eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Computación

Trabajo de Fin de Grado

Deep Learning para Microscopía de Superresolución

Autor/a

Andoni Rodríguez Herrero

2020

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Computación

Trabajo de Fin de Grado

Deep Learning para Microscopía de Superresolución

Autor/a

Andoni Rodríguez Herrero

Directore/a(s)

Ignacio Arganda Carreras

Gorka Azkune Galparsoro

Agradecimientos

A Érika, que desde Madrid ha vivido conmigo toda esta aventura desde el principio, y que me ha acompañado en el sofá mientras yo trabajaba en este TFG por las noches.

A mis padres y mi hermano, que me animaron a estudiar ciencias y que me han dado todo el apoyo del mundo, tanto personal como académico.

A mis compañeros más cercanos, Joseba, Josu, Dorron, Tasio, Beñat, Merke, Borja, Berrondo, Arechaga, Aner, Ortega, que me han hecho este camino mucho más fácil y, sobre todo, divertido.

A los que han colaborado con el proyecto: Paqui Lucio, que tanto me ha enseñado (y enseñará) sobre las ciencias formales; a Miren Bermejo e Imanol Usandizaga, por enseñarme a gestionar proyectos y a mí mismo, a Ricardo Andrade por su colaboración con los datasets necesarios y al grupo Computer Vision and Pattern Recognition de la facultad, por su confianza para la beca Ikasiker de investigación.

Además, a dos profesores: Marisa Navarro, por su ayuda el curso pasado y a Iñigo Mendialdua, por toda su motivación.

Y, por supuesto, a mis directores, Nacho y Gorka, que han sido unos tutores excelentes y profesionales, siempre dispuestos a ayudar y que me han hecho aprender muchísimo. A ellos les debo un buen jamón. O quizás dos.

Resumen

El problema principal que este trabajo anual ha solucionado ha sido el aumento artificial de la resolución de imágenes procedentes de microscopios electrónicos. Los centros de investigación biomédica suelen contar con un limitado número de microscopios de super-resolución y la detección de ciertos patrones sólo se puede lograr a alta resolución, lo que implica más tiempo de exposición de las muestras y/o la utilización de microscopios más caros.

Para aliviar estos problemas, este TFG propone la obtención de imágenes a baja resolución de forma rápida y barata, para después aumentar artificialmente su resolución utilizando técnicas computacionales y consiguiendo así imágenes muy cercanas o equivalentes a las que se habrían adquirido con las mismas muestras en microscopios de alta resolución. Para ello, (1) se ha estudiado profundamente el estado del arte del problema junto con las publicaciones más relevantes, (2) se ha propuesto una solución basada en la red convolucional U-Net y se han realizado experimentos con sus hiperparámetros y arquitectura, y (3) se han comparado los resultados de esa propuesta con los conjuntos de datos públicos del estado del arte. Las implementaciones se han realizado en Python, utilizando principalmente las librerías Skimage, TensorFlow y Keras.

Los resultados que se han obtenido superan los encontrados en el estado del arte en las condiciones experimentales descritas.

Abstract

The main problem that this annual work has solved has been the artificial increase of the resolution of images from electron microscopes. Biomedical research centers usually have a limited number of super-resolution microscopes and the detection of certain patterns can only be achieved at high resolution, which implies more time of exposure of the samples and / or the use of more expensive microscopes.

To alleviate these problems, this Bachelor's thesis proposes obtaining low-resolution images quickly and cheaply, and then artificially increase its resolution using computational techniques, thus obtaining images that are very close to or equivalent to those that would have been acquired with the same samples in microscopes at high resolution. For this, (1) the state of the art of the problem has been studied in depth together with the most relevant publications, (2) a solution based on the U-Net convolutional network has been proposed and experiments have been carried out with its hyperparameters and architecture, and (3) the results of this proposal have been compared with the state of the art public data sets. The implementations have been carried out in Python, mainly using the Skimage, TensorFlow and Keras libraries.

The results that have been obtained exceed those found in the state of the art under the described experimental conditions.

Índice general

Agradecimientos I

Resumen III

Abstract V

Índice general VII

Índice de figuras XI

Índice de tablas XV

1. Introducción 1

1.1. Motivación 1

1.2. Objetivos generales y específicos 3

1.3. Organización de la memoria 4

2. Fundamentos Teóricos 7

2.1. Visión por Computador 7

2.1.1. Imagen digital 8

2.1.2. Otros conceptos teóricos 11

2.2. Aprendizaje Automático 18

2.2.1. Tipos de aprendizaje automático 19
2.2.2. Redes Neuronales Artificiales 22

3. Estado del arte **35**
3.1. Métricas 35
3.1.1. Métricas más utilizadas 36
3.1.2. Métricas menos utilizas 41
3.2. Superresolución general 44
3.2.1. Fundamentos 45
3.2.2. Tipos de modelos de DL según su estructura 47
3.2.3. Tipos de upsampling 50
3.2.4. Funciones de pérdida o loss 53
3.2.5. Otras técnicas 56
3.2.6. CNNs en el estado del arte 58
3.3. Superresolución en imágenes de microscopía 59
3.3.1. PSSR 59
3.3.2. Otros artículos 64

4. Investigación **65**
4.1. Solución propuesta 65
4.1.1. Arquitecturas U-Net propuestas 65
4.1.2. Algoritmo prototipo 66
4.2. Experimentos 68
4.2.1. Conjuntos de datos y puntos de referencia 68
4.2.2. Búsqueda de hiperparámetros 69
4.2.3. Cambios en la arquitectura, el upsampling y la función de loss . . 73
4.2.4. Comparación con el estado del arte 77

5. Conclusiones **81**

5.1. Sobre el trabajo 81

5.1.1. Hitos del trabajo 81

5.1.2. Líneas paralelas 82

5.1.3. Trabajo futuro 83

5.2. Conclusiones personales 83

Anexos

A. Gestión del proyecto **87**

A.1. Inicio 87

A.1.1. Descripción del proyecto 87

A.1.2. Descripción de interesados 88

A.2. Planificación 89

A.2.1. Planificación del alcance 89

A.2.2. Planificación del tiempo 91

A.2.3. Planificación de riesgos 94

A.2.4. Otras planificaciones 97

A.3. Control 99

A.4. Finalización 99

B. Comparación de imágenes **105**

B.1. Imagen 1 105

B.2. Imagen 2 105

C. Tabla de experimentos de búsqueda de hiperparámetros **109**

Bibliografía **113**

Índice de figuras

2.1. Relación de la VC con otros campos de estudio. 8
2.2. Capas de intensidades de rojo, verde y azul. 10
2.3. Una imagen gris. 12
2.4. Imagen original (a la izquierda) y con ruido gaussiano (a la derecha). . . . 13
2.5. Imagen original (a la izquierda) y con ruido SP (a la derecha). 13
2.6. Una misma imagen de tamaño 1×1 a tamaño 100×100. 14
2.7. Una misma imagen con mayor (derecha) y menor (izquierda) resolución. . 15
2.8. Misma imagen a dimensiones 40x40 y a 160x160. 16
2.9. Interpolación de imágenes hecha por método bicúbico. 16
2.10. Imagen a LR (izquierda), que es consecuencia de pasársele un desenfoque gaussiano a la imagen HR (derecha). 18
2.11. Idea general del AS. 20
2.12. Encuadre típico de un escenario de aprendizaje por refuerzo: un agente se se encuentra en un estado (state) y realiza una acción (action) en un entorno (environment), a lo cual un intérprete (interpreter) reacciona otorgándole una recompensa (reward) al agente, y haciéndolo cambiar de estado 21
2.13. Función de activación sobre función de transferencia. 23
2.14. Función ReLU. 24
2.15. Función sigmoide. 25

2.16. Una NN organizada por capas: se ven una capa de entrada (input layer), dos capas ocultas (hidden layer) y una capa de salida (output layer) 25

2.17. La topología de NN más común: el perceptrón multicapa fully connected. 26

2.18. Convolución de imagen con filtro. 27

2.19. Reducción de una imagen por max pooling. 28

2.20. Una CNN completa. 28

2.21. Ejemplo de bloque residual. 29

2.22. Una CNN tradicional (izquierda) frente a una Resnet de 34 capas (derecha), también conocida como ResNet34. 31

2.23. Arquitectura de la U-Net. 32

2.24. Esquema de backpropagation. 34

3.1. Funcionamiento de cálculo de SSIM. Las propiedades de luminosidad y contraste se calculan dos veces (con las dos imágenes originales y con las resultantes de los cálculos) y la estructura solamente con las resultantes; y después se combinan los resultados. 38

3.2. Evolución de las gráficas de PSNR y SSIM en función de la covarianza . 42

3.3. Esquema del método MS-SSIM; al igual que en SSIM 3.1, se calculan iterativamente las disparidades en luminosidad y contraste, y la estructura se calcula solamente al final. 43

3.4. Taxonomía del campo de la SR. 46

3.5. Arquitecturas de CNN, en función de la posición del upsampling en la red; en a una arquitectura pre-upsampling 3.2.2, en b una arquitectura post-upsampling 3.2.2, en c una arquitectura de upsampling progresivo 3.2.2 y en d una arquitectura con upsampling iterativo 3.2.2. 48

3.6. Pasos del algoritmo de interpolación bilineal. 51

3.7. Pasos de una capa convolucional transpuesta. 52

3.8. Pasos de una capa subpíxel. 52

3.9. Batch normalization intuitivamente: se normaliza la salida de cada capa. . 57

3.10. Data augmentation con varias traslacione sobre la imagen original (arriba) y varias rotaciones sobre la imagen original (abajo). 58

3.11. CNNs significativas para el problema de SR (las referencias de la primera columna se corresponden con las de la revisión [Wang et al., 2020], no con las de este trabajo). 59

3.12. Tríangulo de compromiso en VC 60

3.13. Entrenamiento de PSSR: Creación de datos de entrenamiento semi-sintéticos (a la izquierda) y arquitectura de red propuesta (a la derecha). 61

3.14. En la subfigura b, el dataset semisintético reconstruido de diferentes maneras con el método bilineal y el método propuesto, juntos con sus resultados en las métricas; en la subfigura c, lo mismo con el dataset de datos reales. Los recuadros blancos muestran diferentes zooms de la misma imagen. 62

4.1. Una diferente configuración de la U-Net: con ELU, dropout, average pooling y una entrada de 256×256. 66

4.2. Dos imágenes del conjunto de datos EM (son en realidad el recorte de una misma imagen más grande). 69

4.3. Resultados en PSNR y SSIM; se muestran sus resultados estadísticos con la media y valores atípicos. 70

4.4. Experimentos de 4.2.3 en base a la taxonomía de [Wang et al., 2020]; en verde los apartados investigados y en azul los apartados investigados e implementados. 73

A.1. Correspondencia entre grupos de procesos y áreas de conocimiento de la dirección de proyectos, según PMBok. 101

A.2. EDT/WBS simplificado del proyecto 102

A.3. Cronograma simplificado del proyecto 102

A.4. EDT/WBS simplificado de la segunda iteración del proyecto 103

A.5. Cronograma simplificado de la segunda iteración del proyecto 103

B.1. Imagen 1 a LR con ruido. 106

B.2. Imagen 1 reconstruida a HR por el modelo. 106

B.3. Imagen 1 original a HR (el objetivo). 107

B.4. Imagen 2 a LR con ruido. 107

B.5. Imagen 2 reconstruida a HR por el modelo. 108

B.6. Imagen 2 original a HR (el objetivo). 108

Índice de tablas

4.1. Características comunes de los experimentos en la Sección 4.2.2 y 4.2.3. . 71

4.2. Resultados en SRx2. 74

4.3. Resultados en SRx4. 75

4.4. Primeros resultados de nuestros modelos para el dataset de la Sección 4.2.1. 78

4.5. Resultados definitivos de nuestros modelos para el dataset de la Sección 4.2.1. 78

5.1. Mejor resultado sobre el dataset de la Sección 4.2.1 82

A.1. Estimación de la duración del proyecto en base a sus paquetes de trabajo. 94

1. CAPÍTULO

Introducción

1.1. Motivación

La motivación científica y una motivación a nivel de aplicación justifican el porqué del trabajo, además de ofrecerle un contexto más entendible.

Motivación científica

Durante décadas, los sistemas expertos [1] y la lógica simbólica [2] han sido los subcampos dominantes en la Inteligencia Artificial (o IA)[3]; tanto los que más inversión han recibido por parte de la industria, como los que mayor foco de atención han tenido en la academia.

Sin embargo, en los últimos años ha habido un cambio de paradigma. Pese a que la tecnología del aprendizaje automático (o Machine Learning) [4] no sea en absoluto nueva, han sido los constantes avances en el mundo del hardware los que han conseguido que sea viable, acelerando así todas las investigaciones sobre el tema y sustituyendo a los sistemas basados en conocimiento como el estandarte de la IA.

Dentro de ese paradigma que actualmente es líder encontramos una familia de algoritmos llamada aprendizaje supervisado [5], y el modelo computacional tan de moda que son las

[1] https://es.wikipedia.org/wiki/Sistema_experto
[2] https://es.wikipedia.org/wiki/Lógica_matemática
[3] https://es.wikipedia.org/wiki/Inteligencia_artificial
[4] https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico
[5] https://es.wikipedia.org/wiki/Aprendizaje_supervisado

redes neuronales. Como parte de ellos, el Deep Learning (o DL) [6] se ha visto extremadamente favorecido e intenta modelar las abstracciones de más alto nivel que se hayan visto hasta la fecha. Así, fruto de los trabajos en este campo, tenemos hoy en día aplicaciones que antes no se habían logrado en visión por computador [7] o en procesamiento del lenguaje natural [8], como la transferencia de estilo (del inglés style transfer [9]) o la clonación de voz [10].

Una de las varias temáticas abordadas por el DL ha sido la superresolución, que, de manera muy simplificada, consiste en coger una imagen y pasarla por un proceso que nos permitirá obtener la misma imagen pero con mayor resolución, lo cual supone un reto debido a que se debe inferir nueva información para reconstruir la imagen a mayor resolución.

La Superresolución (o SR) tampoco es un problema nuevo y ya se ha explorado desde múltiples puntos de vista, pero ha sido el DL lo que ha permitido hacer escalar más que antes a sus resultados, desarrollando modelos que aprenden a mejorar imágenes de forma muy positiva, haciendo difícil distinguir cualitativamente imágenes reales de las predichas.

Existe un subcampo de la SR por DL (llamado Deep Learning Super-resolution o DLSR) que explora cómo mejorar imágenes de microscopio, para así obtener imágenes más nítidas y detalladas. Esto supone ventajas al mundo de la medicina y la biología, pues ahora se podrían detectar formas y detalles de una manera mucho más acertada, lo cual ayudaría a la detección de enfermedades o de elementos deseados.

El trabajo de referencia en el estado del arte ha sido el artículo Point Scanning Super-resolution (que llamaremos PSSR) [Fang et al., 2019], que estudiaba cómo se habían logrado aumentar considerablemente de resolución imágenes de microscopía electrónica, utilizando diferentes técnicas de DL.

A día de hoy, el campo DLSR tiene más publicaciones que nunca antes y ofrece numerosos retos e incógnitas para los investigadores, como para interesarse por este campo.

[6] https://es.wikipedia.org/wiki/Aprendizaje_profundo
[7] https://es.wikipedia.org/wiki/Visi%C3%B3n_artificial
[8] https://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales
[9] https://en.wikipedia.org/wiki/Neural_Style_Transfer
[10] https://medium.com/the-research-nest/voice-cloning-using-deep-learning-166f1b8d8595

Motivación práctica

La SR tiene una aplicación general, puesto que mejora la resolución de imágenes, y esto a su vez puede ayudar a tareas de dominio específico como detección de características en rostros [Chen et al., 2017] o desarrollo de videojuegos [11].

En el caso concreto de este TFG, interesa la aplicación que la SR por DL tiene en el campo de la miscroscopía, es decir, mejorar la resolución de imágenes obtenidas por microscopios.

En un laboratorio de Bilbao, el Dr. Ricardo Andrade (del Servicio General de Microscopía Analítica y de Alta Resolución en Biomedicina) y el resto de investigadores utilizan microscopios ópticos de diferentes tipos, y unos obtienen las muestras con mayor calidad que otros, y además, a veces, se necesita obtener las muestras más rápido que en otras ocasiones, debido a la fototoxicidad [Tosheva, 2020]. El equipo, en rasgos generales, obtiene resoluciones de 140nm en los ejes X e Y, mientras que la resolución en el eje Z siempre es peor, de aproximadamente 350nm. Sin embargo, los equipos de superresolución rondan los 20-80nm.

