente, la segunda capa de la red es expresada como una operación F_2:

$$F_2(Y) = max(0,\ W_2\ * F_1(Y) + B_2)$$

donde W_2 contiene n_2 filtros de tamaño $n_1 \times f_2 \times f_2$, y B_2 es n_2-dimensional. Cada uno de los vectores n_2-dimensionales de salida de la transformación no lineal es conceptualmente una representación de un parche de alta resolución que se usará para la reconstrucción.

Es posible agregar más capas convolucionales para aumentar la no linealidad. Pero esto puede aumentar la complejidad del modelo en $n_{i-1} \times f_i \times f_i \times n_i$ parámetros por cada nueva capa y, por lo tanto, exige más recursos de entrenamiento (tiempo, capacidad de cómputo, muestras de entrenamiento, etc.).

3.1.3 Reconstrucción

En los métodos tradicionales, los parches superpuestos de alta resolución predecidos a menudo se promedian para producir la imagen final completa. El promediado se puede considerar como un filtro predefinido en un conjunto de mapas de características (donde cada posición es la forma vectorial "aplanada" de un parche de alta resolución). La ventaja de las redes neuronales profundas de extremo a extremo es que son capaces de aprender este filtro, adaptándolo a la naturaleza de los datos, en lugar de usar uno genérico.

Así pues, expresamos formalmente la última capa de la red como la salida de la misma:

$$F(Y) = W_3\ * F_2(Y) + B_3$$

donde W_3 corresponde a c filtros de tamaño $n_2 \times f_3 \times f_3$ y B_3 es un vector c-dimensional. El parámetro c corresponde al número de canales de la imagen de entrada; así, el modelo es capaz de reconstruir una imagen de salida con la estructura esperada.

3.2 Hiperparámetros

Aunque la arquitectura de la red está completamente definida, esta debe ser configurada con diferentes valores de hiperparámetros que harán a cada configuración única.

La configuración propuesta en el trabajo que sirve de referencia a este proyecto es $f_1 = 9$, $f_2 = 1$, $f_3 = 5$, $n_1 = 64$ y $n_2 = 32$. Para elegirla, Dong et al. [1] realizaron experimentos variando el número de filtros, el tamaño de los filtros, e incluso la profundidad de la red.

3.2.1 Número de filtros

En general, el rendimiento de la red suele mejorar si aumentamos el ancho de la red, es decir, si aumentamos el número de filtros, lo que también aumentará el coste el aprendizaje y el tiempo de ejecución.

Tomando como punto de partida los valores por defecto $n_1 = 64$ y $n_2 = 32$, Dong et al. [1] experimentaron con más filtros ($n_1 = 128$ y $n_2 = 64$), y con menos ($n_1 = 32$ y $n_2 = 16$). Los mejores resultados, obtenidos tras 8×10^8 iteraciones, se muestran en la tabla 3.1.

Número de filtros	$n_1 = 128$ y $n_2 = 64$	$n_1 = 64$ y $n_2 = 32$	$n_1 = 32$ y $n_2 = 16$
PSNR	32.60	32.52	32.26
Tiempo (seg)	0.60	0.18	0.05

Tabla 3.1. Resultados de utilizar diferente número de filtros.

3.2.2 Tamaño de los filtros

Las dimensiones de los filtros determina el tamaño del área local (parches) que intervendrá en la extracción de características de cada píxel.

En el caso del trabajo de Dong et al. [1], los tamaños de los filtros de la solución base fueron 9, 1 y 5 en las capas 1, 2 y 3, respectivamente. El filtro de tamaño 1 en la segunda capa se trataría, como se ha comentado antes, de una transformación no lineal del espacio de representación de los mapas de características que resultan de la primera capa.

Primero experimentaron aumentando el tamaño de los filtros de la primera y última capa a 11 y 7 respectivamente, y esta configuración 11-1-7 dió un resultado de 32.57 dB, lo que es sensiblemente mejor que los 32.52 dB obtenidos con la configuración base 9-1-5. Esto demuestra que un tamaño de filtro razonablemente mayor podría captar información estructural más rica, lo que a su vez conduce a mejores resultados.

Luego experimentaron aumentando el tamaño de los filtros en la segunda capa a 3 y a 5. Las curvas de convergencia mostradas en la figura 3.2 muestran que usar un tamaño de filtro mayor contribuye a mejorar el rendimiento.

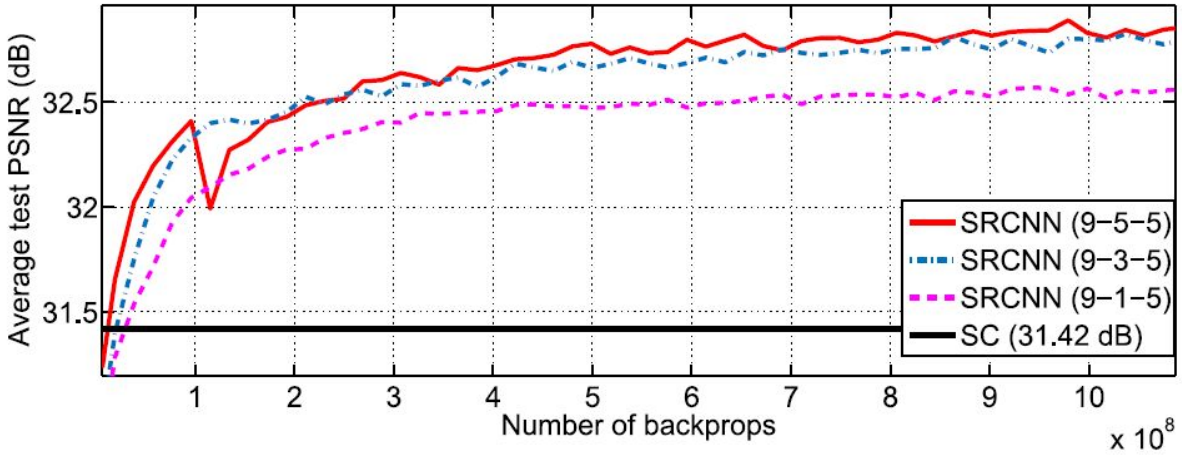

Figura 3.2. Un tamaño de los filtros mayor conduce a resultados mejores.

Sin embargo, la velocidad se ve reducida con un tamaño de filtros mayor. Por ejemplo, el número de parámetros de las configuraciones 9-1-5, 9-3-5 y 9-5-5 es 8.032, 24.416 y 57.184 respectivamente. La complejidad de 9-3-5 es más del doble y sin embargo la mejora de rendimiento es marginal. Por lo tanto, la elección del tamaño de los filtros siempre debe ser un compromiso entre el rendimiento y la velocidad.

3.2.3 Número de capas

La cantidad de capas determina la profundidad de la red. Un mayor número de capas permitiría explorar más opciones de identificación de patrones de alto nivel. No obstante, también implicaría un incremento significativo del coste del proceso de aprendizaje.

Además, durante la ejecución del algoritmo de propagación hacia atrás, un número elevado de capas podría causar el denominado *gradient vanishing problem* [4], en el cual el gradiente que se retropropaga podría hacerse muy pequeño llegando a impedir la actualización efectiva de los pesos.

En el caso del trabajo de Dong et al. [1], como se ha comentado antes, se propone usar 3 capas como solución base. Al contrastar esta con soluciones con un mayor número de capas, como se muestra en la figura 3.3, no se obtuvieron mejoras en el rendimiento y la convergencia fue más lenta. Estos resultados demuestran que, para superresolución, no siempre es mejor una red convolucional más profunda. Además, dado que esta red no contiene ninguna capa de *pooling* ni ninguna capa totalmente conectada, lo que la hace más sensible a la inicialización de los parámetros, resulta más difícil ajustar los parámetros apropiados para garantizar la convergencia cuando la red es más profunda.