1.2. Objetivos generales y específicos

El objetivo general ha sido implementar una red neuronal artificial [12] para hacer SR a la que, dándosele un conjunto de entrenamiento de imágenes de microscopía electrónica a baja y a alta resolución, aprende a aumentar la resolución de las imágenes de un conjunto de test.

A partir del cual se han desarrollado numerosos objetivos específicos:

1. Experiencia básica:
    - *a*) Aprender el uso básico de librerías como TensorFlow o Keras.
    - *b*) Aprender a utilizar un entornorno para trabajar (en nuestro caso Nube Google Collaboratory).
    - *c*) Entender el funcionamiento general de la red U-Net (ver Sección 2.2.2).

[11] https://riull.ull.es/xmlui/bitstream/handle/915/16588/Deep%20Learning%20en%20videojuegos%20Superresolucion.pdf?sequence=1

[12] https://es.wikipedia.org/wiki/Red_neuronal_artificial

d) Leer y entender el artículo PSSR [Fang et al., 2019].

2. Implementaciones necesarias:

   a) Implementar funciones básicas como la función para el ruido o para dividir las imágenes.

   b) Implementar una U-Net adecuada para las imágenes del problema.

   c) Implementar un método para evaluar de la mejor manera posible los resultados, tanto cuantitativa como cualitativamente.

3. Mejorar el modelo:

   a) Estudiar en profundidad el estado del arte de DL en general y en particular (ver Sección 3.2 y 3.3).

   b) Estudiar en profundidad el estado del arte de métricas las métricas, y su posible uso como función de loss (ver Sección 3.1).

   c) Realizar una búsqueda exhaustiva de hiperparámetros 4.2.2.

   d) Cambiar la arquitectura y sus características utilizando criterios propuestos por el estado del arte (ver Sección 4.2.3).

1.3. Organización de la memoria

La memoria consta de cinco capítulos, además de dos anexos:

1. Introducción (ver Sección 1): El presente capítulo, donde se da una introducción al TFG.

2. Fundamentos teóricos (ver Sección 2): Se detallan las bases teóricas necesarias para poder comprender el trabajo, tanto en visión por computador (ver Sección 2.1) como en aprendizaje automático (ver Sección 2.2).

3. Estado del arte (ver Sección 3): Se estudia en profundidad el estado del arte relacionado con el problema a solucionar, en en cuanto a métricas (ver Sección 3.1), superresolución general (ver Sección 3.2), y superresolución en microscopía (ver Sección 3.3).

4. Investigación (ver Sección 4): Se propone una solución en forma de algoritmo y arquitectura (ver Sección 4.1), así como los experimentos realizados y sus resultados (ver Sección 4.2).

5. Conclusiones (ver Sección5): Se plantea el impacto del trabajo, sus discusiones y el trabajo futuro a realizar.

2. CAPÍTULO

Fundamentos Teóricos

Se introducirán de una manera formal los campos y conceptos más relevantes para esta investigación, así como sus conceptos subyacentes que después serán obligatorios para comprender el desarrollo del TFG en la parte de estado del arte e investigación.

2.1. Visión por Computador

***Definición* 2.1** *La Visión Artificial es una disciplina de las ciencias de la computación cuyo objetivo es procesar y después utilizar, analizar o tratar las imágenes por máquina, ya sea para aplicaciones de inteligencia artificial o de otras disciplinas científicas o no científicas.*

La visión por computador (VC) es, por tanto, un campo interdisciplinar que combina diferentes campos de estudio, como se puede observar en la Figura 2.1.

Como se puede observar, la VC toca muchos y diferentes campos. Sin embargo, a lo largo de trabajo, no nos hemos centrado en cómo se obtienen las imágenes, sino en cómo trabajar con ellas una vez las tenemos.

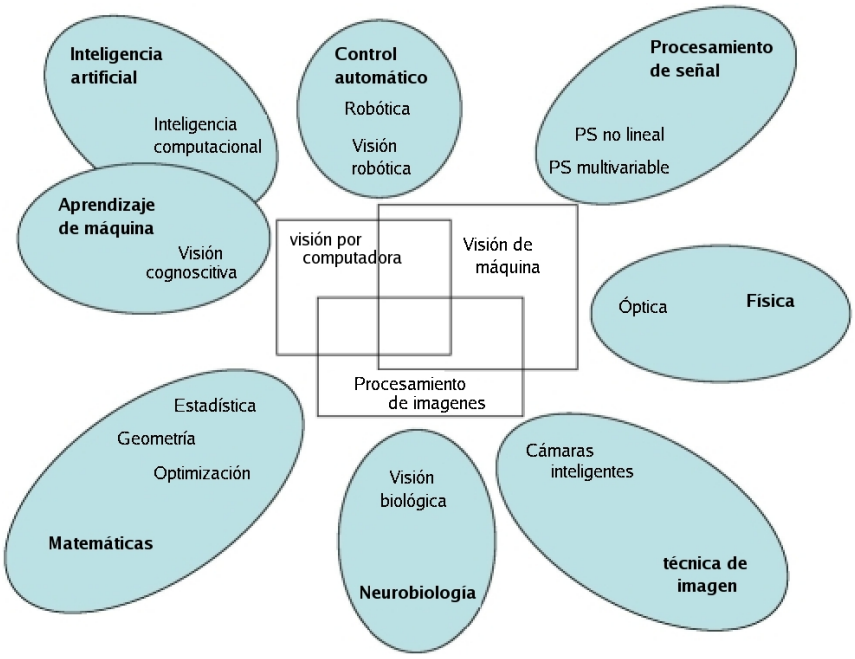

Figura 2.1: Relación de la VC con otros campos de estudio.
Fuente: https://es.wikipedia.org/wiki/Visión_artificial

2.1.1. Imagen digital

Dentro de la VC sólo utilizaremos las imágenes representables por máquina, las imágenes por computador o imágenes digitales [1].

Imagen digital formalmente

Una imagen digital tiene otra manera de representarse (como grafo vectorial [2]), pero la clásica y la que se ha utilizado aquí es la que se conoce por imagen como matriz o imagen como mapa de bits.

Una imagen digital es una matriz de dimensiones $N \times M$, donde cada elemento de la matriz es un valor discreto que, en forma de un número finito de bits, cuantifica el nivel de información de ese mismo elemento.

Podemos formalizar esta definición como función discreta [3] de dos dimensiones, como en la Definición 2.2.

[1] https://es.wikipedia.org/wiki/Imagen_digital
[2] https://www.albaplazadesigner.com/diferencias-entre-imagen-de-mapa-bits-e-imagen-vectorial/
[3] https://es.wikipedia.org/wiki/Funci%C3%B3n_discreta

Definición 2.2 *La imagen es una función discreta:*

$$I(x,y)\text{ , donde } 0 \leq x \leq N-1 \wedge 0 \leq y \leq M-1\text{, donde } N,M \in \mathbb{N}$$

Es decir, los valores x e y se mueven de 0 a N-1 y de 0 a M-1, respectivamente, donde N y M pertenecen al conjunto de los números naturales. Entonces, la función I asigna un valor a la entrada (x_i, y_j)*:*

$$0 \leq I(x,y) \leq p-1\text{, donde } p = 2^q\text{, donde } q \in \mathbb{N}$$

Esto es, el codominio [4] *de I se mueve desde 0 hasta un múltiplo de dos menos 1.*

La imagen digital es, por tanto, una función bidimensional que proporciona información por cada uno de sus valores. A estos valores discretos también se les llama punto o, principalmente, píxel [5].

Cabe destacar que, en el mundo de la imagen digital, y, por tanto en la VC, los píxeles de una imagen de 8 bits [6] (que son las que se utilizan) siempre oscilan entre los valores 0 y 255, siendo el 0 el valor de menor intensidad y el 255 el de mayor intensidad.

El motivo de esto es que la precisión de bits típica (es decir, la información que cada píxel proporciona) suele ser de 8.

Imagen como matriz de capas

La forma más común de representar una imagen es como mapa de bits [7]:

- Imaginemos una matriz de N matrices o capas (con N$\geq$1) de $A \times B$, donde cada capa posee unas propiedades concretas y, por tanto, los elementos de cada capa nos dicen cuánto de esa propiedad se cumple. A estas capas se les llama canal [8].
- Estas propiedades, pueden cuantificar diferentes cualidades. Por ejemplo, en la Fi-

[4] https://es.wikipedia.org/wiki/Codominio
[5] https://es.wikipedia.org/wiki/P%C3%ADxel
[6] https://es.wikipedia.org/wiki/Bit
[7] https://es.wikipedia.org/wiki/Imagen_de_mapa_de_bits
[8] https://es.wikipedia.org/wiki/Canal_(imagen_digital)

gura 2.2 vemos una imagen del conocido tipo RGB [9], cuyos canales se corresponden con las intensidades Rojo, Verde y Azul.

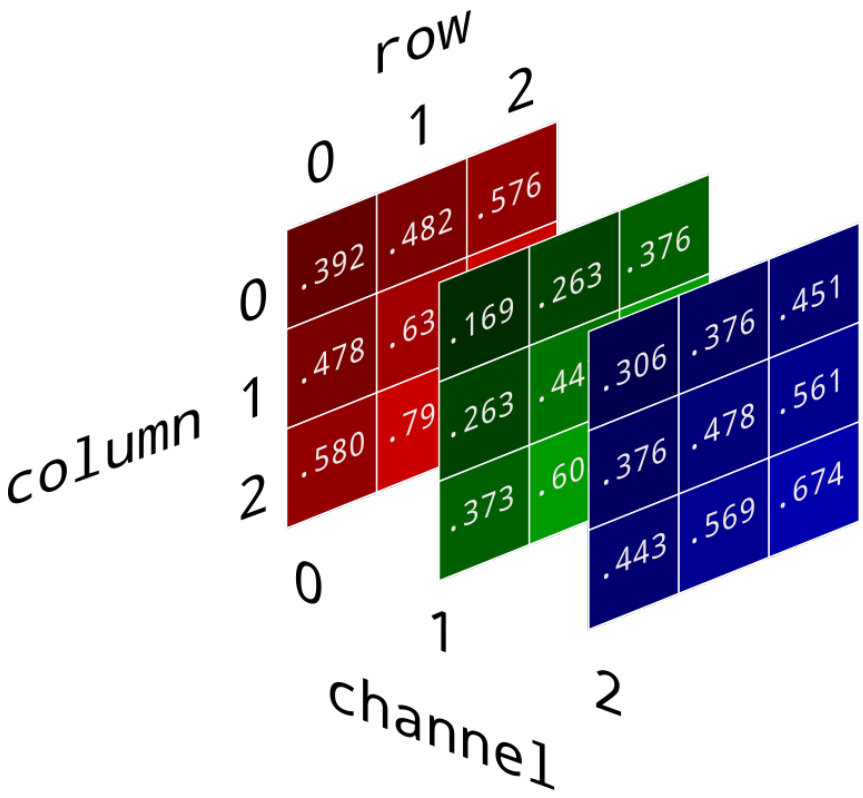

Figura 2.2: Capas de intensidades de rojo, verde y azul.
Fuente:
https://brohrer.github.io/images/image_processing/three_d_array.png

- Es decir, dicho de manera informal, los píxeles (o elementos de matriz) de la primera capa nos indican cuánto de rojo tienen; los píxeles de la segunda capa cuánto de verde y los de la tercera capa cuánto de azul.

- Una imagen, es, por tanto, la combinación de todos sus canales. Es decir, el píxel 1 de la imagen contendrá la combinación del píxel 1 de las capas 1,2,...,n; el píxel 2 de la imagen será la combinación del píxel 2 de todas las capas y así para cada píxel. Por ejemplo, cogiendo la matriz RGB de la Figura 2.2, el píxel 1 de la imagen será la combinación de 392, 169 y 306, resultando para nosotros en un color determinado, igual que mezclar colores de pintura.

[9] https://es.wikipedia.org/wiki/RGB

Imagen gris

También llamado imagen en escala de grises (o grayscale) [10], es el tipo más básico de imagen que podemos procesar. Se puede formalizar como en la Definición 2.3.

***Definición* 2.3** *Una imagen grayscale es una en la que sus únicos colores son intensidades de grises, es decir, que en una imagen gris, cada píxel representará sólo la cantidad de luz que lleva.*

Esta intensidad se mueve entre los rangos negro y blanco; significando el negro que apenas hay intensidad (o 0 % de intensidad) y el blanco que está al máximo (o 100 % de intensidad). Es por ello que se lo denomina escala de grises.

Recordemos que el valor más bajo (el negro) se representa generalmente por el 0 y el valor más alto (el blanco) por el 255. En cuanto a dimensiones, definiríamos así una imagen gris como una que se compone de un solo canal que es la propia imagen; es decir, sus dimensiones son de $N \times M (\times 1\ canal)$ ó simplemente $N \times M$.

Un ejemplo de una imagen gris se presenta en la Figura 2.3.

2.1.2. Otros conceptos teóricos

Una vez hemos explicado lo que es la VC y el concepto de imagen, veamos ahora lo que son otros fundamentos de este trabajo: el ruido, la interpolación, la resolución y la superresolución.

Ruido

En disciplinas científicas, se llama ruido a toda señal no deseada que se mezcle con las señales que se quiera transmitir [11]. En el caso de la VC, por tanto, el ruido tiene una definición relacionada con las imágenes (ver Definición 2.4).

***Definición* 2.4** *El ruido digital es toda variación de señales de la imagen original, es decir, de píxeles, que resultan en otra imagen más perturbada y, generalmente, de peor calidad que la original.*

[10] https://es.wikipedia.org/wiki/Escala_de_grises
[11] https://es.wikipedia.org/wiki/Ruido_(comunicaci%C3%B3n)

Figura 2.3: Una imagen gris.
Fuente: https://upload.wikimedia.org/wikipedia/commons/f/fa/Grayscale_8bits_palette_sample_image.png

El ruido digital (que llamaremos solamente ruido), que es en principio un problema, ha sido también origen de numerosos experimentos donde se prueban aplicaciones de VC como, por ejemplo, detección de bordes [12]. Esto es interesante, porque, en el problema de la superresolución (y, por tanto, también en el artículo de referencia [Fang et al., 2019]), el ruido juega un papel fundamental; en especial dos tipos; el Ruido Gaussiano y el Ruido SP.

El tipo de ruido más popular es el recién mencionado ruido gaussiano [13]. Cuando se aplica este ruido a una imagen, todos y cada uno de sus píxeles cambian su valor de acuerdo a una distribución normal (o gaussiana)[14], como se puede ver en la Figura 2.4

Por otro lado, el ruido llamado SP [15], que viene del inglés Salt & Pepper (sal y pimienta), es otro de los tipos de ruido más populares. Al contrario que en el ruido gaussiano, aquí los píxeles que se varían no son todos (de hecho, suelen ser muy pocos) y se caracterizan por

[12] https://es.wikipedia.org/wiki/Detector_de_bordes
[13] https://es.wikipedia.org/wiki/Ruido_gaussiano
[14] https://es.wikipedia.org/wiki/Distribuci%C3%B3n_normal
[15] https://es.wikipedia.org/wiki/Ruido_sal_y_pimienta

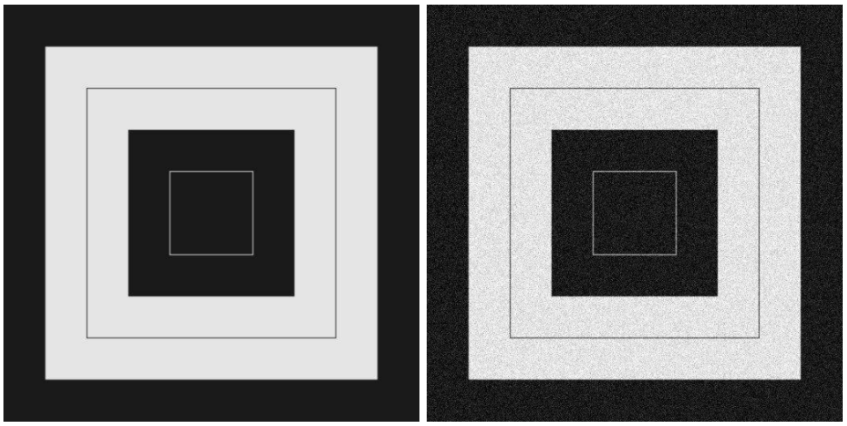

Figura 2.4: Imagen original (a la izquierda) y con ruido gaussiano (a la derecha).
Fuente: https://en.wikipedia.org/wiki/Gaussian_noise

no tener ninguna relación con sus píxeles circundantes. Se le llama sal y pimienta porque cubre la imagen de forma dispersa con una serie de puntos de intensidades máximas y mínima de grises, es decir, negros y blancos. Por ejemplo, la Figura 2.5.