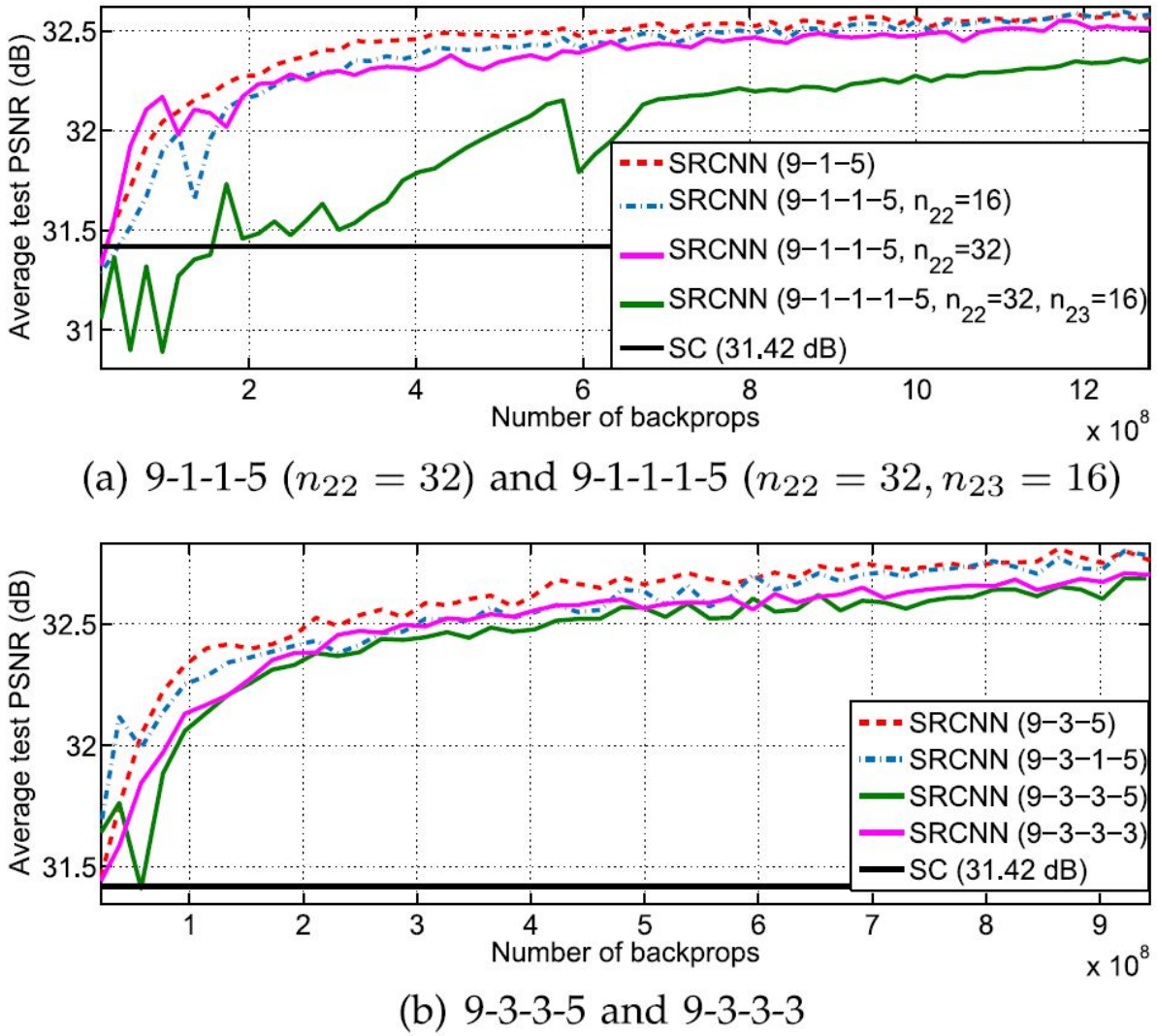

Figura 3.3. Más profundidad no siempre conduce a mejores resultados.

3.3 Entrenamiento

Aprender la función de mapeado F consiste básicamente en estimar los parámetros de la red $\theta = \{W_1, W_2, W_3, B_1, B_2, B_3\}$. Esto se consigue a través de la minimización, mediante el algoritmo *backpropagation* (o propagación hacia atrás), de una función de coste que calcula una medida de la diferencia entre las imágenes reconstruidas $F(Y;\theta)$ (salida de la red) y las imágenes X de alta resolución o *ground truth* correspondientes.

3.3.1 Función de coste (MSE)

Dado un conjunto de imágenes de alta resolución $\{X_i\}$ y sus correspondientes imágenes de baja resolución $\{Y_i\}$, utilizamos el error cuadrático medio (MSE) como la función de coste:

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left\| F(Y_i; \theta) - X_i \right\|^2$$

donde n es el número de muestras de entrenamiento.

3.3.2 Descenso de gradiente

El método de descenso de gradiente actualiza cada peso de la red en cada iteración de forma proporcional a la derivada parcial de la función de coste con respecto a dicho peso. Formalmente, esta función de actualización queda como sigue:

$$\Delta_{l,\, i+1} = 0.9 \cdot \Delta_i + \eta \cdot \frac{\delta L}{\delta W_i^l} \ , \ W_{i+1}^l = W_i^l + \Delta_{l,\, i+1}$$

donde $l \in \{1, 2, 3\}$ e i son los índices de las capas e iteraciones, η es el coeficiente de aprendizaje, y $\frac{\delta L}{\delta W_i^l}$ es la derivada parcial de L con respecto a W^l. La constante 0.9 es el *momentum* que multiplica el cambio aplicado en la iteración anterior, que no es parte natural del descenso por gradiente, pero que contribuye a la estabilidad del proceso de actualización de los parámetros.

Dong et al. [1] propone que los pesos de cada capa sean inicializados con una distribución Gaussiana de media 0 y desviación estándar 0.001, mientras que para la inicialización del bias propone simplemente 0. El ratio de aprendizaje será 10^{-4} para las primeras dos capas y 10^{-5} para la tercera. Han demostrado empíricamente que aplicar un ratio de aprendizaje más pequeño en la última capa es importante para que la red converja.

3.4 Test

Una vez entrenada, la red se evalúa a través de una función que nos permita medir el error de reconstrucción cometido por la red sobre imágenes independientes de baja resolución (imágenes de prueba). Para ello se comparan las imágenes resultantes (alta resolución) con las imágenes de referencia (*ground truth*). Además, este error se contrasta con el que se obtiene de comparar las imágenes de entrada (baja resolución) y las propias imágenes de referencia, pudiendo así evaluar de forma más objetiva el rendimiento de la red.

3.4.1 Función de error (PSNR)

La función de error utilizada es Proporción Máxima de Señal a Ruido o PSNR, que calcula la relación entre la energía máxima de una señal y el ruido que la afecta. Es habitual usar la función PSNR junto con el MSE pues, como se muestra abajo, la primera es es inversamente proporcional a la segunda.

El resultado de PSNR se mide en decibelios y suele expresarse en una escala logarítmica debido al gran rango dinámico que pueden tener las señales. Los valores típicos de PSNR están entre 30 y 50 dB, y a mayor PSNR, mejor resultado.

Típicamente el PSNR es utilizado para medir cuantitativamente la calidad de reconstrucción de imágenes. Debido a que PSNR depende del MSE, a continuación se reformula MSE para dos imágenes monocromas I y K de tamaño $M \times N$:

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} ||I(i,j) - K(i-j)||^2$$

Así, el PSNR se define como:

$$PSNR = 10 \cdot log_{10} \left(\frac{MAX_I^2}{MSE} \right) = 20 \cdot log_{10} \left(\frac{MAX_i}{\sqrt{MSE}} \right)$$

donde MAX_I denota el máximo valor que puede tomar un píxel en la imagen. Cuando éstos se representan usando B bits, $MAX_I = 2^B - 1$.

Capítulo 4. Experimentos

Este capítulo tiene como objetivo presentar los conjuntos de datos utilizados durante las fases de entrenamiento y test, así como el preprocesamiento previo de las imágenes utilizadas. Por último, se detallan los experimentos realizados y se realiza una discusión de sus resultados.

4.1 Conjunto de datos

Se ha utilizado un conjunto de datos obtenido a partir de un repositorio de la Agencia Espacial Europea[6], que consta de diez pares de imágenes multiespectrales alineadas de productos S2 y S3 que pertenecen a cinco regiones europeas diferentes: Sierra de Andújar (España), Burdeos (Francia), Madrid (España), Milán (Italia) y Utrech (Holanda). Concretamente, de cada ubicación, tenemos dos muestras tomadas en momentos diferentes entre 2016 y 2017. En cada pareja tenemos la versión S2 de alta resolución espacial (20 mpp) y dimensiones 5490x5490 con 13 bandas espectrales, y la versión S3 de baja resolución espacial (300 mpp) y dimensiones 366x366 con 21 bandas espectrales.

Cabe destacar que las diferentes bandas del sensor MSI de S2 han sido adecuadamente remuestreadas a 20 mpp para homogeneizar la resolución espacial y el procesamiento del conjunto. Adicionalmente, las imágenes multiespectrales originales han sido transformadas a imágenes de un solo canal, para facilitar el funcionamiento de la red. Para ello se ha seguido el método de reducción espectral basado en la metodología de sustitución de componentes descrita en [6], donde se utiliza la primera componente de la transformación PCA para agrupar la información espacial de las imágenes. Este conjunto de imágenes en PCA1, que se muestra en la figura 4.1, será nuestro conjunto de datos de partida.