Figura 2.5: Imagen original (a la izquierda) y con ruido SP (a la derecha).
Fuente: https://es.wikipedia.org/wiki/Ruido_sal_y_pimienta

Resolución

La resolución de una imagen indica la cantidad de detalles que pueden observarse en ella [16]. Informalmente, se utiliza para describir cuán nítida es una imagen, por lo que tener

[16] https://es.wikipedia.org/wiki/Resoluci%C3%B3n_de_imagen

mayor resolución se traduce en obtener una imagen con más detalle o calidad visual. Esto se formaliza en la Definición 2.5.

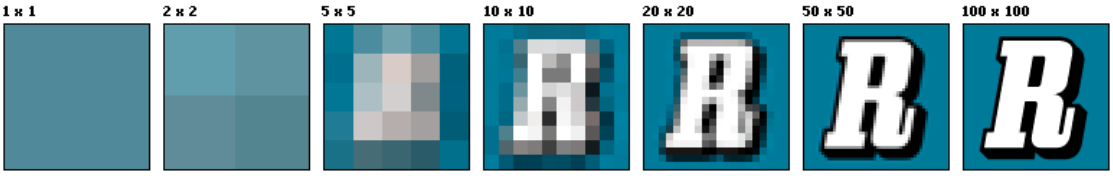

Figura 2.6: Una misma imagen de tamaño 1×1 a tamaño 100×100.
Fuente: https://es.wikipedia.org/wiki/Resolución_de_imagen

***Definición* 2.5** *La resolución indica la densidad de píxeles en un centímetro o pulgada de una imagen que se muestra, es decir, cuánto píxel hay por tramo; y, generalmente, se mide en píxeles por pulgada (ppp)* [17] *y cuántos más ppp, más calidad.*

Por tanto, si la imagen se muestra a exactamente el mismo tamaño que su matriz, entonces la resolución es igual al tamaño, pero lo normal es que esto no sea así. Por ejemplo, si tenemos una matriz de 600×1200 y vamos a mostrarla a 2×4 pulgadas, su resolución será de 300 ppp.

Para ilustrar esto, podemos efectuar un zoom [18] sobre una imagen y ver cómo la densidad de píxeles (la calidad) es mayor en la imagen con más resolución. Véanse las dos imágenes de la Figura 2.7, de mismo tamaño, donde la de la izquierda tiene 72 ppp y la de la derecha 300 ppp y se les ha aplicado un zoom de 200 % en una zona.

Así, las imágenes pueden reescalarse [19] de tamaño, pero no siempre mantendrán sus resolución:

- Si la imagen se empequeñece, su resolución puede mantenerse igual.
- Si la imagen se amplia, se tienen que generar nuevos píxeles "inventados" por interpolación.

Interpolación en imágenes

En VC, la interpolación es una técnica para generar nuevos píxeles en una imagen (ver Definición 2.6).

[17] https://es.wikipedia.org/wiki/P%C3%ADxeles_por_pulgada
[18] https://es.wikipedia.org/wiki/Zoom_digital
[19] https://en.wikipedia.org/wiki/Image_scaling

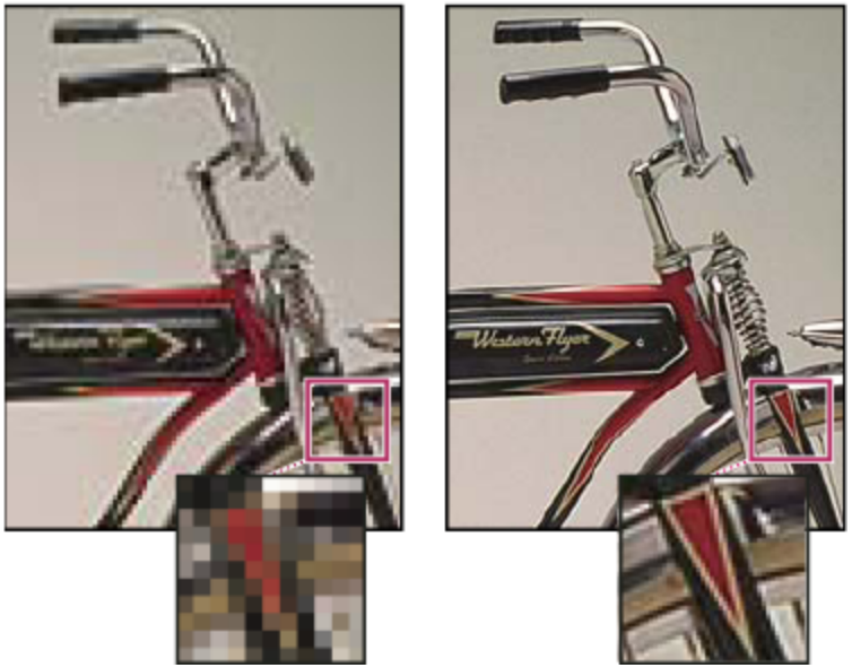

Figura 2.7: Una misma imagen con mayor (derecha) y menor (izquierda) resolución.
Fuente: http://www.tecnomovida.com/2010/11/12/que-es-el-pixelado/

***Definición* 2.6** *Se le llama interpolación* [20] *a la obtención de nuevos puntos partiendo del conocimiento de otro conjunto discreto de puntos.*

En el campo de la imagen digital, este método se aplica para conseguir un tamaño diferente de la imagen inicial (función resize [21]), rellenando la información que falta con nuevos datos calculados a partir de un algoritmo específico.

El algoritmo más utilizado es la interpolación lineal [22], pero además de ella, existen numerosos tipos más de interpolación.

Imaginemos que tenemos la Figura 2.8 de 40×40 y queremos aumentar su tamaño a 160×160, para lo cual habrá que interpolar, generando nuevos píxeles. En ella se ve la imagen original de 160×160, que se submuestreará a 40×40 y que cuya reducción se utilizará como imagen para interpolar.

Existen numerosos métodos para lograr acercarse a la imagen original, como pueden ser la interpolación bilineal [23] o la bicúbica (ver Figura 2.9) [24]. Además, existen redes con-

[20] https://es.wikipedia.org/wiki/Interpolaci%C3%B3n_(fotograf%C3%ADa)
[21] La versión de Matlab: https://www.mathworks.com/help/images/ref/imresize.html
[22] https://es.wikipedia.org/wiki/Interpolaci%C3%B3n_lineal
[23] https://es.wikipedia.org/wiki/Interpolaci%C3%B3n_bilineal
[24] Fuente: https://es.qwe.wiki/wiki/Bicubic_interpolation

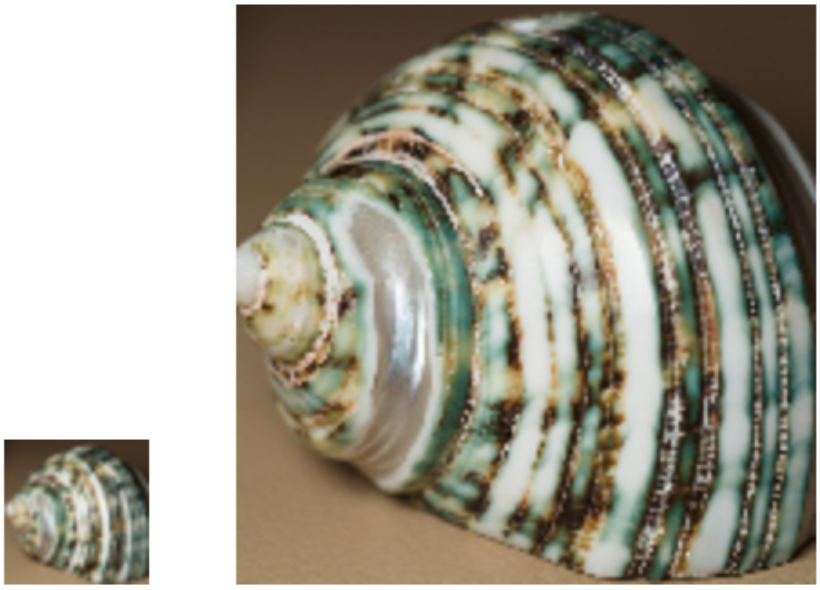

Figura 2.8: Misma imagen a dimensiones 40x40 y a 160x160.
Fuente: https://en.wikipedia.org/wiki/Image_scaling

volucionales profundas, que logran resultados positivos en interpolación.

Figura 2.9: Interpolación de imágenes hecha por método bicúbico.
Fuente: https://en.wikipedia.org/wiki/Comparison_gallery_of_image_scaling_algorithms

Por último, cabe destacar que estas no son técnicas de superresolución, sino solamente aumentar el tamaño de una imagen. Como se ha explicado al final de la Sección 2.1.2, al aumentar el tamaño de una imagen no se gana, sino que se pierde resolución.

Superresolución

La SR [25] se define en la Definición 2.7.

***Definición* 2.7** *Se le llama superresolución a toda técnica computacional para conseguir una imagen de alta resolución (HR, del inglés high resolution) partiendo de una imagen de baja resolución (LR, del inglés low resolution).*

Una imagen puede pasar de HR a LR por dos motivos:

- Por una resolución espacial menor, es decir, por menor tamaño.
- Por un proceso de degradación, como por ejemplo, añadir ruido o difuminado (blurring). Teniendo esto en cuenta, podríamos relacionar la imagen HR y su homólogo LR como se muestra en la Definición 2.8:

  ***Definición* 2.8** *Sea degradation una función que produce una degradación cualquiera:*

  $$LR = degradation(HR)$$

  Es decir, la imagen LR es la imagen HR tras pasársele una función cualquiera que degrade la imagen HR.

Como ejemplo, en la Figura 2.10 tenemos una comparación entre imagen LR y HR.

En el caso de la Figura 2.10, se puede observar la Definición 2.9.

***Definición* 2.9** *Sea gaussian blur una función de difuminado [26]:*

$$LR = gaussianBlur(HR)$$

Claramente, se puede obtener una imagen LR de la imagen HR, aplicando solamente una función de degradación. La pregunta es: ¿se puede conseguir lo contrario? En el caso

[25] https://es.wikipedia.org/wiki/Superresoluci%C3%B3n
[26] https://es.wikipedia.org/wiki/Desenfoque_gaussiano

Figura 2.10: Imagen a LR (izquierda), que es consecuencia de pasársele un desenfoque gaussiano a la imagen HR (derecha).
Fuente: https://www.mavericksinvitational.com/travel/best-budget-travel-destinations-for-2019/

ideal sería muy sencillo; si se tiene la función de degradación, sólo hace falta aplicarle la inversa [27] para obtener la imagen HR.

Sin embargo, esto no suele suceder así, y el problema de superresolución resulta ser más complejo. Esto es debido a que hay que lograr estimar esa función de degradación, tarea que es muy complicada porque hay que lograr regenerar, vía estimaciones, los datos que han producido la degradación; es decir, hay que inventar información de la manera más correcta posible.

En este trabajo se utilizan métodos de superresolución para imágenes de microscopio, lo cual no se debe confundir con la microscopía de superresolución [28], que es el conjunto de técnicas de microscopía para obtener imágenes de alta resolución.

2.2. Aprendizaje Automático

El Aprendizaje Automático (ML del inglés Machine Learning), es una de las subramas más importantes de la inteligencia artificial (ver Definición 2.10).

***Definición* 2.10** *El ML es la rama de la IA que estudia herramientas que permitan que las máquinas aprendan.*

[27] https://es.wikipedia.org/wiki/Funci%C3%B3n_inversa_(an%C3%A1lisis_matem%C3%A1tico)
[28] https://es.wikipedia.org/wiki/Microscop%C3%ADa_de_super-resoluci%C3%B3n

Para comprender esto, es necesario definir lo que significa aprendizaje, lo cual haremos de manera formal utilizando la definición de Tom M. Mitchell [29] (ver Definición 2.11).

***Definición* 2.11** *Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna clase de tareas T y la medida de desempeño P, si su desempeño en tareas en T, medido por P, mejora con la experiencia E.*

2.2.1. Tipos de aprendizaje automático

Existen tres tipos de aprendizaje automático: el aprendizaje supervisado el aprendizaje no supervisado y el aprendizaje por refuerzo.

Aprendizaje supervisado

Se llama Aprendizaje supervisado (AS) a las técnicas para deducir una función a partir de datos de entrenamiento. Los datos de entrenamiento son pares de objetos (normalmente vectores): una componente del par son los datos de entrada y el otro, los resultados deseados. La salida de la función puede ser un valor numérico (como en los problemas de regresión) o una etiqueta de clase (como en los de clasificación).

El objetivo del AS es el de crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada válida después de haber visto una serie de ejemplos: los datos de entrenamiento. Para ello, tiene que generalizar a partir de los datos presentados a las situaciones no vistas previamente.

Los problemas en aprendizaje supervisado se dividen en dos tipos [30]:

- Clasificación [31]: Se trata de asignar categorías o etiquetas a los datos que se hayan recibido; por ejemplo, clasificar si en una imagen hay gato o un perro, o si una voz es de chico o de chica...
- Regresión: También llamado optimización, se refiere al hecho de ser capaz de generar; por ejemplo, dada una imagen que la convierta a más resolución, o dada una canción que le quite los bajos...

[29] http://profsite.um.ac.ir/~monsefi/machine-learning/pdf/Machine-Learning-Tom-Mitchell.pdf

[30] https://iartificial.net/clasificacion-o-regresion/

[31] https://medium.com/datos-y-ciencia/aprendizaje-supervisado-introducci%C3%B3n-a-la-clasificaci%C3%B3n-y-principales-algoritmos-dadee99c9407

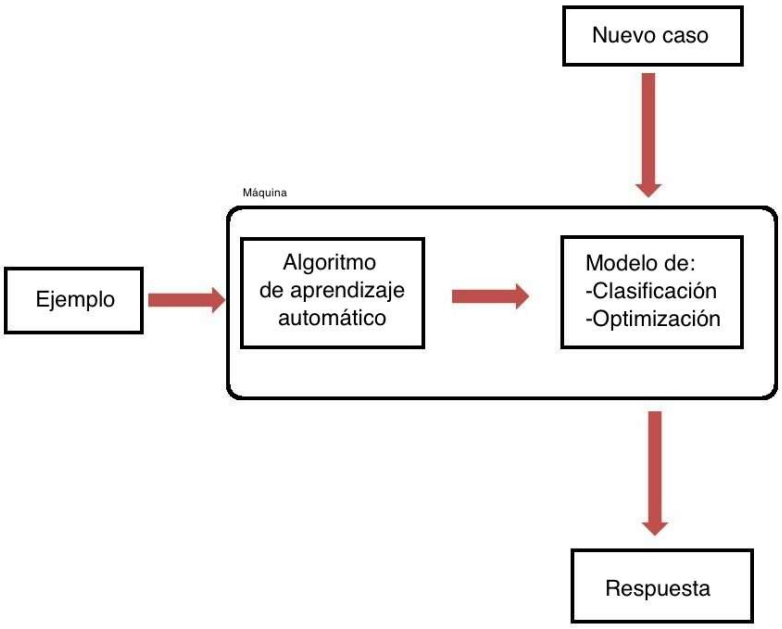

Figura 2.11: Idea general del AS.
Fuente: https://es.wikipedia.org/wiki/Aprendizaje_automático

El problema de la SR se formula como problema de AS, concretamente de regresión, ya que la idea es lograr una función que estime lo mejor posible una imagen LR a HR.

Aprendizaje no Supervisado

Aprendizaje no supervisado (del inglés Unsupervised Learning)[32] es un método de AA donde un modelo se ajusta sólo a las observaciones. Se distingue del AS por el hecho de que no hay un conocimiento a priori. Así, el aprendizaje no supervisado típicamente trata los objetos de entrada como un conjunto de variables aleatorias, siendo construido un modelo para el conjunto de datos.

El aprendizaje no supervisado suele tener lugar cuando no se dispone de datos etiquetados para el entrenamiento.

Aprendizaje por Refuerzo

Aprendizaje por refuerzo [33] o Aprendizaje reforzado (del inglés Reinforcement Learning), es un área del aprendizaje automático inspirada en la psicología conductista, cuya ocupa-

[32] https://es.wikipedia.org/wiki/Aprendizaje_no_supervisado
[33] https://es.wikipedia.org/wiki/Aprendizaje_por_refuerzo

ción es determinar qué acciones debe escoger un agente inteligente en un entorno dado con el fin de maximizar alguna noción de recompensa".

Existen dos elementos atómicos en el aprendizaje reforzado, que son el agente y el ambiente. El agente realiza acciones sobre el ambiente y el ambiente cambia su contexto o estado en base a esas acciones. Adicionalmente, el ambiente emite una señal de recompensa para cada par estado-acción. Esto se entiende mejor con la Figura 2.12.

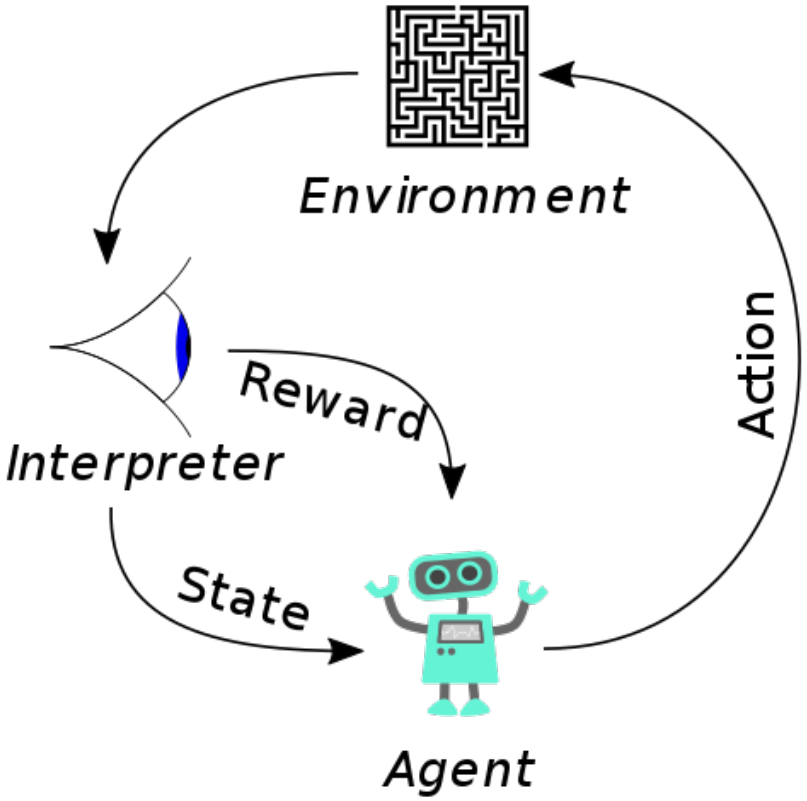

Figura 2.12: Encuadre típico de un escenario de aprendizaje por refuerzo: un agente se se encuentra en un estado (state) y realiza una acción (action) en un entorno (environment), a lo cual un intérprete (interpreter) reacciona otorgándole una recompensa (reward) al agente, y haciéndolo cambiar de estado

Fuente: https://es.wikipedia.org/wiki/Aprendizaje_por_refuerzo

Para elegir qué acción tomar en cada momento, el agente sigue lo que se llama una política, que no es más que una función con las entradas el estado y un conjunto de acciones posibles.

2.2.2. Redes Neuronales Artificiales

Las redes neuronales artificiales (NN, de sus siglas en inglés de Neural Networks), son un modelo de ML inspirado en las neuronas biológicas [34] y constituyen el paradigma de IA más popular hoy en día.

La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo unos valores de salida. Visualmente, se representan mediante grafoz [35], pero su definición se pueden formalizar de manera precisa (ver Definición 2.12).

***Definición* 2.12** *Las NN son proyecciones algebraicas de vectores de entrada x, de dimensión parametrizable, a vectores de salida y, también parametrizables, que se definen como:*

$$y = f(x), \textit{ donde } x \in R^a,\ y \in R^b;\ a,b \geq 0$$

Las NNs no son un invento nuevo, sino que lleva con nosotros desde hace décadas [36]. Pero no ha sido hasta hoy, con la mejora del hardware y de las propias técnicas de NN, que su uso ha crecido exponencialmente para reconocimiento de caracteres, de imágenes [37], predicciones bursátiles [38], traducción de idiomas [39], conducción autónoma [40] etcétera.

La neurona

La mayoría de comportamientos y estructuras avanzadas suelen estar constituidas por varias partes más simples trabajando conjuntamente y, como se ha dicho, en el caso de una NN, a cada una de estas partes se les llama neurona.

Con estas neuronas se pueden realizar modelos de regresión lineal [41] sencillos como pueden ser las puertas lógicas AND [42] y OR [43].

[34] https://es.wikipedia.org/wiki/Neurona
[35] https://es.wikipedia.org/wiki/Grafo
[36] https://es.wikipedia.org/wiki/Perceptr%C3%B3n
[37] https://medium.com/better-programming/handwriting-recognition-using-an-artificial-neural-network-78060d2a7963
[38] http://www.scielo.org.pe/scielo.php?script=sci_arttext&pid=S2077-18862016000200007
[39] https://www.deepl.com/translator
[40] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6539483/
[41] https://es.wikipedia.org/wiki/Regresi%C3%B3n_lineal
[42] https://es.wikipedia.org/wiki/Puerta_AND
[43] https://es.wikipedia.org/wiki/Puerta_OR

La neurona es la unidad básica de procesamiento en una NN. Al igual que las neuronas biológicas, se trata de una unidad que recibe información a través de unos estímulos externos (que serían las conexiones sinápticas [44]) y produce una salida en base a ellos. Visto de esa manera, una neurona no es más que una función matemática de orden n.

Concretamente, una neurona se compone de dos funciones, como puede verse en la Figura 2.15:

- La función de transferencia: La suma ponderada de las entradas con los pesos. Es en realidad una regresión lineal.
- La función de activación: Función no lineal que se aplica sobre la salida de la función de transferencia.

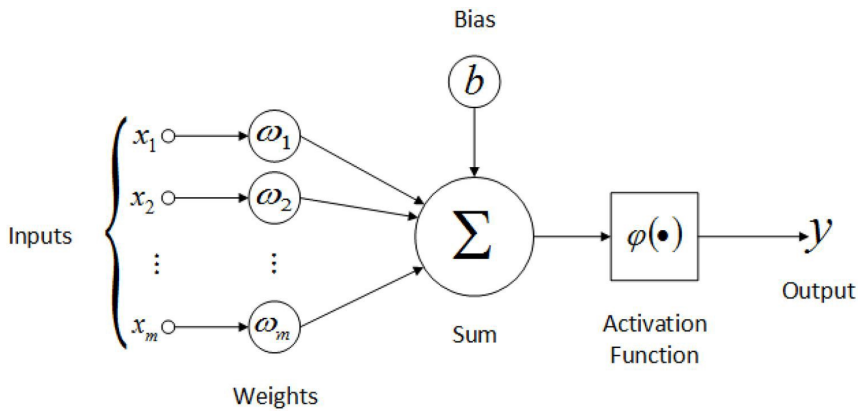

Figura 2.13: Función de activación sobre función de transferencia.
Fuente: https://medium.com/@ricardojmv85/aplicación-del-gradiente-en-redes-neuronales-78bff0d802d5

Las funciones de activación aportan no linealidad a la NN, necesaria porque sino sólo habría regresiones lineales y está matemáticamente probado que la suma de varias regresiones lineales es a su vez una regresión lineal: es decir, toda la NN podría colapsar en una sola.

Existen muchas funciones con ese objetivo, pero en el campo del ML las más utilizadas son la ReLU y la Sigmoide (Definiciones 2.13 y 2.14).

***Definición* 2.13** *La unidad lineal rectificada (ReLU, por sus siglas en inglés) [45], se comporta como una función lineal cuando la entrada es positiva y constante de 0 si la entrada*

[44] https://es.wikipedia.org/wiki/Sinapsis
[45] https://es.wikipedia.org/wiki/Rectificador_(redes_neuronales)

es negativa:

$$f(x) = max(0,x)$$

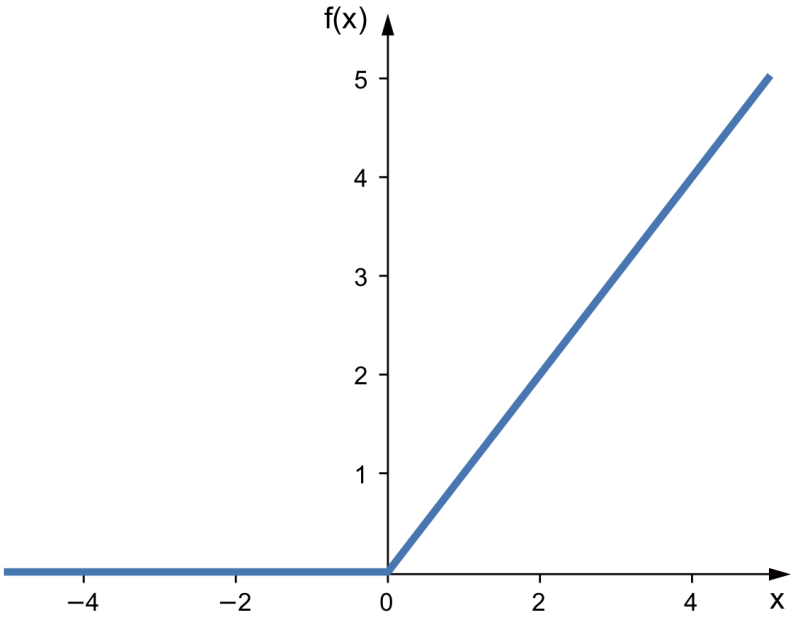

Figura 2.14: Función ReLU.
Fuente: https://medium.com/@ricardojmv85/aplicación-del-gradiente-en-redes-neuronales-78bff0d802d5

***Definición* 2.14** *La función sigmoide* [46] *hace que los valores muy altos se saturen en 1 y los valores muy bajos se saturen en 0:*

$$f(x) = \frac{1}{1+e^{-x}}$$

Por otro lado, las neuronas pueden organizarse en capas, de manera que las neuronas pertenecientes a una capa reciben la información procesada de la capa anterior, es decir de las neuronas pertenecientes a la capa anterior. A su vez, estas capas enviarán sus cálculos (regresión lineal+función de activación) a la siguiente, como puede verse en la Figura 2.16.

A la primera capa se le denomina capa de entrada, a las capas intermedias se les llama capas ocultas y a la última capa se le denomina capa de salida, que es la que devuelve la clasificación o regresión definitiva.

[46] https://es.wikipedia.org/wiki/Funci%C3%B3n_sigmoide

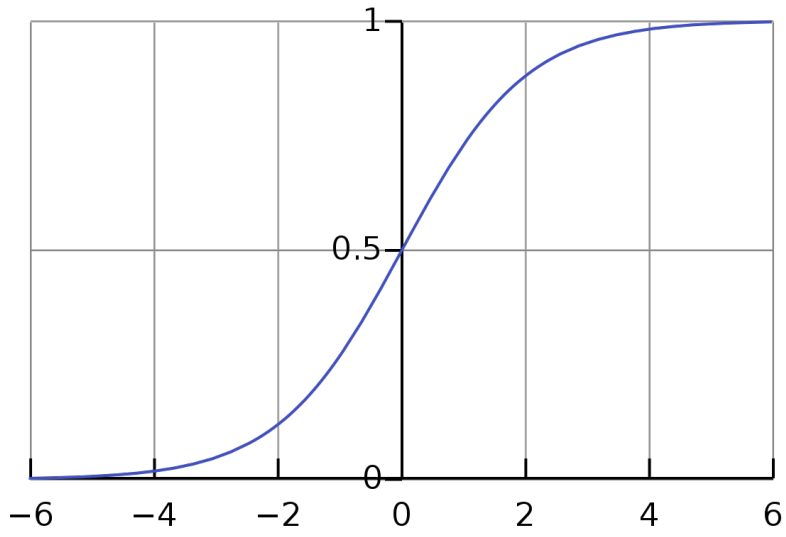

Figura 2.15: Función sigmoide.
Fuente: https://en.wikipedia.org/wiki/Sigmoid_function

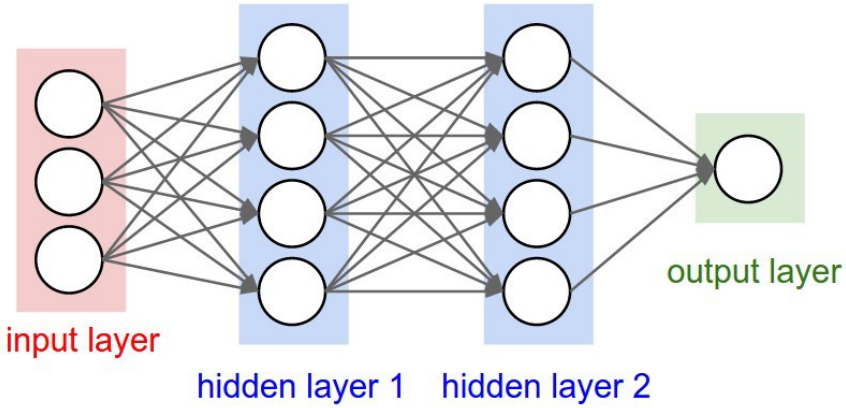

Figura 2.16: Una NN organizada por capas: se ven una capa de entrada (input layer), dos capas ocultas (hidden layer) y una capa de salida (output layer)
Fuente: https://medium.com/@ricardojmv85/aplicación-del-gradiente-en-redes-neuronales-78bff0d802d5

Mediante estas capas logramos formar una red y generar conocimiento jerarquizado. La secuencia de neuronas permite información cada vez más compleja y abstracta.

Las redes con una gran cantidad de capas (de capas ocultas) son las que dan nombre al DL.

Perceptrón multicapa

El perceptrón multicapa [47] es una NN formada por múltiples capas, de tal manera que tiene capacidad para resolver problemas que no son linealmente separables, lo cual es el mayor problema del perceptrón simple (NN de una única neurona).

El perceptrón multicapa puede estar totalmente (del inglés fully connected [48]) o localmente conectado. En el primer caso cada salida de una neurona de la capa i es entrada de todas las neuronas de la capa $i+1$, mientras que en el segundo cada neurona de la capa i es entrada de una serie de neuronas (región) de la capa $i+1$.

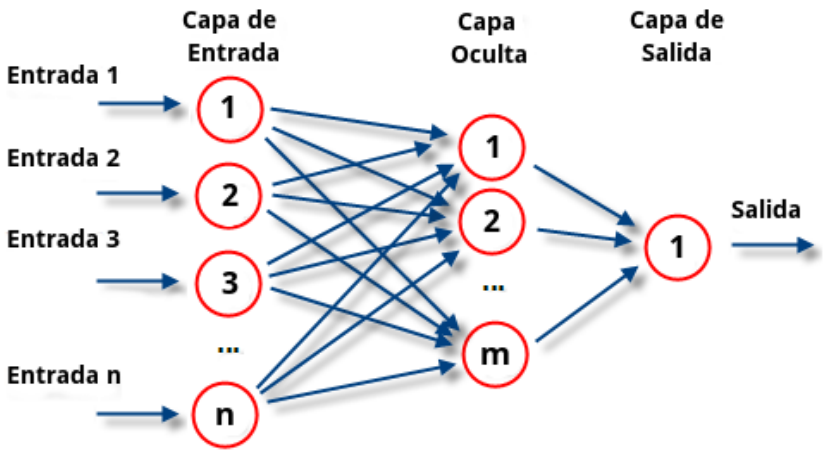

Figura 2.17: La topología de NN más común: el perceptrón multicapa fully connected. Fuente: https://es.wikipedia.org/wiki/Perceptrón_multicapa

Redes neuronales convolucionales

Las redes neuronales convolucionales (o CNN, del inglés Convolutional Neural Network) surgen por la necesidad de optimizar el aprendizaje automático en problemas de imágenes.

En las redes neuronales convolucionales no cada píxel está conectado con todas las neuronas de la primera capa oculta, sino que hay conjuntos de píxeles que comparten una sola neurona de la primera capa oculta.

Esto se logra mediante la combinación de dos conceptos: el convolving y el pooling (ver Definiciones 2.15 y 2.16).