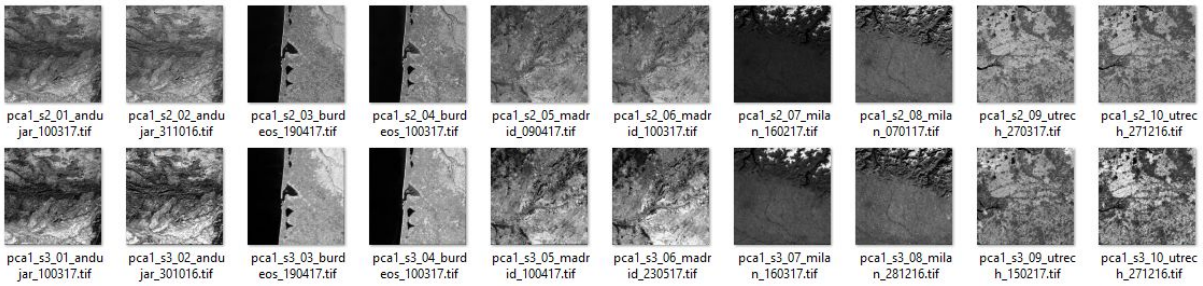

Figura 4.1. Miniaturas de las imágenes S2 (arriba) y S3 (abajo) proyectadas en la primera componente de la transformación PCA (conjunto de datos PCA1).

Se ha dividido el conjunto de diez parejas de muestras (S2, S3) dejando las muestras impares para entrenamiento, y las pares para test. Esto hace que en ambas fases tengamos un ejemplo de cada ubicación, aunque las imágenes de test no se corresponden con las de entrenamiento al existir diferencias temporales que afectan a factores como la vegetación, nieve, luz, etc.

[6] https://scihub.copernicus.eu/

4.2 Preprocesamiento

Previamente a su utilización, las imágenes deben ser preprocesadas para obtener tanto las muestras de entrada que precisa la red como las muestras de referencia que definen el objetivo a conseguir. El primer paso consiste en igualar los tamaños de las imágenes de entrada S3, más pequeñas y con menos resolución, y de las imágenes de salida S2. Para obtener las imágenes de entrada escalaremos las muestras de entrenamiento S3 por el factor de escala al que queramos entrenar la red (2x, 3x, 4x…) mediante interpolación bicúbica, y para obtener las imágenes de referencia reduciremos el tamaño de las muestras S2 al mismo tamaño resultante de escalar las muestras S3.

Como se ha mostrado en la literatura, el aprendizaje profundo generalmente se ve beneficiado al utilizar grandes cantidades de datos. Por ello, de cada par de imágenes de alta y baja resolución se extrae un conjunto de pares de subimágenes alineadas con la intención de multiplicar la cantidad de datos. Concretamente, se extraen subimágenes de 33x33 píxeles con cierto solapamiento determinado por un paso o *stride* de 14.

Por último, debido a que la red no define *padding*, en las operaciones de convolución, se produce una pequeña pérdida de tamaño en las imágenes de salida con respecto al de las imágenes de entrada. En particular, se pierden 12 pixeles por eje, 6 a cada lado, quedando las imágenes de salida con un tamaño de 21x21 píxeles. De esta forma, para comparar las imágenes de salida con las esperadas de alta resolución, debemos recortar estas últimas para adaptarlas al contenido y tamaño esperado, es decir, a 21x21.

4.3 Experimentos

En esta sección se explicarán los experimentos realizados y los resultados obtenidos.

4.3.1 Mejoras sobre el método original

El primer conjunto de experimentos consistió en variantes menores de la implementación original, con el fin de explorar más detalladamente su potencial.

Se parte de la configuración básica propuesta en el artículo de Dong et al., implementada en TensorFlow durante el Trabajo de Iniciación a la Investigación, a la cual nombramos v1. Esta versión inicial tiene un arquitectura de tres capas con 64 filtros de 9x9x1 píxeles en la primera capa, 32 de 1x1x64 en la segunda, y 1 filtro de 5x5x32 en la tercera. Tiene un coeficiente de aprendizaje base de 0.0001, el cuál se ve multiplicado por 0.1 en la última capa y utiliza un optimizador `MomentumOptimizer`[7] que aplica un *momentum* de 0.9 al método de descenso de gradiente. En la figura 4.2 se muestra que la evolución del PSNR medio obtenido sobre las imágenes de test (en rojo), con respecto a las imágenes de referencia o *ground truth*, es bastante superior al obtenido con las imágenes de entrada de baja resolución. La tendencia decreciente de la curva v1 parece deberse a que la red está sobreajustándose al conjunto de entrenamiento con el paso de las épocas, y por ello se ve resentida sobre el conjunto de test.

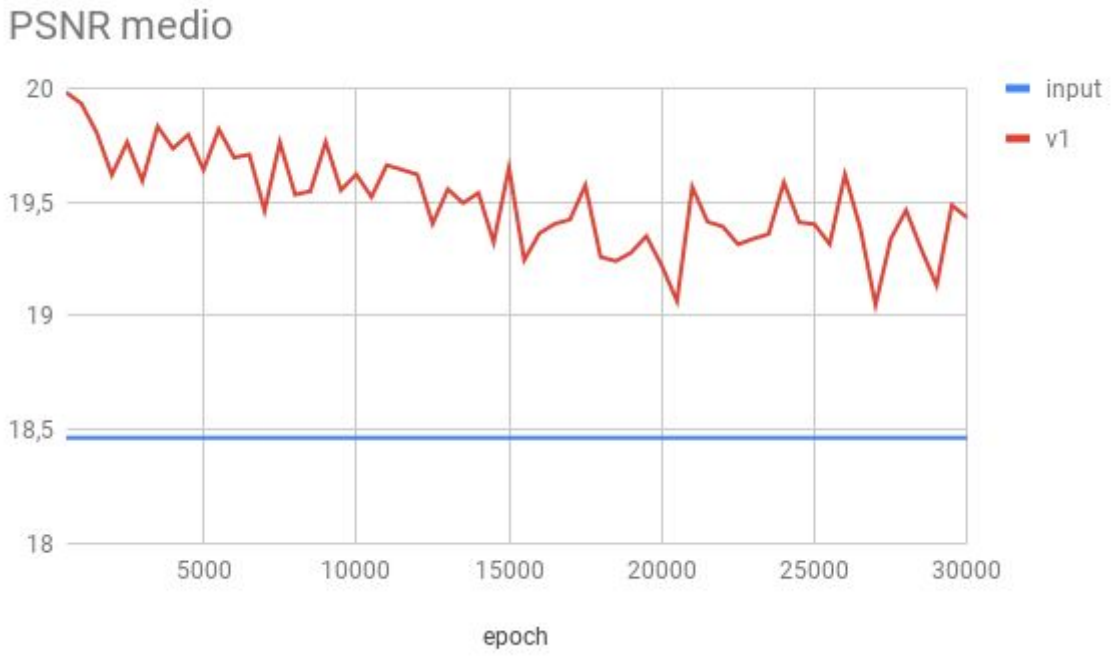

Figura 4.2. PSNR medio sobre el conjunto de test de la versión 1 frente a la interpolación bicúbica utilizada en la entrada.

[7] https://www.tensorflow.org/api_docs/python/tf/train/MomentumOptimizer

La versión 2 contiene la primera mejora y consiste en añadir un término regularizador a la función de coste. El término regularizador añade restricciones adicionales con la intención de prevenir el sobreajuste. El término regularizador elegido es la norma L2 de los pesos multiplicada por un coeficiente λ de 0.01, que indica cuánto queremos penalizar la flexibilidad de nuestro modelo para explorar el espacio de pesos. En la figura 4.3 se observa que añadir un término regularizador a la función de coste ha tenido un éxito claro, evitando en cierta medida el sobreajuste que se observaba en la versión 1 e incluso aumentado el valor de PSNR. Debido a estos resultados, optamos por mantener el término regularizador en el resto de experimentos.

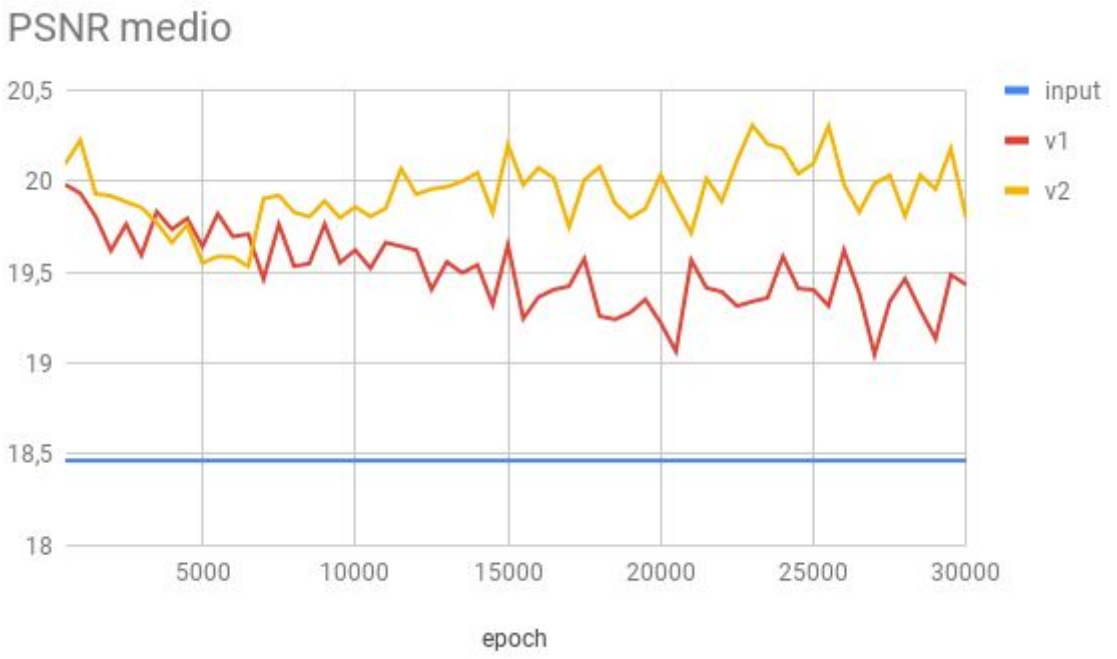

Figura 4.3. PSNR medio sobre el conjunto de test de la versión 2 frente a la versión 1.

La tercera versión añade un coeficiente de aprendizaje dinámico que decrece conforme avanza el entrenamiento, técnica conocida como *learning rate decay*. Esta práctica lo que busca es ir disminuyendo el coeficiente de aprendizaje con el fin de afinar cada vez más. Normalmente se empieza con un coeficiente generoso para avanzar rápidamente en las primeras épocas, pero puede que más adelante se llegue a un punto en que el método de descenso de gradiente se ve atascado dado que las actualizaciones de los parámetros con tal coeficiente son demasiado grandes y no encuentra la forma de descender. Nosotros hemos implementado un decrecimiento exponencial del coeficiente de aprendizaje con la función `tf.train.exponential_decay`[8] de TensorFlow, con un decremento base de 0.95 cada 1.000 iteraciones. En la figura 4.4 se observa como la versión 3 consigue una pequeña mejora del PSNR en general, destacando sobretodo en las primeras épocas (2.500 a 7.500) y consiguiendo a partir de la época 15.000 una mejor estabilización.

[8] https://www.tensorflow.org/api_docs/python/tf/train/exponential_decay

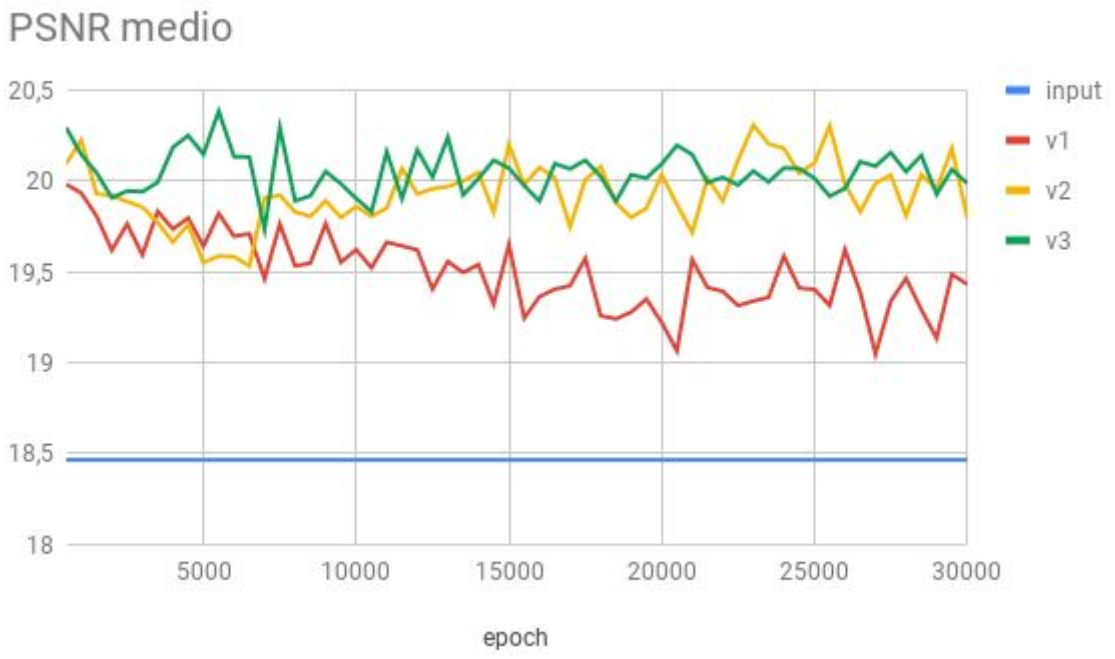

Figura 4.4. PSNR medio sobre el conjunto de test de la versión 3 frente a las previas.

La versión 4 parte de la anterior, es decir, que incluye tanto el decrecimiento del coeficiente de aprendizaje como el término regularizador de la función de coste. En esta versión se ha querido probar si activando la opción de usar Nesterov Momentum que ofrece el optimizador aportaba alguna mejora. En al figura 4.5 vemos que aunque parece que los resultados están algo más estabilizados no llegan a ser tan buenos como los conseguidos con la versión 3. Por ello, descartamos el uso de esta opción en otros experimentos.

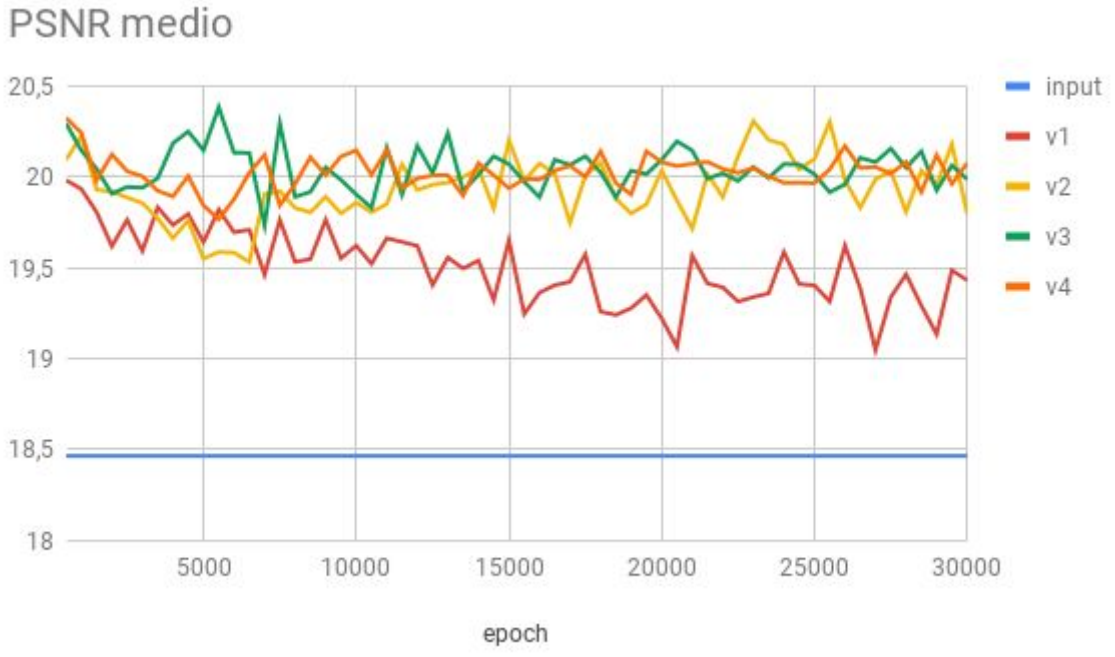

Figura 4.5. PSNR medio sobre el conjunto de test de la versión 4 frente a las previas.