[47] https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa
[48] https://iq.opengenus.org/fully-connected-layer/

***Definición* 2.15** *En matemáticas, llamamos convolving (o convolución)* [49] *a un operador que transforma dos funciones f y g en una tercera, que indica de cierta manera la magnitud en la que se superponen. En las CNNs, la capa de convolución es su elemento básico. Los parámetros de la capa consisten en un conjunto de filtros (o núcleos* [50]*, o kernels) que son aprendidos por la misma durante el entrenamiento. Cada filtro se convoluciona con el elemento de entrada, calculando el producto escalar entre el filtro y la entrada, produciendo un mapa.*

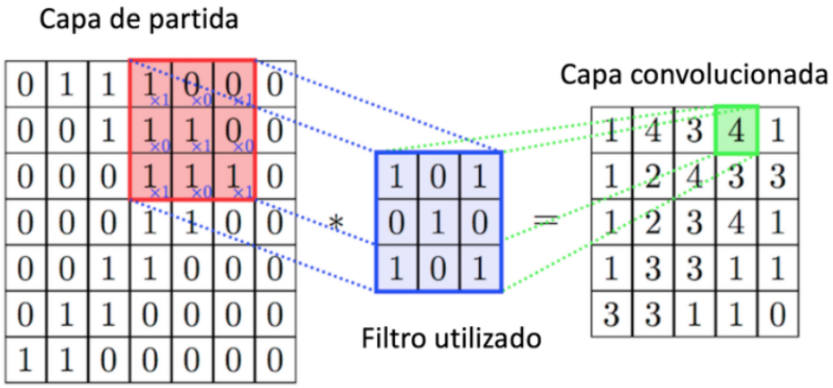

Figura 2.18: Convolución de imagen con filtro.
Fuente: https://www.diegocalvo.es/red-neuronal-convolucional/

***Definición* 2.16** *El pooling* [51]*, es un tipo de submuestreo (subsampling en inglés); es decir, de reducción el tamaño de esas matrices. Esto se hace para que las capas ocultas no tengan que procesar tantas entradas. La forma de reducir parámetros se realiza mediante la extracción de estadísticas como el promedio (llamado average pooling) o el máximo (llamado max pooling, ver Figura 2.19) de una región fija del mapa de características.*

El proceso de convolution-pooling se aplica iterativamente hasta el final, donde se conecta la salida con una NN tradicional que hará clasificación o regresión, como puede verse en la Figura 2.20. Para ello, se aplana la salida de la red para que sea la entrada de la nueva NN.

Esto es una gran ventaja al momento del aprendizaje pues como vimos cada kernel es de un tamaño reducido.

[49] https://es.wikipedia.org/wiki/Convoluci%C3%B3n
[50] https://es.wikipedia.org/wiki/N%C3%BAcleo_(procesamiento_digital_de_im%C3%A1genes)
[51] https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

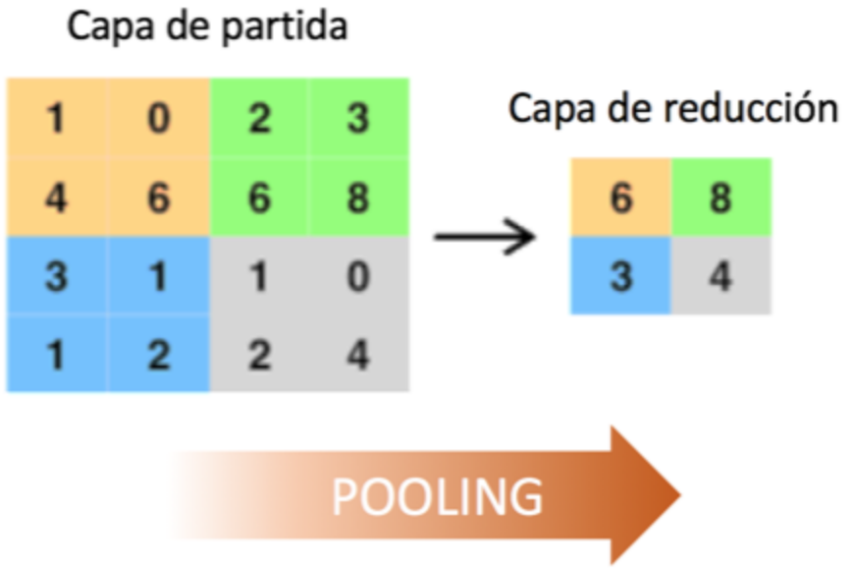

Figura 2.19: Reducción de una imagen por max pooling.
Fuente: https://www.diegocalvo.es/red-neuronal-convolucional/

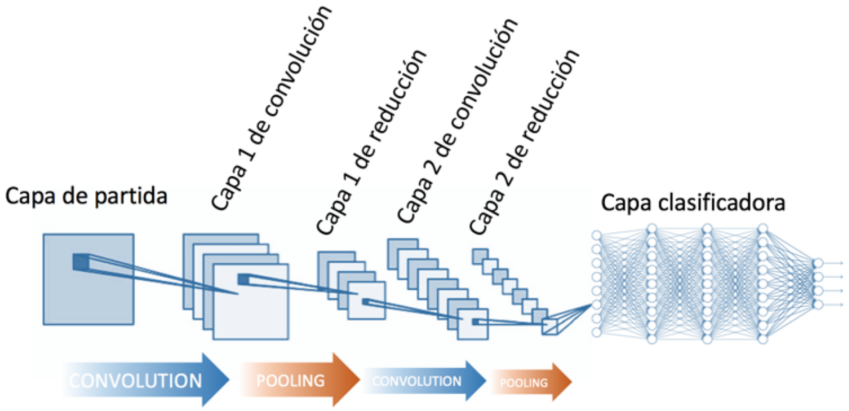

Figura 2.20: Una CNN completa.
Fuente: https://www.diegocalvo.es/red-neuronal-convolucional/

Existen numerosas CNNs en el estado del arte, como pueden ser:

- LeNet [52].
- AlexNet (bueno para empezar) [53].
- VGGNet [54].

[52] https://en.wikipedia.org/wiki/LeNet
[53] https://en.wikipedia.org/wiki/AlexNet
[54] https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

- GoogleLeNet [55].
- ResNet (particularmente interesante para este trabajo) [Kaiming He, 2015].
- ZFNet [56].

Redes residuales

La intuición detrás de las capas CNN es que aprenden progresivamente características más complejas. Por ejemplo, la primera capa aprende los bordes, la segunda capa aprende las formas, la tercera capa aprende los objetos, la cuarta capa aprende los ojos, etcétera. Sin embargo, [Kaiming He, 2015] mostró empíricamente que hay un umbral máximo de profundidad con el modelo tradicional CNN.

Para evitar ese problema, se introducen las redes neuronales residuales (ResNet, por sus siglas en inglés Residual Network) en [Kaiming He, 2015]. Su modificación más importante con respecto a una CNN tradicional es la llamada skip connection [57] o mapeo de identidad (ver Figura 2.21). Este mapeo de identidad no tiene ningún parámetro y solo está ahí para agregar la salida de la capa anterior a la capa siguiente.

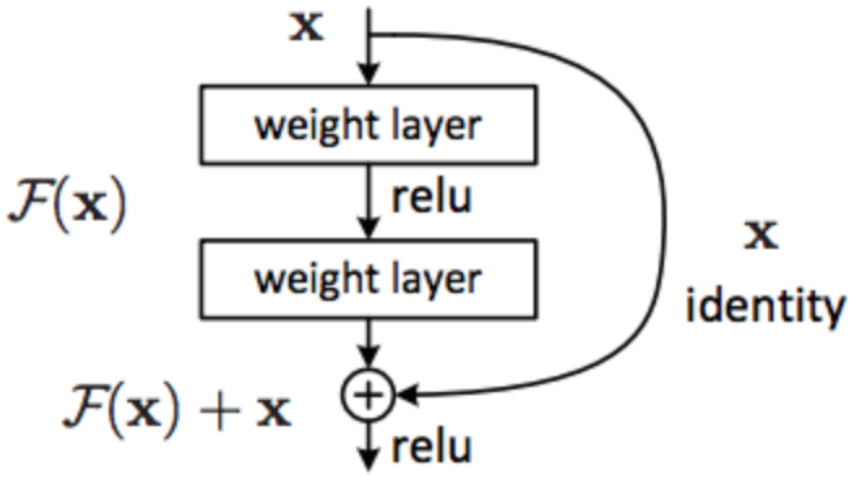

Figura 2.21: Ejemplo de bloque residual.
Fuente:
https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4

[55] https://es.mathworks.com/help/deeplearning/ref/googlenet.html
[56] shorturl.at/jrty1
[57] https://kharshit.github.io/blog/2018/09/07/skip-connections-and-residual-blocks

Sin embargo, a veces x y $F(x)$ no tendrán la misma dimensión. Recordemos que una operación de convolución generalmente reduce la resolución espacial de una imagen; por ejemplo, una convolución de 3×3 en una imagen de 32×32 da como resultado una imagen de 30×30. El mapeo de identidad se multiplica por una proyección lineal W para expandir los canales de acceso directo para que coincida con el residual, lo cual permite que la entrada x y $F(x)$ se combinen como entrada a la siguiente capa. Esto se ve en la Definición 2.17.

***Definición* 2.17** *El bloque residual se define matemáticamente de la siguiente manera, si se le aplica la proyección lineal al mapeo de identidad:*

$$y = F(x, W_i) + x$$

Esto permite que la entrada x y $F(x)$ se combinen como entrada a la siguiente capa, como se puede ver en la Figura 2.22.

U-Net

La U-Net [58] [59] es una arquitectura para CNNs ampliamente usada en el contexto de la bioinformática, específicamente en segmentación de imágenes biomédicas.

Ha sido, desde el primer momento del trabajo, la arquitectura sobre la que se han hecho los experimentos y, en caso de haberla modificado, nunca ha dejado de ser una U-Net, sino versiones de la misma.

La U-Net es una CNN y como tal, enfoca su tarea en la clasificación de imágenes, donde la entrada es una imagen y la salida es una etiqueta, pero en casos biomédicos, se nos exige no sólo distinguir si hay o no una enfermedad, sino también localizar el área de anormalidad.

La razón por la que es capaz de localizar y distinguir bordes es haciendo una clasificación en cada píxel, por lo que la entrada y la salida comparten el mismo tamaño. Por ejemplo, para una imagen de entrada de tamaño 2×2, la salida tendrá el mismo tamaño de 2×2.

La arquitectura tiene la forma que se muestra en la Figura 2.23:

[58] https://en.wikipedia.org/wiki/U-Net
[59] https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

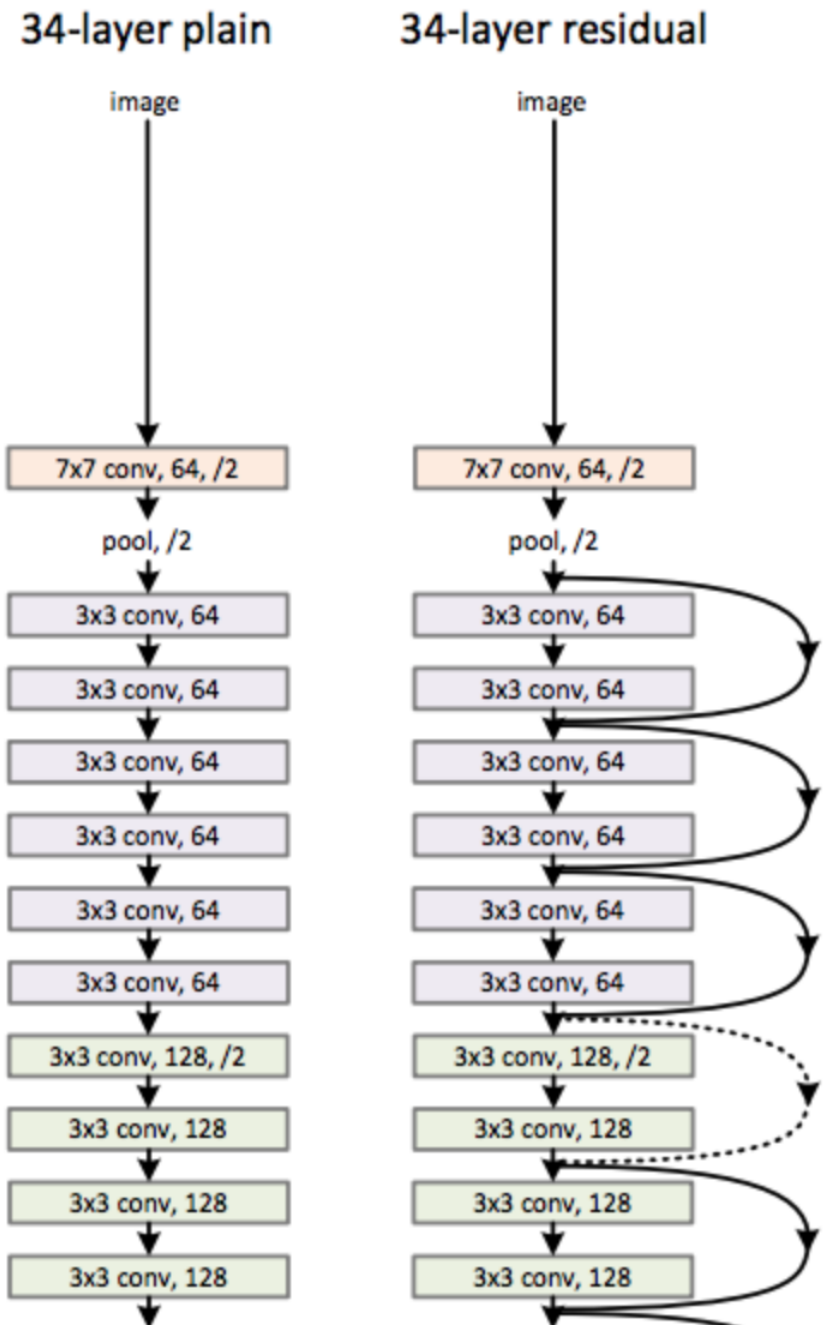

Figura 2.22: Una CNN tradicional (izquierda) frente a una Resnet de 34 capas (derecha), también conocida como ResNet34.

Fuente:
https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4

A primera vista, tiene forma de U. La arquitectura es simétrica y consta de dos partes principales: la parte izquierda se llama ruta de contracción (del inglés contracting path), que está constituida por el proceso convolucional general; la parte derecha es la ruta ex-

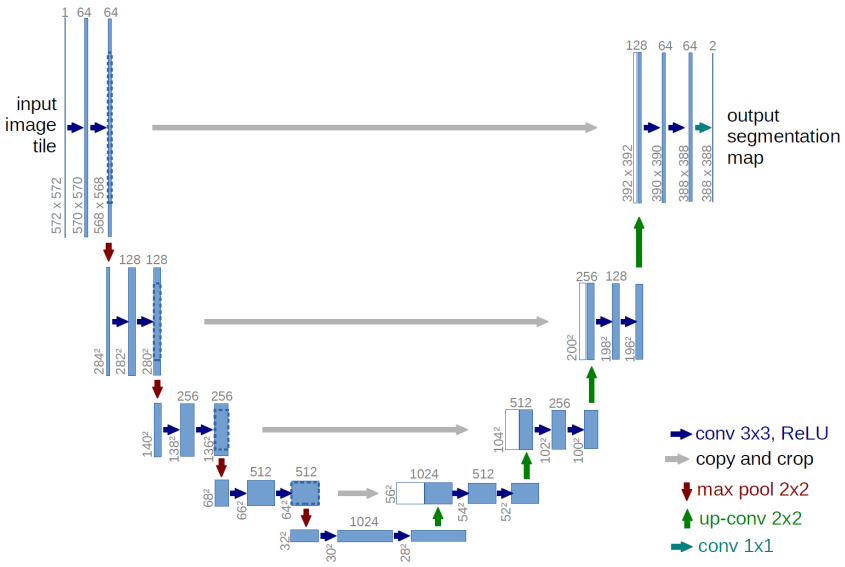

Figura 2.23: Arquitectura de la U-Net.
Fuente: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

pansiva (del inglés expansive path), que está constituido por capas convolucionales 2D transpuestas.

Es una red fully-convolutional del tipo codificador y decodificador (del inglés encoder-decoder) [60], donde gradualmente el encoder submuestrea la imagen de entrada y el encoder lo escala a su tamaño original. Además, tiene skip connections y conexiones de feature maps de izquierda a derecha.

Como se puede observar, no es una arquitectura difícil de visualizar, y tampoco de implementar [61], por lo que se ha propuesto desde el principio como una arquitectura digerible para empezar a trabajar con ella.

Aprendizaje en redes neuronales

El estudio formal y matemático de cómo aprende una red neuronal es realmente extenso [62], e incluye el estudio de las derivadas parciales [63] (regla de la cadena [64]...), pero en este apartado solamente desarrollaremos una pequeña intuición de su funcionamiento.

[60] https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346
[61] https://github.com/zhixuhao/unet
[62] https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd
[63] https://es.wikipedia.org/wiki/Derivada_parcial
[64] https://es.wikipedia.org/wiki/Regla_de_la_cadena

Según Jordi Torres en [65], entrenar una red neuronal es, en realidad, modificar iterativamente los valores de los pesos asociados a las neuronas. Podemos ver este proceso de aprendizaje en una red neuronal como un proceso iterativo de ir y venir por las capas de neuronas.