La quinta y última versión de esta batería de experimentos consiste en cambiar por completo el optimizador utilizado, pasando de tener un `MomentumOptimizer` a un `AdamOptimizer`[9]. El término ADAM es acrónimo de *adaptive moment estimation*. A diferencia del descenso por gradiente convencional, `AdamOptimizer` calcula un coeficiente de aprendizaje adaptativo por parámetro, y basa su funcionamiento en estimaciones de la media y la varianza de los gradientes. Este optimizador requiere de otros parámetros en vez de *momentum*, que serán los recomendados por TensorFlow: beta1 = 0.9, beta2 = 0.999 y epsilon = 1e-08. Además, este optimizador también recibe como parámetro el coeficiente de aprendizaje inicial y él mismo encierra un cálculo adaptativo por parámetros, por lo que se elimina también el decrecimiento exponencial añadido en la versión 3. Sí se mantiene el término regularizador que se introdujo en la versión 2. En la figura 4.6, esta versión ha demostrado ser la más efectiva, obteniendo resultados de PSNR bastante superiores en las primeras 5.000 épocas y estando a un nivel igual o superior en el resto del entrenamiento pero con una mayor estabilidad. Además, parece que también tiene síntomas de sobreajuste, por lo que se podría mejorar el resultado de esta versión modificando otros aspectos. Finalmente, optamos por elegir este optimizador para ser usado en el resto de experimentos.

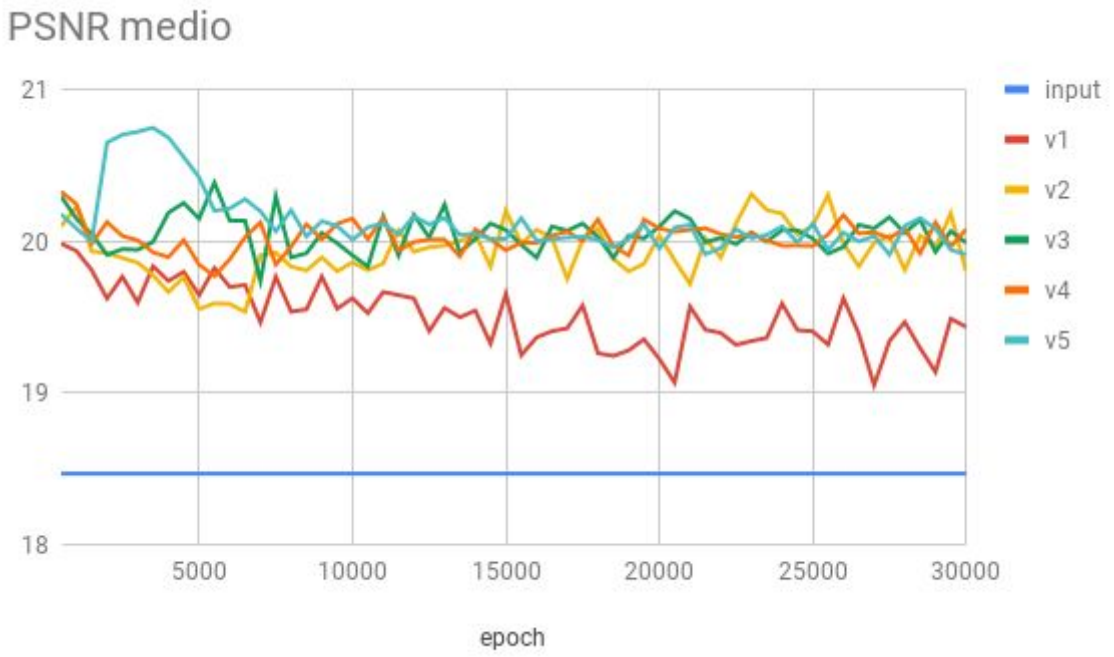

Figura 4.6. PSNR medio sobre el conjunto de test de la versión 5 frente a las previas.

[9] https://www.tensorflow.org/api_docs/python/tf/train/AdamOptimizer

4.3.2 Factor de escala

Para conseguir ir de una imagen S3 de baja resolución (366x366), de la que partimos, a una imagen S2 de alta resolución (5490x5490), hace falta un escalado de 15x. Pretender que la red pueda obtener mejoras a tal factor de escala sería muy optimista. Por ello, se ha querido probar hasta qué factor de escala la red puede ofrecer mejoras sustanciales. Por desgracia y debido a las limitaciones de cómputo, solo hemos podido probar con factores de escala de 2x, 3x y 4x.

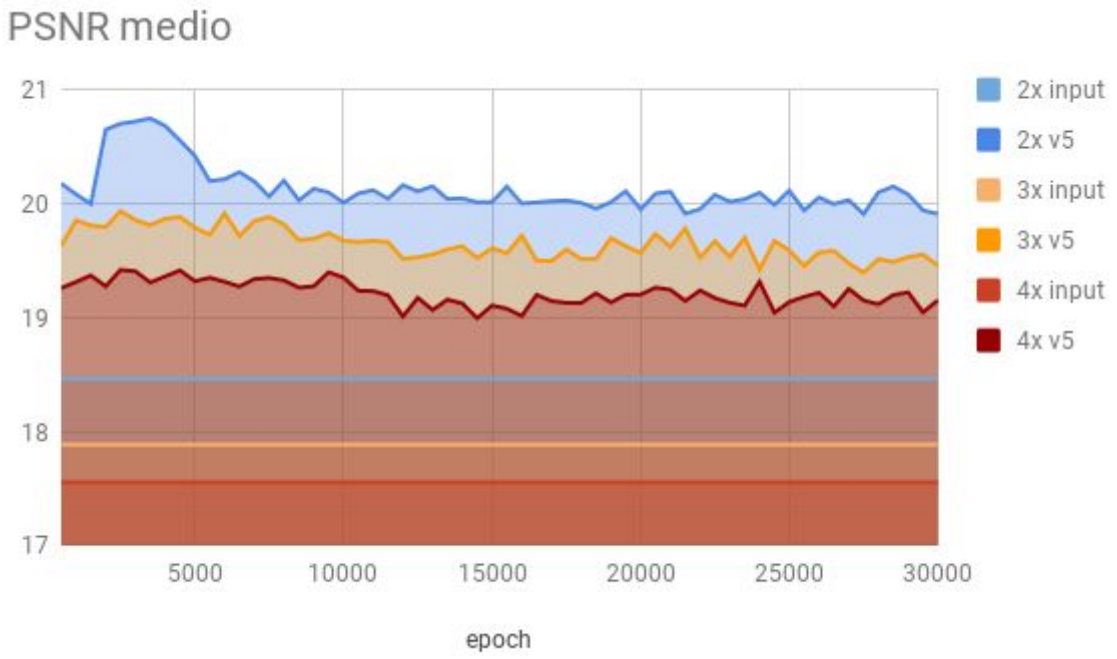

Figura 4.7. PSNR de entrada y salida en las escalas 2x, 3x y 4x.

En la figura 4.7 se puede ver para cada factor de escala cuál sería el PSNR medio sobre las imágenes de entrada de test y del resultado de nuestra red. Como es lógico, la precisión disminuye a medida que aumenta el factor de escala, tanto en la entrada como en la salida. En las tres escalas probadas la diferencia en el valor de PSNR entre la entrada y la salida tiene una magnitud similar, por lo que podemos decir que hasta 4x la red consigue resultados sin ningún tipo de penalización drástica. También es interesante observar cómo a mayor escala, y por tanto a mayor número de muestras de entrada, la red parece no sobreajustarse tanto al conjunto de entrenamiento y se estabiliza mejor.

4.3.3 Duración del entrenamiento

En un inicio los experimentos se han limitado a un entrenamiento de 30.000 épocas debido a las limitaciones de cómputo, pero como no sabíamos si existía posibilidad de mejora probamos a entrenarla durante 100.000 épocas.

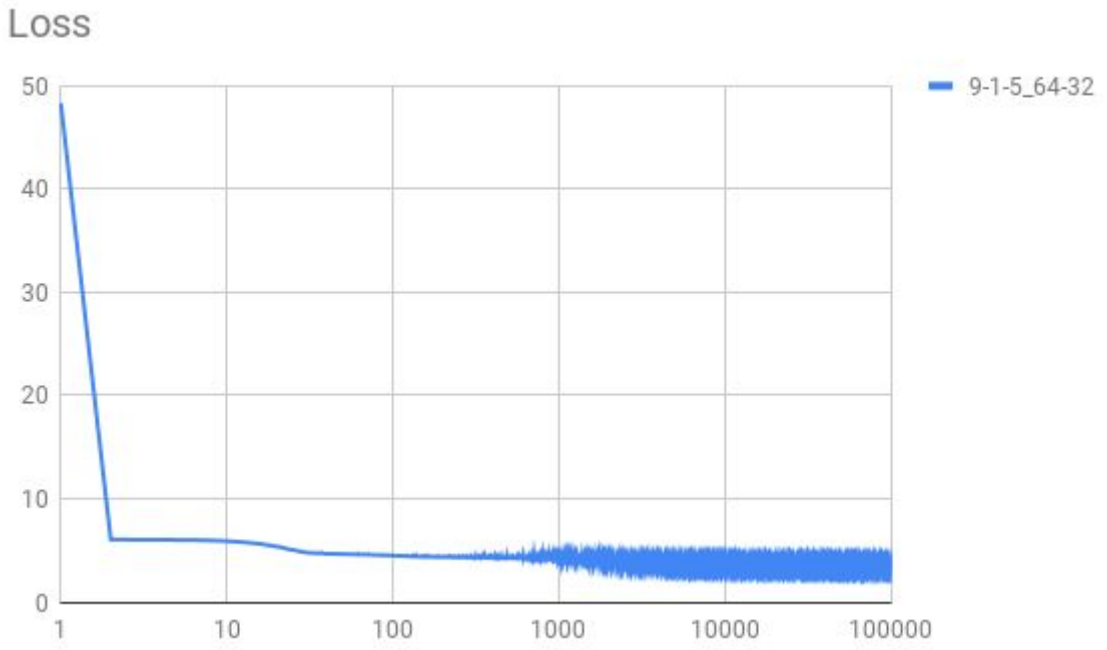

Figura 4.8. Resultado de la función de coste por época sobre el conjunto de entrenamiento.