El "ir" propagando hacia adelante es el forward propagation y el "venir" retropropagando información en la red les la clave de este método: el backpropagation.

La primera fase, forward propagation [66], se da cuando se expone la red a los datos de entrenamiento y estos cruzan toda la red neuronal para ser calculadas sus predicciones. Es decir, pasar los datos de entrada a través de la red de tal manera que todas las neuronas apliquen su transformación a la información que reciben de las neuronas de la capa anterior y la envíen a las neuronas de la capa siguiente. Cuando los datos hayan cruzado todas las capas, y todas sus neuronas han realizado sus cálculos, se llegará a la capa final con un resultado de predicción para aquellos ejemplos de entrada.

A continuación usaremos una función de pérdida (o función de loss) [67] para medir cómo de bueno/malo fue nuestro resultado de la predicción en relación con el resultado correcto (recordemos que estamos en un entorno de aprendizaje supervisado y disponemos de la etiqueta que nos indica el valor esperado); esa función de loss es, en nuestro caso, MSE [68]. Idealmente, queremos que nuestro coste sea cero, es decir, sin divergencia entre valor estimado y el esperado. Por eso a medida que se entrena el modelo se irán ajustando los pesos de las interconexiones de las neuronas de manera automática hasta obtener buenas predicciones.

Una vez se tiene calculado el loss, se propaga hacia atrás esta información. De ahí su nombre, retropropagación, en inglés backpropagation. Partiendo de la capa de salida, esa información de loss se propaga hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo las neuronas de la capa oculta sólo reciben una fracción de la señal total de la loss, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de loss que describa su contribución relativa al loss total. Visualmente, podemos resumir lo que hemos contado con el esquema visual de la Figura 2.24.

[65] https://torres.ai/deeplearning/

[66] https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250

[67] https://es.wikipedia.org/wiki/Funci%C3%B3n_de_p%C3%A9rdida

[68] https://es.wikipedia.org/wiki/Error_cuadr%C3%A1tico_medio

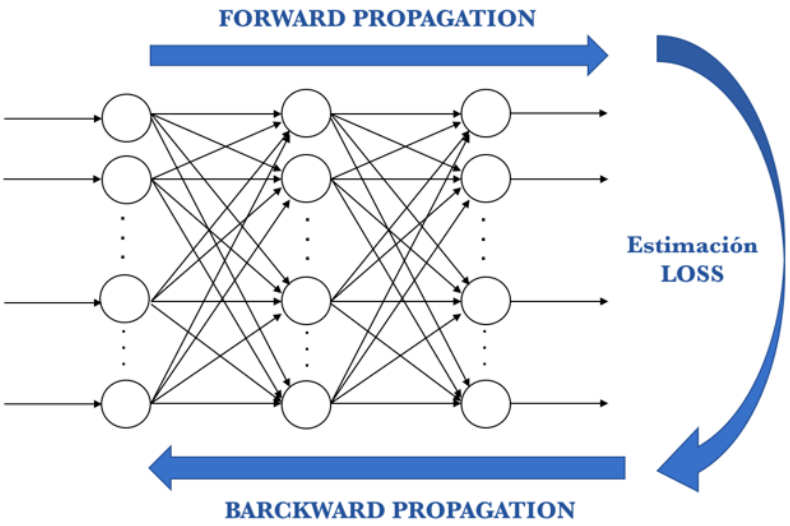

Figura 2.24: Esquema de backpropagation.
Fuente: https://torres.ai/deep-learning-inteligencia-artificial-keras/

Ahora que ya hemos propagado hacia atrás esta información, podemos ajustar los pesos de las conexiones entre neuronas. Lo que estamos haciendo es que el loss se aproxime lo más posible a cero la próxima vez que volvamos usar la red para una predicción. Para ello usaremos una técnica llamada descenso del gradiente (del inglés gradient descent [69]). Esta técnica va cambiando los pesos en pequeños incrementos con la ayuda del cálculo de la derivada (o gradiente) de la función de loss, cosa que nos permite ver en qué dirección "descender" hacia el mínimo global; esto lo va haciendo en general en lotes de datos (batches) en las sucesivas iteraciones (epochs) del conjunto de todos los datos que le pasamos a la red en cada iteración [70].

Para aprender, las CNN también utilizan backpropagation, pero lo que se hace es ajustar el valor de los pesos de los distintos kernels, es decir, lo que la red aprende es a ver qué mapas de características quiere obtener.

[69] https://en.wikipedia.org/wiki/Gradient_descent
[70] https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

3. CAPÍTULO

Estado del arte

Una parte esencial del TFG ha sido el estudio riguroso y profundo del estado del arte [1] en los campos de la SR y de las métricas de evaluación de la calidad de imágenes. Se introducen aquí los artículos más destacables, tanto de la SR en general como los del dominio específico de la SR en imagen de microscopía, junto con sus mayores contribuciones y las ideas más importantes que se han recopilado de ellos para desarrollar la investigación.

3.1. Métricas

Antes de estudiar en profundidad todo el estado del arte relacionado con las técnicas de SR (ver Sección 3.2 y 3.3), las métricas para evaluación de los modelos han merecido un estudio aparte.

Existen diferentes métricas para medir la calidad de una imagen (IQA, del inglés image quality assessment) [2], las cuales se dividen en métodos subjetivos y computacionales (u objetivos). Los métodos subjetivos tienden a ser caros e inviables, por lo que los métodos que más se utilizan son los objetivos. Sin embargo, estos métodos no capturan la percepción visual humana perfectamente, por lo que puede haber grandes diferencias entre sí.

Las métricas objetivas se dividen en los siguientes tipos:

[1] https://es.wikipedia.org/wiki/Estado_del_arte
[2] https://en.wikipedia.org/wiki/Image_quality#Image_quality_assessment_methods

- Métodos de referencia completa (full-reference methods), que evalúan utilizando imágenes de referencia.
- Métodos de referencia reducida (reduced-reference methods), que se basan en las características que se extraen de las imágenes.
- Métodos sin referencia (no-reference methods), sin imágenes de referencia. También se le llama IQA ciega (blind IQA).

3.1.1. Métricas más utilizadas

Por encima del resto de métricas, hay dos métricas computacionales de referencia completa que se utilizan de manera estándar a la hora de evaluar la calidad de las imágenes: PSNR y SSIM.

Es convención utilizar estas dos en cualquier artículo, por mucho que se añadan otras métricas también, puesto que son el lenguaje más general de evaluación. Sin embargo, ambas tienen tanto ventajas como limitaciones, y parten de ideas y propiedades matemáticas distintas.

Ambas funciones utilizan los mismos parámetros de entrada: dos imágenes de mismas dimensiones $N \times N$, donde una es la imagen referencia y la otra es la imagen distorsionada. Por tanto, ambas métricas se pueden comparar.

PSNR

Significa Proporción máxima de señal a ruido (del inglés Peak Signal to Noise Ratio) [3], que se refiere al hecho de medir el valor máximo que puede tener una señal (un píxel, en este caso) y el ruido que le afecta.

Se define formalmente en la Definición 3.1.

***Definición* 3.1** *Siendo x e y la imagen predicha y la imagen objetivo, respectivamente, se verifica que:*

$$PSNR(x,y) = 10\log_{10}\left(\frac{L^2}{MSE(x,y)}\right)$$

[3] https://es.wikipedia.org/wiki/PSNR

Donde L es el valor máximo que puede tener un bit, que es generalmente 255 debido a que se utiliza representación de 8 bits.

Como se puede ver, PSNR mide de alguna manera el inverso del MSE, por lo que cuanto mayor es el MSE menor será PSNR y viceversa. Por tanto, como PSNR sólo está relacionado con el MSE a nivel de píxel, no evalúa las imágenes igual que la percepción humana y no evalúa correctamente (de la misma manera que los humanos) la calidad de una imagen.

No obstante, debido a que no existen métricas perfectas y que la mayoría de trabajos de SR han utilizado ya PSNR, ésta sigue siendo la métrica más utilizada con diferencia.

SSIM

De nombre Índice de similaridad estructural (SSIM, del inglés Structural Similarity Index) [4] [Z. Wang and Simoncelli, 2004], mide cómo de similares son estructuralmente dos imágenes.

Su idea es la siguiente: el sistema visual humano (HSV de Human Visual System) [5] está altamente adaptado para extraer estructuras [Z. Wang and Lu, 2002] y así distinguir objetos, por lo que SSIM pretende medir la similitud estructural entre dos imágenes.

Su origen es, precisamente, mejorar los problemas de PSNR, puesto que este se centraba solamente en los valores individuales de los píxeles, mientras que SSIM pretende medir de alguna manera la dependencia entre los píxeles más cercanos entre sí, que es lo que sucede en la realidad, ya que esas dependencias conllevan una información visual mucho mayor, en forma de estructuras de objetos [H. R. Sheikh and Bovik, 2006] [Wang and Bovik, 2009].

La métrica SSIM realiza numerosos cálculos en ambas imágenes de entrada, en los que calcula sus índices de luminosidad, de contraste y de estructura, como se puede ver en la Figura 3.1, donde x e y son las señales de entrada (en este caso, dos imágenes).

Concretamente, teniendo en cuenta los siguientes parámetros:

- μ_x es la media de x.
- μ_y es la media de y.

[4] https://en.wikipedia.org/wiki/Structural_similarity
[5] https://es.wikipedia.org/wiki/Sistema_visual_humano

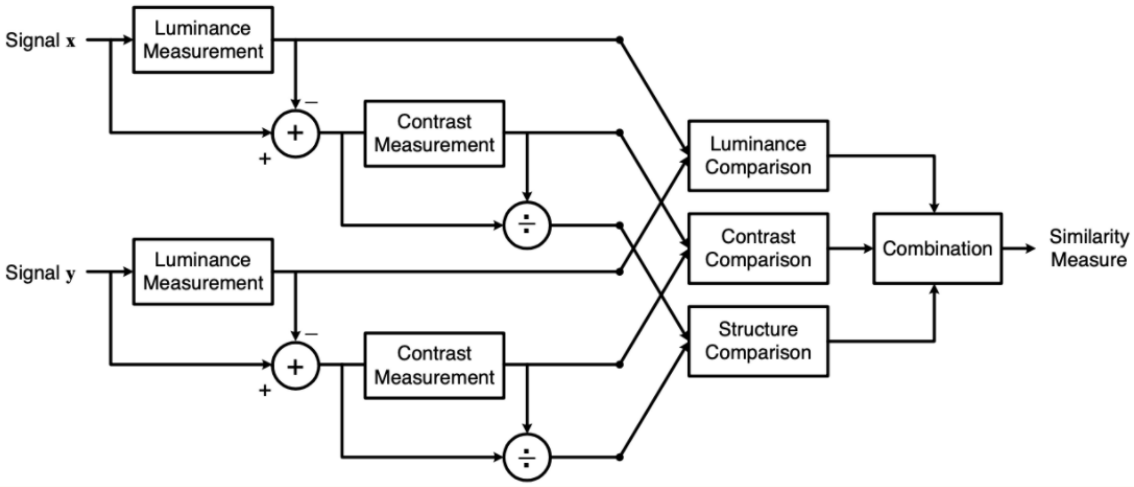

Figura 3.1: Funcionamiento de cálculo de SSIM. Las propiedades de luminosidad y contraste se calculan dos veces (con las dos imágenes originales y con las resultantes de los cálculos) y la estructura solamente con las resultantes; y después se combinan los resultados.
Fuente: [Z. Wang and Simoncelli, 2004]

- σ_x^2 es la varianza [6] de x.
- σ_y^2 es la varianza de y.
- σ_{xy} es la covarianza [7] de x e y.
- $c1 = (k_1 * L)^2$ es una variable para estabilizar la división.
- $c2 = (k_2 * L)^2$ es otra variable para estabilizar la división.
- $c3 = c2/2$ es otra variable para estabilizar la división.
- $L = 2^{bitsporpixel} - 1$ es el máximo valor de los píxeles.
- $k_1 = 0{,}001$ y $k_2 = 0{,}003$ por defecto.

Así se calculan las tres componentes de la función (ver Definición 3.2, 3.3 y 3.4)

***Definición* 3.2** *Sea L una función que pretende calcular la diferencia de luminosidad entre dos imágenes:*

$$L(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2\mu_y^2 + c_1}$$

[6] https://es.wikipedia.org/wiki/Varianza
[7] https://es.wikipedia.org/wiki/Covarianza

***Definición* 3.3** *Sea C una función que pretende calcular la diferencia de contraste entre dos imágenes:*

$$C(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2\sigma_y^2 + c_2}$$

***Definición* 3.4** *Sea S una función que pretende calcular la diferencia de estructura entre dos imágenes:*

$$S(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

Como SSIM es la combinación de estos factores, matemáticamente es su producto junto con unos pesos opcionales (ver Definición 3.5).

***Definición* 3.5** *Sea SSIM una función que pretende calcular la combinación que cuantifica la diferencia entre dos imágenes, donde α, β y γ son enteros positivos:*

$$SSIM(x,y) = [L(x,y)^\alpha * C(x,y)^\beta * S(x,y)^\gamma]$$

Si los pesos fueran iguales a 1, entonces la fórmula se puede reducir a su caso particular, conocido como *universal image quality index (ver Definición 3.6)* [Z. Wang, 2002] [Z. Wang and Simoncelli, 2004].

***Definición* 3.6** *Sea el caso particular de SSIM con $\alpha = \beta = \gamma = 1$:*

$$SSIMParticular(x,y) = [L(x,y)^1 * C(x,y)^1 * S(x,y)^1] = \frac{(2\mu_x\mu_y + c_1)(\mu_x^2\mu_y^2 + c_1)}{(2\sigma_{xy} + c_2)(\sigma_x^2\sigma_y^2 + c_2)}$$

En su artículo original se menciona que se desea que SSIM satisfaga las propiedades simétrica, no-negativa y máximo único, lo cual se reafirma en el artículo [Wang, 2003] (en el que se introduce una variante de SSIM llamada SSIM multiescala o MS-SSIM por sus siglas en inglés Multi-Scale SSIM), asegurando que para el caso particular de SSIM se cumplen la Definición 3.7, 3.8 y 3.9.

***Definición* 3.7** *Simetría: El orden de los parámetros de entrada no altera su resultado, es decir, no importa si comparamos la imagen x con la imagen y, o la imagen y con x.*

SSIMParticular(x,y)=SSIMParticular(y,x)

***Definición* 3.8** *Cota superior: Nunca se puede obtener un valor mayor que 1, es decir, el condominio está limitado superiormente por 1.*

$$SSIMParticular(x,y) \leq 1$$

***Definición* 3.9** *Máximo único: El codominio más alto, 1, sólo se puede obtener si los dos parámetros de entrada son el mismo término, es decir, si y sólo si las dos imágenes son estrictamente iguales.*

$$SSIMParticular(x,y) = 1 \Longleftrightarrow x = y$$

Por otro lado, está comprobado que SSIM no satisface la desigualdad triangular, por lo que no es una métrica de distancia, pero sí bajo ciertas circunstancias. Esta y muchas propiedades matemáticas más se estudian en [Brunet et al., 2012].

PSNR vs SSIM

La principal diferencia entre ambas métricas es la más conocida: PSNR mide los píxeles individualmente y SSIM trata de obtener una visión más global. Pero esto no significa que una sea mejor que la otra, sino que se centran en diferentes objetivos.

Experimentalmente, enfocándose en casos concretos, es donde más se puede observar la diferencia entre ambas funciones.

En [Kotevski and Mitrevski, 2010] se estudian las diferencias de ambas métricas en instantáneas de vídeos, comparando cómo varían los resultados de PSNR y SSIM en función de cambios en parámetros como brillo o suma de ruido gaussiano.

Las conclusiones son que PSNR no es recomendable para evaluar la calidad en vídeo, puesto que es muy sensible a cambios en brillo o contraste, mientras que SSIM se comporta mucho más parecido al HSV, pero su problema es que es demasiado insensible ante estos cambios.

Por otro lado, se han estudiado las limitaciones expresamente de SSIM para el estudio de la calidad de imágenes biomédicas [Pambrun and Noumeir, 2015] como subestimación de distorsión en esquinas e insensibilidad en regiones de alta intensidad. Por tanto, se recomienda el MSE para este tipo de imágenes.