Viendo las figuras 4.8 y 4.9, claramente podemos decir que alargar el entrenamiento a 100.000 épocas no ha aportado ninguna mejora significativa, aunque esto no sería suficiente para afirmar que con todavía más entrenamiento pudiese aparecer alguna mejora en el rendimiento de la red.

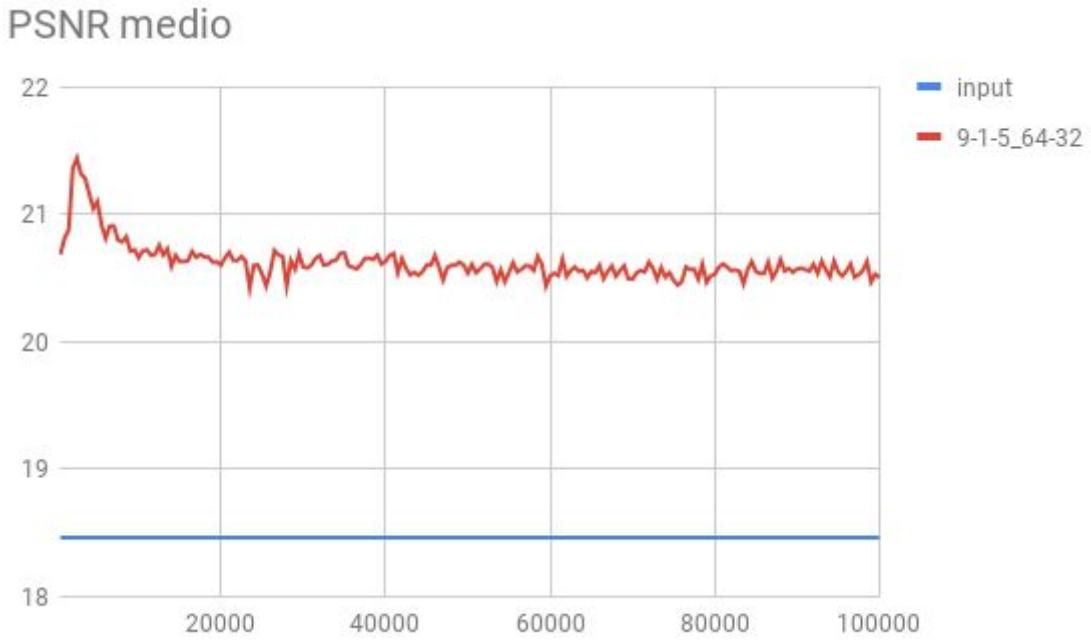

Figura 4.9. PSNR medio sobre el conjunto de test.

4.3.4 Anchura de la red y ruido en la entrada

En los experimentos previos la red ha mostrado síntomas de sobreentrenamiento u *overfitting*, debido quizás en parte a la relativamente poca cantidad de datos que disponemos para su entrenamiento. La configuración inicial propuesta en el trabajo de Dong et al. tiene un número de filtros, cuya capacidad de aprendizaje podría ser demasiado elevada para la cantidad y variedad de muestras disponibles. Es decir, el problema podría ser demasiado simple para la capacidad de la red, y los parámetros aprendidos estarse ajustando en exceso a este conjunto específico de muestras. Así, la red podría no terminar de generalizar como debería, y su precisión verse afectada sobre el conjunto de test.

Se han realizado diversos experimentos reduciendo el número de filtros de las capas ocultas de 64-32 a 32-16 y a 16-8; además se ha probado a añadir ruido gaussiano sobre las imágenes de entrada en los tres modelos para ver si esto añadía algo más de varianza en los datos y permitía a la red generalizar mejor.

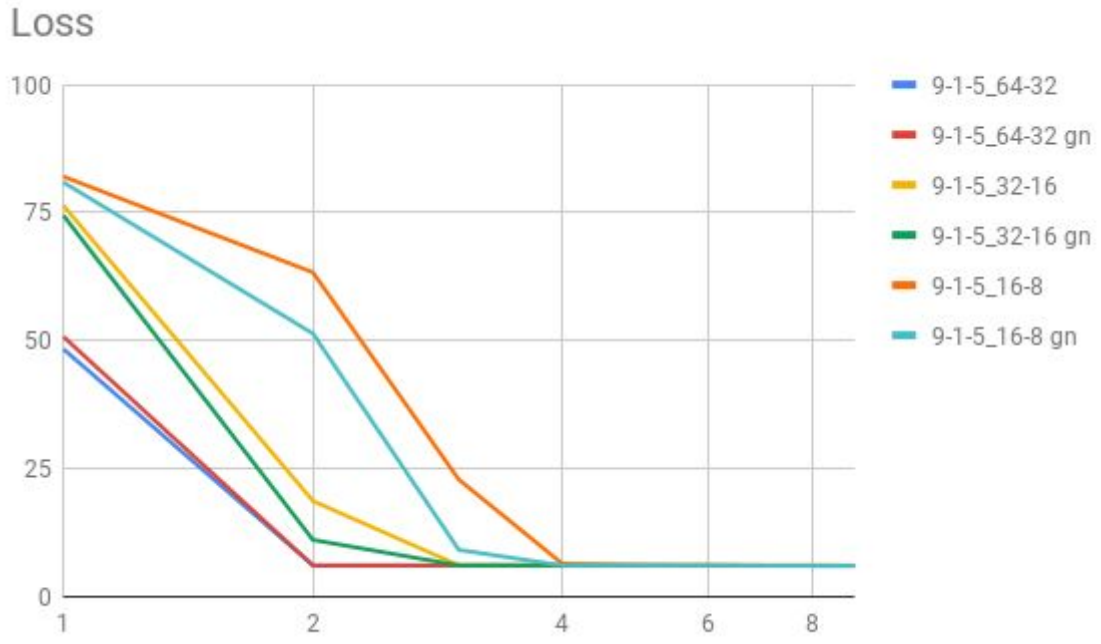

Figura 4.10. Resultado de la función de coste en las primeras 10 épocas, a partir de ahí ya no baja ni sube más.

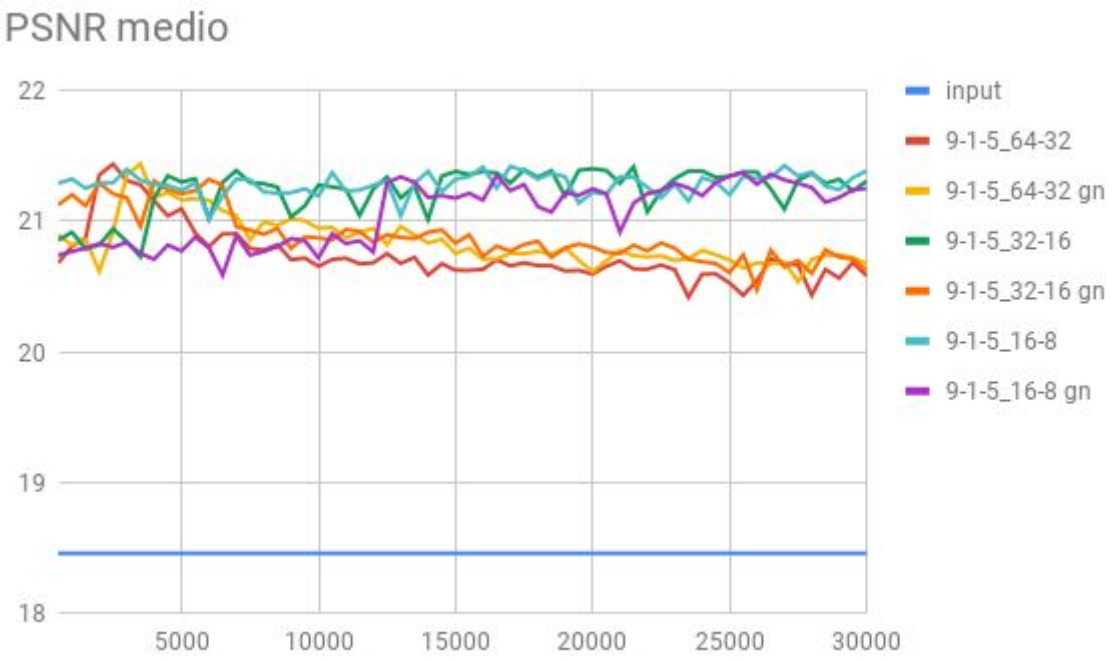

Figura 4.11. PSNR medio obtenido durante las primeras 30.000 épocas con los diferentes número de filtros y usando ruido gaussiano en la entrada (gn) o no.

Los resultados que se muestran en la figura 4.10 y 4.11 demuestran que usando un menor número de filtros la red tarda más en converger durante el entrenamiento, lo que indica que tiene menos grados de libertad, pero sin embargo obtienen un rendimiento superior durante el test lo que indica que estamos ante un proceso de aprendizaje más sano. Por otro lado, aplicar ruido gaussiano en la entrada no dio los resultados esperados. No obstante, debe resaltarse que la red con la configuración de 16-8 con ruido gaussiano tuvo un rendimiento notablemente mejor que la red con 64-32 sin ruido, y terminó mostrando un comportamiento similar a los modelos 16-8 y 32-16 sin ruido añadido en los datos de entrada.

4.4 Comparación con otros métodos

Tras los diferentes experimentos previos, en los que el modelo de CNN ha mostrado un rendimiento notable en la superresolución de imágenes S3, hemos querido comparar sus resultados con los de otros métodos más convencionales de superresolución. En la comparación vamos a usar la interpolación bicúbica (BCI) como referencia base; así pues el resto de métodos deberán superar a esta para considerar que realmente aportan valor en la solución del problema. El resto de métodos serán IBP (Iterative Back Projection) [8], que es un algoritmo de reconstrucción sin entrenamiento, ANR (Anchored Neighborhood Regression) [9], ANR+ (Adjusted Anchored Neighborhood Regression) [10] y LKR (Learnable Kernel Regression) [11] que son algoritmos de aprendizaje con entrenamiento y, por supuesto, nuestra implementación de SRCNN (Super-Resolution Convolutional Neural Network) que es un método basado en redes neuronales convolucionales.

El tiempo de entrenamiento depende de cada método. En el caso de IBP no se realiza ningún entrenamiento. Los algoritmos ANR, ANR+ y LKR han sido entrenados utilizando los parámetros por defecto que proponen los autores de los métodos durante 3, 8 y 2 horas respectivamente. Nuestro modelo SRCNN lo hemos entrenado en la mayoría de experimentos durante 30.000 épocas. Los tiempos de entrenamiento entre métodos no son comparables ya que depende mucho del tipo de optimización de cada técnica, no obstante se han utilizado las configuraciones que según los autores proporcionan el mejor rendimiento sobre colecciones de imágenes estándar.

Los resultados se basan en el PSNR medio máximo obtenido sobre el conjunto de imágenes de test, por cada método y para los factores de escala 2x, 3x y 4x. Concretamente hemos utilizado la versión 5 de nuestro modelo SRCNN pues es la que mejores resultados ha aportado, pero dado que queremos comparar la precisión de los métodos en diferentes escalas hemos utilizado los resultados de los correspondientes experimentos y no el de la configuración con un menor número de filtros ya que a pesar de que ofrecía un rendimiento superior, solamente teníamos datos para el factor de escala 2x.

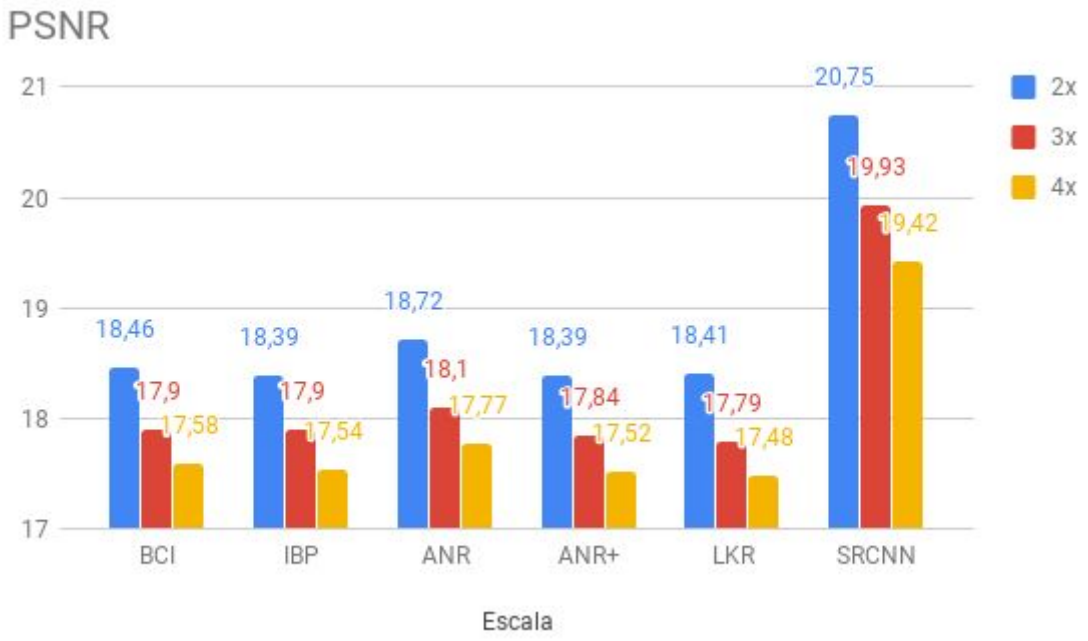

Figura 4.12. PSNR medio máximo por método y factor de escala.

En la figura 4.12 se puede observar como todos los métodos parecen verse afectados de igual manera con el aumento de escala. Los métodos IBP, ANR+ y LKR han resultado ser peores que la interpolación bicúbica, mientras que ANR y SRCNN han aportado mejora. La notable diferencia entre SRCNN y el resto de métodos es un claro indicador del potencial que este método tiene para la superresolución de imágenes gracias al uso de las redes neuronales convolucionales profundas.

Además de centrarnos en la capacidad de superresolución espacial, también se ha querido comprobar la capacidad de superresolución inter-sensor de nuestro modelo, es decir, no sólo aprender los detalles estructurales de la imagen sino también las diferencias radiométricas entre los sensores S3 y S2. Dado que el método SRCNN es el único capaz de llegar a realizar esta superresolución inter-sensor, en el resto de métodos se ha tenido que realizar una post-ecualización de las imágenes resultantes con el fin de que sean equiparables.

En la figura 4.13 podemos observar como la post-ecualización de los resultados, excepto en el método SRCNN, ha producido un notable mejora. Ahora los métodos IBP y ANR+ han superado a la interpolación bicúbica mientras que el método ANR ha pasado a estar por debajo de la referencia. El método LKR ha superado a BCI en 2x, pero no en 3x ni 4x. Nuestro modelo SRCNN no ha variado respecto a la anterior gráfica, pero aún así continúa siendo, sin duda, la mejor apuesta.

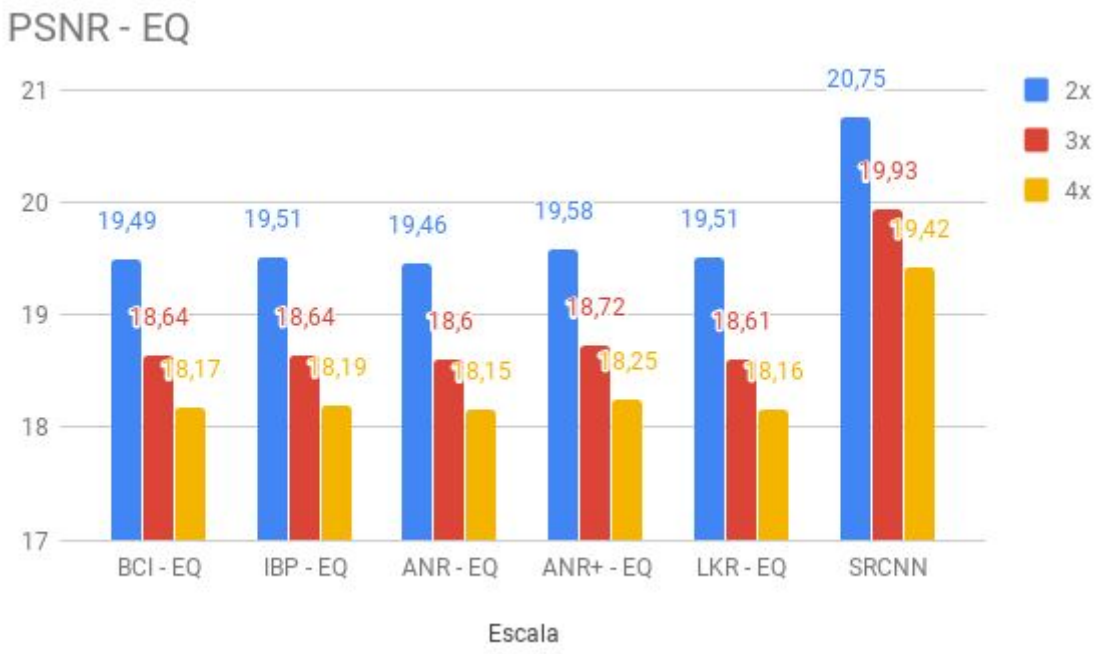

Figura 4.13. PSNR medio máximo de los resultados ecualizados por método y factor de escala. Los resultados de SRCNN no han sido ecualizados.