Pese a parecer muy diferentes, recientes estudios revelan que la diferencia real (matemática) entre ambas métricas no es tal. Las primeras sospechas vinieron cuando se notó una relación entre el MSE (y por tanto, PSNR) y SSIM en [Dosselmann and Yang, 2005]. Finalmente, en [Horé and Ziou, 2010] se propone que dado un resultado con una métrica, se puede obtener el resultado con la otra, como se muestra en la Definición 3.10.

***Definición* 3.10** *Se define PSNR en función de SSIM de la siguiente manera:*

$$PSNR(f,g) = 10\log[\frac{2\sigma_{x,y}(l(f,g) - SSIM(f,g))}{L^2 SSIM(f,g)} + (\frac{\mu_f + \mu_g}{L})^2]$$

Donde f y g son las dos imágenes, la función l es la que calcula la disparidad de luminosidad entre dos imágenes y L es la precisión en bits (generalmente, 255).

De hecho, bajo ciertas asunciones, esta equivalencia puede simplificarse, como en la Definición 3.11.

***Definición* 3.11** *Sea el caso l(f,g)=1* $\Longleftrightarrow \mu_f = \mu_g$:

$$PSNR(f,g) = 10\log[\frac{2\sigma_{x,y}(l(f,g) - SSIM(f,g))}{L^2 SSIM(f,g)} + (\frac{\mu_f + \mu_g}{L})^2]$$

Si se grafica la Definición 3.11 variando la covarianza σ_{fg} en el intervalo $[0,255]$, se puede ver que ambas curvas tienen la misma forma (ver la Figura 3.2).

La conclusión no deja lugar a dudas: se pueden predecir los valores de SSIM a partir de los de PSNR, y viceversa. Ambas métricas difieren principalmente en su grado de sensibilidad ante degradaciones de la imagen.

3.1.2. Métricas menos utilizas

Además de esas dos métricas, existen muchas otras y cada día se proponen nuevas, como pueden ser FSIM [Zhang et al., 2011] o NIQUE [Mittal et al., 2013]. Pero de todas ellas, para el trabajo es relevante comentar dos.

MOS

El testeo MOS es el método subjetivo más utilizado, en el cual se pide a los evaluadores humanos que califiquen la calidad de una imagen de una nota mínima a una máxima para

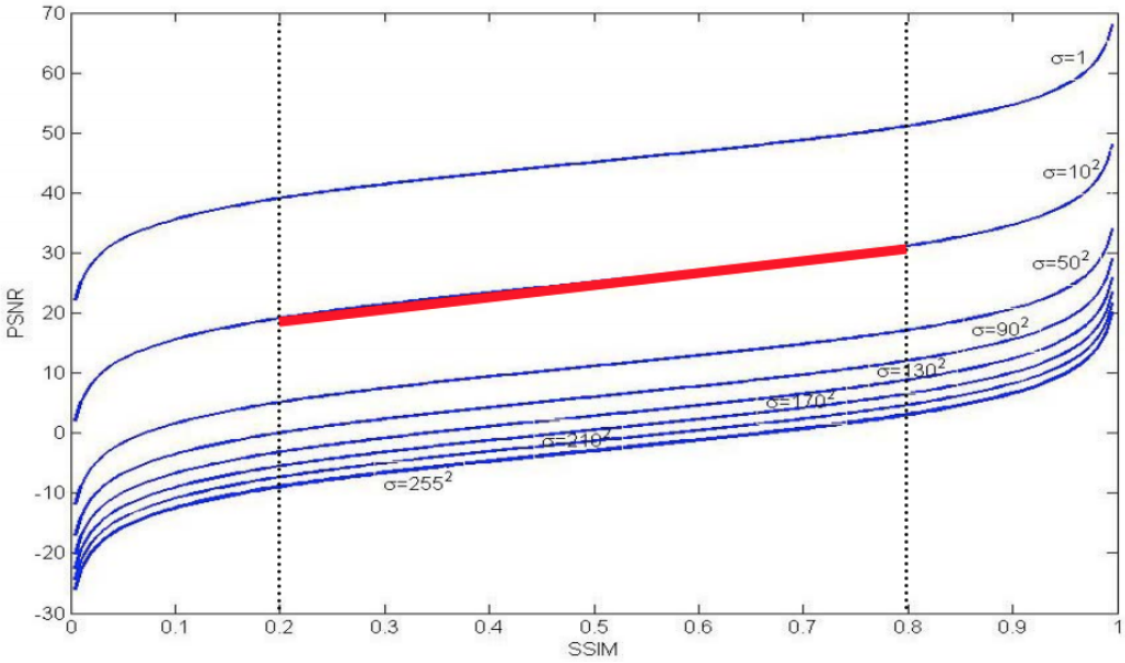

Figura 3.2: Evolución de las gráficas de PSNR y SSIM en función de la covarianza
Fuente: [Horé and Ziou, 2010]

finalmente hacer su media.

Este método conlleva problemas como que haya variaciones de criterio entre personas, pero existen modelos SR que obtienen resultados más pobres en métricas objetivas y muy buenas en métricas como MOS.

Asimismo, este método puede combinarse con el llamado orientado a tareas (task-based evaluation en inglés). Puesto que la SR puede ayudar en otros quehaceres de la VC como reconocimiento de objetos [Lim et al., 2017], [Zhang et al., 2018b], el reconocimiento de rostros [Fookes et al., 2012], [Zhang et al., 2018a] o el alineamiento de rostros [Bulat and Tzimiropoulos, 2017], [Chen et al., 2017], una opción para medir el resultado puede plantearse como quedarse con la media de efectividad en esos quehaceres concretos.

MS-SSIM

Una de las muchas variantes de la métrica SSIM es la SSIM de varias escalas (MS-SSIM), que se propone en el artículo [Wang, 2003].

Funciona basándose en que la perceptibilidad de los detalles de una imagen depende de tres factores:

- La densidad de muestreo de la señal de la imagen.
- La distancia del plano de la imagen con respecto al observador.
- La capacidad perceptiva del HSV del observador.

La evaluación subjetiva de una imagen dada varía cuando estos tres factores varían. Y un método de escala única como SSIM puede ser apropiado sólo para configuraciones específicas de esos factores.

En cambio, un método de múltiples escalas es una forma conveniente de incorporar detalles de imagen a diferentes resoluciones. En concreto, se propone un método SSIM multiescala (MS-SSIM) para la evaluación de calidad de imagen, cuyo diagrama del sistema se ilustra en la Figura 3.3.

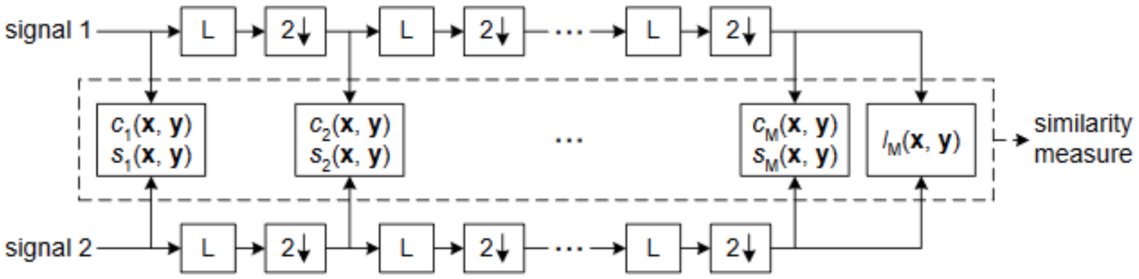

Figura 3.3: Esquema del método MS-SSIM; al igual que en SSIM 3.1, se calculan iterativamente las disparidades en luminosidad y contraste, y la estructura se calcula solamente al final.
Fuente: [Wang, 2003]

Donde L es un filtro de pase bajo (low pass filter) [8] y ↓ es un submuestreo (downsample) por factor de 2.

Como se ve en la Figura 3.3, el funcionamiento del algoritmo MS-SSIM es el siguiente:

- Se toman la imagen objetivo y la imagen predicha por el modelo como las dos señales de inicio.
- Se les aplica iterativamente un filtro de pase bajo y un submuestreo de 2.
- Se consiguen las comparaciones de contraste y de estructura entre las dos imágenes originales, y a esto se le llama escala 1.

[8] https://es.wikipedia.org/wiki/Filtro_paso_bajo

- Se realizan $M-1$ iteraciones de filtro+submuestreo, donde tras cada una se consiguen las comparaciones de contraste y de estructura entre las imágenes. Por tanto, en la iteración $(j-1)$-ésima tenemos la escala j-ésima con valores c_j y s_j.
- Tras $M-1$ iteraciones, nos queda que, a la última escala, se le llama la escala M, con c_m y s_m.
- La comparación de iluminación sólo se calcula en la última escala, la M, y se le llama l_m.

Este algoritmo se puede modelizar matemáticamente, definiendo la función MS-SSIM como se hace en la Definición 3.12.

***Definición* 3.12** $MSSIM(x,y) = [l_M(x,y)]^{\alpha_M} \dot{\prod}_{j=1}^{M} [c_j(x,y)]^{\beta_j} [s_j(x,y)]^{\gamma_j}$

Para evaluar el rendimiento de MS-SSIM, se compararon los resultados de las dos métricas más importantes PSNR y SSIM (en realidad, SS-SSIM, es decir, Single-Scale SSIM; o sea, que se han cogido SSIM con diferentes escalas) y en sus resultados contra las evaluaciones MOS con la base de datos LIVE [9], observándose que, respecto a los valores MOS, MS-SSIM (o en su defecto, a más escalas de SSIM) obtenía los mejores resultados [Wang, 2003].

3.2. Superresolución general

La SR no es un campo de estudio reciente, pero sus mayores contribuciones (que han llegado desde el mundo del DL) se han producido en los últimos años, a partir del 2014.

Para el análisis del estado del arte de la SR de imágenes en general nos hemos basado en la revisión de [Wang et al., 2020]. Este artículo es de febrero del 2020 y resume todas las tendencias del ámbito de la SR, junto con sus referencias y recomendaciones futuras.

La SR, es una clase importante de técnicas de VC y procesamiento de imagen, que tiene una gran cantidad de aplicaciones como en medicina [Kouame and Ploquin, 2009], [Isaac and Kulkarni, 2015], [Huang et al., 2017], seguridad [Zhang et al., 2010], [Rasti et al., 2016], u otro tipo de tareas [Dai et al., 2015], [Haris et al., 2018], [Sajjadi et al., 2016], [Bai et al., 2018].

[9] https://live.ece.utexas.edu/research/Quality/

Tradicionalmente, se han utilizado técnicas de todo tipo incluyendo métodos basados en predicciones [Keys, 1981], [Duchon, 1979], [Irani and Peleg, 1991], en bordes (edge-based) [Freedman and Fattal, 2011], [Jian Sun et al., 2008], en estadística [Kim and Kwon, 2010], [Xiong et al., 2010], en parches [Freeman et al., 2002], [Hong Chang et al., 2004], [Glasner et al., 2009] etcétera.

Sin embargo, con el rápido crecimiento del DL, se han desarrollado numerosos modelos para hacer SR que han superado los registros del estado del arte hasta ese momento. Estos métodos van desde las primeras CNN hasta las prometedoras GANs [Goodfellow et al., 2014]. Los algoritmos de SR basados en DL difieren entre sí en aspectos como la arquitectura, la función de pérdida o loss, etc. que se detallarán más adelante.

Antes de la revisión de [Wang et al., 2020], la mayoría de revisiones del campo se centraban en técnicas tradicionales, mientras que Wang repasa el estado del arte basado en DL. En concreto, Wang nos ofrece:

- Una visión general de todos los métodos y alternativas típicos de SR en DL.
- Los avances más recientes en este campo.
- Los retos a futuro, problemas abiertos y las nuevas tendencias.

En la Figura 3.4 se presenta una rica taxonomía del estado del arte de la SR (y, por consiguiente, de este artículo), que puede servir como mapa conceptual. Según esta taxonomía, el problema de la SR se divide en dos aproximaciones: SR supervisado (Supervised Image Super-resolution) y SR no supervisado (Unsupervised Image Super-resolution), siendo el primero el que más se desarrolla, y compartiendo ambos características comunes como las métricas (Performance Evaluations) o sus aplicaciones (Domain-specific Applications). En cuanto al SR supervisado, se estudian cinco estrategias suyas: las posiciones del upsampling (Model Frameworks), los métodos de upsampling (Upsampling methods), el diseño de red (Network design), las estrategias de aprendizaje (Learning Strategies) y otros métodos (Other Improvements) como aumento de datos.

3.2.1. Fundamentos

Se define el problema de la SR formalmente en la Definición 3.13.

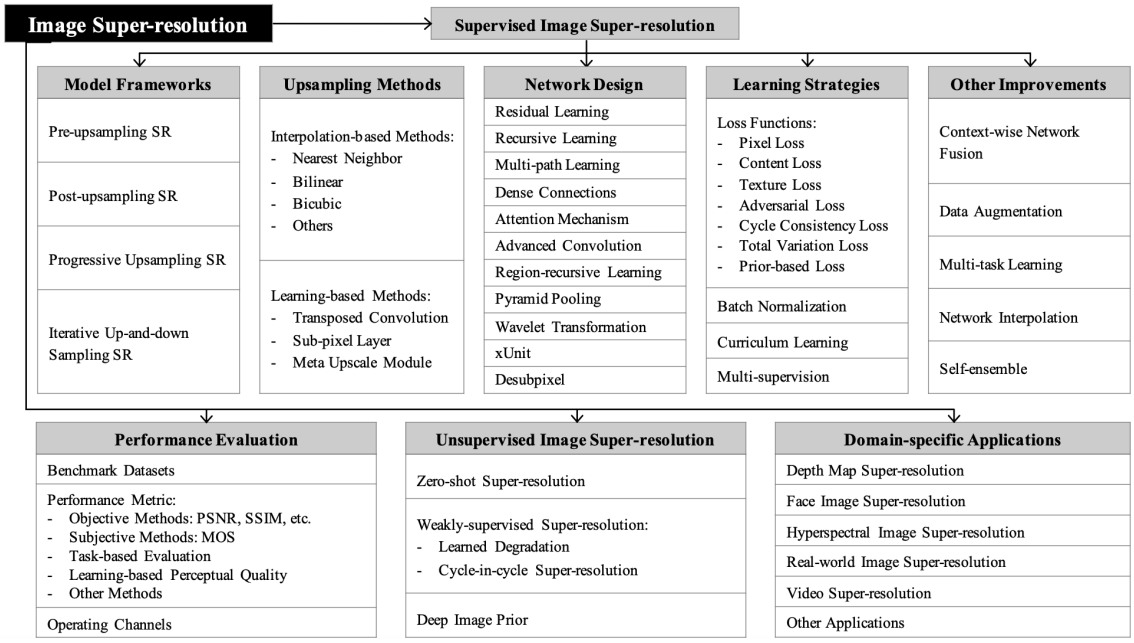

Figura 3.4: Taxonomía del campo de la SR.
Fuente: [Wang et al., 2020]

***Definición* 3.13** *Se puede expresar toda imagen LR I_x como salida de una degradación D:*

$$I_x = D(I_y; \delta)$$

Donde D representa una función de degradación de imagen, Iy es la correspondiente imagen HR, y δ es el conjunto de parámetros de la función D.

Lo habitual es que el valor de D sea desconocido y sólo se tengan las imágenes LR. En este caso, que se conoce como SR ciego (del inglés blind SR), se debe encontrar una aproximación HR I_z de la imagen original I_y, partiendo sólo de la imagen I_x a LR, como se muestra en la Definición 3.14.

***Definición* 3.14** *Definimos una imagen reconstruida a HR de la siguiente manera:*

$$I_z = F(I_x; \omega)$$

Donde F es el modelo o proceso de SR y ω es el conjunto de parámetros de F.

Así, muchos trabajos definen D como un simple submuestreo de la imagen, como en la Definición 3.15.

***Definición* 3.15** *Definimos D de la manera más simple como:*

$$D(I_y;\delta) = (I_y)\downarrow_s$$

Donde $\downarrow_s$ *es una operación de submuestreo con el factor de submuestreo s y siendo s un parámetro perteneciente a* δ.

Sin embargo, hay otros trabajos [Zhang et al., 2017] en los que se define D de una manera más realista y más útil para la SR, que es como combinación de más operaciones, como se puede ver en la Definición 3.16.

***Definición* 3.16** *Definimos D de la manera más compleja como:*

$$D(I_y;\delta) = (I_y \circledast k)\downarrow_s + n_c$$

Donde $(I_y \circledast k)$ *representa la convolución entre la imagen HR* I_y *y un kernel de difuminado k,* n_c *es ruido gaussiano blanco añadido con desviación estándar sigma; siendo k, s y c parámetros pertenecientes a* δ.

Como apunte, el método más típico de submuestreo es la interpolación bicúbica con anti-aliasing [10].