Como se ha comentado antes, la configuración de SRCNN usada en las dos anteriores gráficas no era la mejor de las probadas en los experimentos. Por ello, se añade una comparativa de los métodos solamente en escala 2x y usando la configuración más óptima que es la que utiliza un menor número de filtros (16-8).

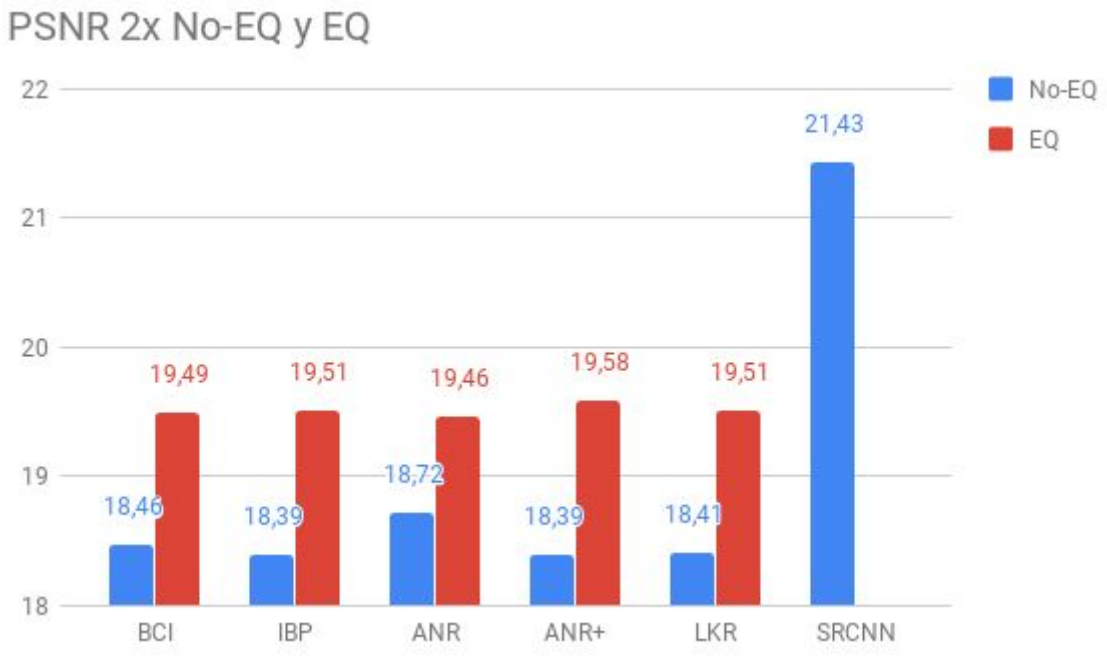

Figura 4.14. PSNR medio máximo de los resultados ecualizados y sin ecualizar por método.

En la figura 4.14 vemos que el método SRCNN tiene todavía mucho más potencial si se optimiza, ofreciendo valores de PSNR imposibles para los métodos convencionales.

Capítulo 5. Conclusiones

En este trabajo se ha evaluado el potencial del modelo SRCNN sobre un problema real: superresolución inter-sensor de imágenes de satélite. Partiendo de las imágenes de un sensor de baja resolución espacial con un determinado número de bandas en un rango de frecuencias (S3), el objetivo de este problema de superresolución no sólo ha sido lograr una mayor resolución espacial, sino también una proyección de la información espectral original a la de un sensor objetivo (S2), utilizado como referencia para la superresolución.

La versión original del modelo fue sucesivamente analizada junto a distintas técnicas de aprendizaje, entre ellas, la introducción de un término de regularización, el uso de un coeficiente de aprendizaje adaptativo, y distintos optimizadores. Los experimentos permitieron evaluar, de forma incremental, el impacto de cada una de estas acciones en el rendimiento de la red. En casi todos los casos se observaron mejoras acumuladas en la calidad de los resultados.

Además, se evaluaron configuraciones del modelo distintas a la original, que nos permitieron explorar la relación entre la complejidad del problema bajo análisis y la capacidad de aprendizaje del modelo. En particular, se redujeron las cantidades de filtro en las capas 1 y 2, siempre manteniendo el patrón de pirámide invertida, es decir, de mayor a menor número de filtros. Se pudieron observar patrones de sobreaprendizaje con capacidades relativamente altas, y procesos de aprendizaje más efectivos cuando se redujo la capacidad del modelo para adaptarla a la complejidad del problema. Estos últimos estuvieron caracterizados por curvas de aprendizaje más lentas, y un mejor rendimiento sobre el conjunto de prueba o test.

Otra técnica de aprendizaje estudiada fue la adición de ruido gaussiano en las muestras de entrada. Esta práctica suele producir datos con mayor variabilidad, los cuales conducen habitualmente a una mejor generalización. En el caso del problema de interés, aunque hubo algún resultado prometedor, añadir ruido gaussiano no introdujo nuevas mejoras.

Finalmente, el método SRCNN fue comparado con otros métodos de superresolución convencionales. Los resultados demostraron el potencial de las redes neuronales convolucionales en la tarea de superresolución inter-sensor, siendo estos claramente superiores a los obtenidos de aplicar el resto de métodos.

Como trabajo futuro se propone ampliar este estudio considerando hiperparámetros no evaluados en este proyecto, entre ellos, el número de capas, el dominio del coeficiente de aprendizaje, así como el diseño de términos de regularización más apropiados para la tarea bajo análisis. Por otra parte, sería interesante explorar la utilidad de otros paradigmas de aprendizaje profundo como el de modelos generativos (autoencoders, GAN).

Bibliografía

1. Dong, C., Loy, C. C., He, K., & Tang, X. (2016). Image super-resolution using deep convolutional networks. IEEE transactions on pattern analysis and machine intelligence, 38(2), 295-307.

2. Jianchao, Y., Wright, J., Huang, T., & Ma, Y. (2008, June). Image super-resolution as sparse representation of raw image patches. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition(pp. 1-8).

3. Yang, J., Wright, J., Huang, T. S., & Ma, Y. (2010). Image super-resolution via sparse representation. IEEE transactions on image processing, 19(11), 2861-2873.

4. Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen. Diploma, Technische Universität München, 91(1).

5. Fernandez-Beltran, R., Latorre-Carmona, P., & Pla, F. (2017). Single-frame super-resolution in remote sensing: a practical overview. International Journal of Remote Sensing, 38(1), 314-354.

6. Vivone, G., Alparone, L., Chanussot, J., Dalla Mura, M., Garzelli, A., Licciardi, G. A., ... & Wald, L. (2015). A critical comparison among pansharpening algorithms. IEEE Transactions on Geoscience and Remote Sensing, 53(5), 2565-2586.

7. Verrelst, J., Muñoz, J., Alonso, L., Delegido, J., Rivera, J. P., Camps-Valls, G., & Moreno, J. (2012). Machine learning regression algorithms for biophysical parameter retrieval: Opportunities for Sentinel-2 and-3. Remote Sensing of Environment, 118, 127-139.

8. Irani, M., & Peleg, S. (1991). Improving resolution by image registration. CVGIP: Graphical models and image processing, 53(3), 231-239.

9. Timofte, R., De Smet, V., & Van Gool, L. (2013). Anchored neighborhood regression for fast example-based super-resolution. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1920-1927).

10. Timofte, R., De Smet, V., & Van Gool, L. (2014, November). A+: Adjusted anchored neighborhood regression for fast super-resolution. In Asian Conference on Computer Vision (pp. 111-126). Springer, Cham.

11. Liao, R., & Qin, Z. (2012, November). Image super-resolution using local learnable kernel regression. In Asian Conference on Computer Vision (pp. 349-360). Springer, Berlin, Heidelberg.