En cualquier caso, objetivo de la SR de manera formal siempre es la misma (ver Definición 3.17).

***Definición* 3.17** *Definimos la SR como un proceso de optimización (concretamente, de minimización):*

$$SR = minimizate(L(I_x, I_y))$$

Donde L representa una función de loss entre dos imágenes,

3.2.2. Tipos de modelos de DL según su estructura

En SR por DL, es esencial determinar cuándo se hará el sobre-muestreo o upsampling de una imagen, es decir, cuándo se aumentará el tamaño correspondiente de LR a HR.

[10] https://es.wikipedia.org/wiki/Antialiasing

Aunque existen numerosas arquitecturas de las CNN, todas se pueden distinguir en base a en qué momento hacen este upsampling.

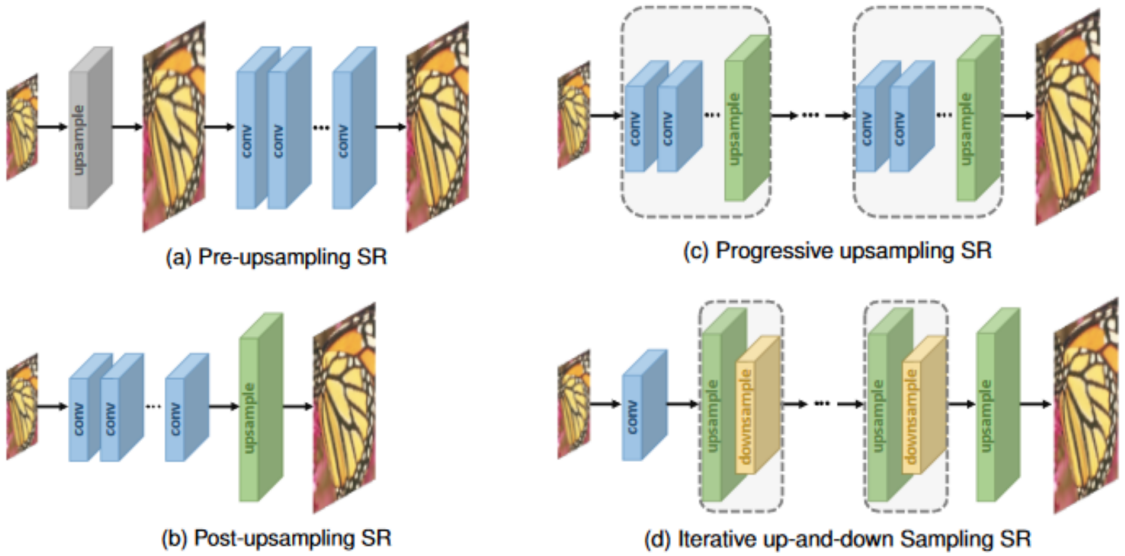

Figura 3.5: Arquitecturas de CNN, en función de la posición del upsampling en la red; en a una arquitectura pre-upsampling 3.2.2, en b una arquitectura post-upsampling 3.2.2, en c una arquitectura de upsampling progresivo 3.2.2 y en d una arquitectura con upsampling iterativo 3.2.2.
Fuente: [Wang et al., 2020]

Modelos de SR con pre-upsampling

Utilizar métodos de upsampling para obtener imágenes de HR y después mejorarlas mediante DL ha sido una de las primeras soluciones para la SR, lo cual se adoptó, por ejemplo, en el modelo SRCNN, propuesto por [Dong et al., 2014].

La idea es que las imágenes LR son sobre-muestreadas por medios tradicionales (como interpolación bicúbica) para tener el mismo tamaño de las imágenes HR (ver Figura 3.5a). Después, mediante CNNs, se reconstruyen los detalles de mayor calidad, es decir, se mejora su resolución.

La gran ventaja de esto es que, al haberse hecho ya la operación de upsampling, la CNN sólo tiene que mejorar una imagen HR, por lo que la dificultad de aprendizaje se reduce mucho. Además, estos modelos pueden coger imágenes de cualquier tamaño al que luego se le aplique un factor de escalado (scaling factor) adecuado.

Sin embargo, el upsampling que se utilice a menudo añade efectos secundarios como mayor ruido o difuminado. Además, como la mayoría de operaciones (las de la CNN) se

realizan en un espacio dimensional de mayor tamaño, se consumen muchos más recursos de espacio y tiempo [Dong et al., 2016], [Shi et al., 2016].

Modelos de SR con post-upsampling

Para mejorar la eficiencia computacional y hacer pleno uso de las ventajas del DL, en estos modelos [Dong et al., 2016], [Shi et al., 2016] se propone hacer la mayoría de operaciones en el espacio dimensional menor, por lo que la CNN recibe imágenes LR (sin upsampling previo) y a su salida se le aplica un upsampling que se ha aprendido (learnable upsampling -ver Sección 3.2.3-), no uno clásico (Figura 3.5b).

Puesto que la extracción de características sólo ocurre en dimensiones bajas y la resolución sólo se aumenta al final, se reduce mucho el coste computacional, por lo que este tipo de modelos se ha convertido en el más usado. Los modelos de este estilo difieren, sobre todo, en el método de learnable-upsampling que utilizan [Ledig et al., 2016], [Lim et al., 2017], [Tong et al., 2017], [Han et al., 2018].

Modelos de SR con progressive-upsampling

El post-upsampling tiene dos desventajas: por un lado, el proceso de upsampling se realiza sólo en un momento, por lo que para factores de escalado más alto, la dificultad de aprender sube mucho; por otro lado, para cada factor de escalado (o sea, para cada tamaño), se necesita entrenar un modelo de SR, por lo que esta técnica no vale para SR de diferentes escalas.

Precisamente para superar estos problemas, se diseñan upsampling progresivos en [Lai et al., 2017a], es decir, CNNs puestas en cascada que combinan aprendizaje con upsampling. En cada paso, las imágenes son aumentadas de resolución y después refinadas por las CNN (Figura 3.5c).

Así, descomponiendo una tarea difícil en numerosas tareas más simples, estos modelos [Wang et al., 2018b], [Lai et al., 2017b] consiguen reducir la dificultad de aprendizaje con factores más altos y, además, permiten aprender SR con diferentes escalas. Sin embargo, estos modelos presentan un diseño complicado de varios pasos y un entrenamiento bastante inestable, por lo que requieren estrategias más avanzadas de entrenamiento .

Modelos de SR con iterative up-and-down sampling

Para capturar mejor la dependencia entre los pares de imágenes LR-HR, se incorporó a la SR [Timofte et al., 2015b] un proceso iterativo llamado refinamiento de retro-proyección o back-projection refinement [Irani and Peleg, 1991], que calcula el error de reconstrucción y lo utiliza para ajustar la intensidad de la imagen HR.

Así, los modelos de tipo iterative up-and-down sampling [Haris et al., 2018], [Haris et al., 2019] plantean aplicar back-projection de forma iterativa, conectando las capas de muestreo ascendente y submuestreo alternativamente y reconstruyendo el resultado final de HR utilizando todas las reconstrucciones intermedias (ver Figura 3.5d). El modelo SRFBN [Li et al., 2019], que aprende representaciones excelentes, le añade skip-connections a esta idea.

Debido a que el diseño de back-projection no es todavía estándar y este mecanismo acaba de introducirse al mundo del DLSR, se considera a estos modelos con gran potencial, pero con mucho que estudiar sobre ellos.

3.2.3. Tipos de upsampling

Además de dónde ubicar el upsampling en el modelo, existe la necesidad de conocer qué método de upsampling utilizar. Lo típico ha solido ser utilizar métodos tradicionales [Jianchao Yang et al., 2008], [Yang et al., 2010], [Timofte et al., 2015a], [Schulter et al., 2015], pero las últimas investigaciones tienden a hacer uso de CNNs para que aprendan cómo hacer un upsampling apropiado.

Upsampling basado en interpolación

El escalado de imágenes basado en interpolación clásica ha sido ampliamente utilizado debido a lo sencillo de su implementación. Ejemplos de ello serían:

- Interpolación de vecino más próximo (de Nearest-neighbor). Es un algoritmo de pocos pasos; selecciona el valor del píxel más próximo para cada posición que será interpolada. Por ello, es un algoritmo muy rápido, pero produce resultados mejorables.
- Interpolación bilineal. Se llama así porque efectúa dos interpolaciones lineales; primero una en un eje y luego otra en el otro eje, donde se cogen 2×2 píxeles para

calcular el nuevo punto. Muestra resultados considerablemente mejores que los del anterior método [Keys, 1981] y sin disminuir demasiado la velocidad. Sus pasos se pueden ver en la Figura 3.6.

- Interpolación bicúbica. Al igual que el anterior, efectúa dos interpolaciones cúbicas; una en cada eje, pero esta vez tomando 4×4 píxeles en cuenta, por lo que es bastante más lento.

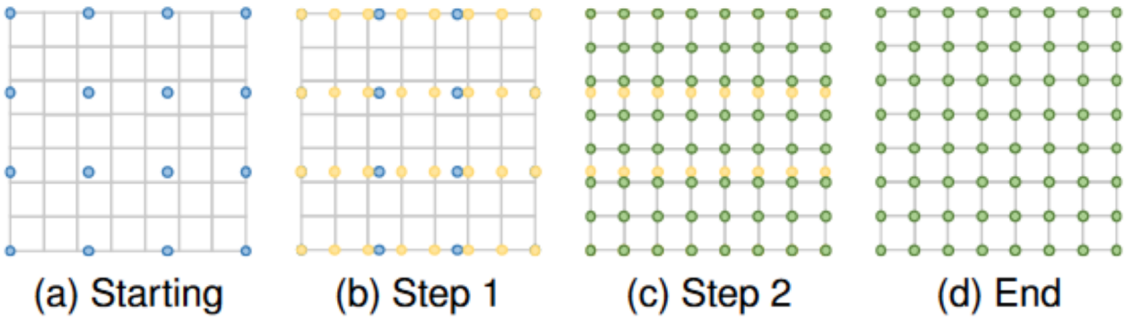

Figura 3.6: Pasos del algoritmo de interpolación bilineal.
Fuente: [Wang et al., 2020]

Como se puede ver, el upsampling basado en la interpolación mejora la resolución de la imagen basándose sólo en sus propias señales, es decir, píxeles. Pero tienen como punto negativo que suelen ser algoritmos costosos, además de que tienen efectos secundarios como aumento de ruido o resultados difusos.

Upsampling aprendido

Para superar los problemas recién citados, se proponen dos métodos de upsampling aprendido, es decir, no predeterminados.

El primero de ellos es la llamada capa convolucional transpuesta (del inglés Transposed Convolutional Layer), es decir, una capa de deconvolución o convolución inversa [Zeiler et al., 2010], [Zeiler and Fergus, 2013]. En este caso, la capa intenta realizar la transformación de manera opuesta a una convolución normal, es decir, prediciendo la entrada basándose en mapas de características expandidos al tamaño de salida de la convolución. Concretamente, aumenta la resolución de la siguiente manera: expandiendo el tamaño de la imagen insertando ceros y aplicando después la convolución, como se puede ver en la Figura 3.7.

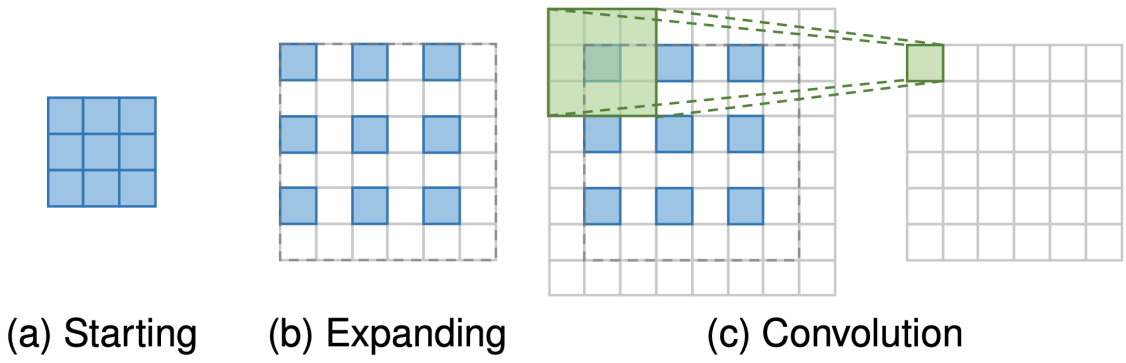

Figura 3.7: Pasos de una capa convolucional transpuesta.
Fuente: [Wang et al., 2020]

Dado que la convolución transpuesta aumenta el tamaño de la imagen mientras mantiene un patrón de conectividad compatible con la convolución tradicional, se utiliza ampliamente como capa de upsampling en numerosos modelos de SR [Haris et al., 2018], [Mao et al., 2016], [Tong et al., 2017].

Sin embargo, esta capa puede causar fácilmente una superposición desigual en cada eje [Odena et al., 2016] y los resultados multiplicados en ambos ejes crean un patrón del estilo del tablero de ajedrez, que perjudican el rendimiento de los modelos.

Otro upsampling aprendido es la capa subpíxel [Shi et al., 2016], que realiza upsampling generando por convolución 2^s canales (donde s es el factor de escala) y después remodelándolos (reshaping), como se muestra en la Figura 3.8.

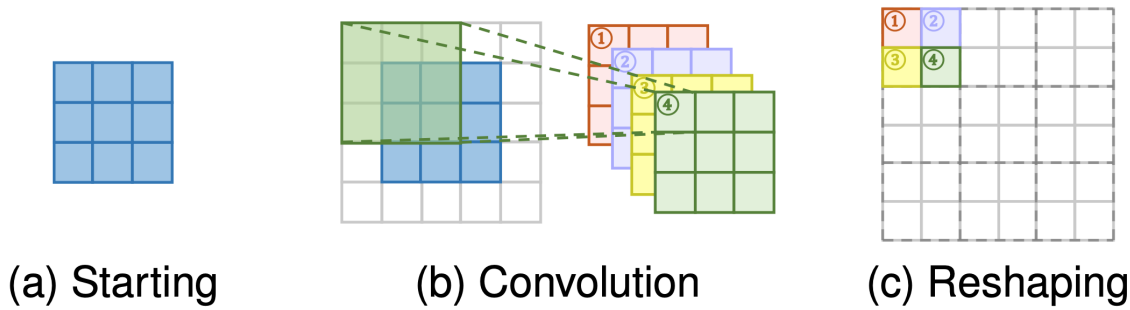

Figura 3.8: Pasos de una capa subpíxel.
Fuente: [Wang et al., 2020]

Comparado con la capa de convolución transpuesta, la capa de subpíxel tiene un campo receptivo más amplio, que proporciona más información contextual para ayudar a generar detalles más realistas, por lo que se utiliza en muchos modelos de SR [Ledig et al., 2016], [Ahn et al., 2018], [Zhang et al., 2017], [Zhang et al., 2018c]. Sin embargo, ya que la dis-

tribución de los campos receptivos es desigual y las regiones en bloque en realidad comparten el mismo campo receptivo, puede dar lugar a algunos artefactos [11] cerca de los límites de diferentes bloques.

Por otro lado, la predicción independiente de píxeles adyacentes en una región en bloque puede causar salidas poco uniformes. Así, se propone PixelTCL [Gao et al., 2020], que reemplaza el predicción independiente a predicción secuencial interdependiente, y produce resultados más suaves y consistentes.

3.2.4. Funciones de pérdida o loss

En el campo de la SR, las funciones de loss se utilizan para cuantificar o medir el error que un modelo ha tenido en una reconstrucción, para después utilizar ese error para optimizar los parámetros de la NN.

Al principio, se utilizaban funciones de pérdida basadas en medir la diferencia de intensidad por píxeles, pero luego se vio que esto no medía bien la diferencia entre imágenes y se propusieron nuevas funciones para lograr reconstruir las imágenes con mayor calidad, como por ejemplo el loss de adversarios [Ledig et al., 2016] o el loss de prioridades [Bulat and Tzimiropoulos, 2017].

Así, se puede dividir las funciones de loss en el menos siete categorías diferentes:

- Loss de píxeles.
- Loss de contenido.
- Loss de texturas.
- Loss de adversarios.
- Loss de ciclo.
- Loss de variación.
- Loss de prioridades.

De las cuales se han estudiado los tres primeros.

[11] https://www.sciencedirect.com/topics/computer-science/image-artifact

Loss de píxeles

El loss de píxeles (del inglés pixel-loss) es el primer tipo que se utilizó. Mide sólo la diferencia de intensidades a nivel de píxeles y sus funciones más populares son el error absoluto medio (MAE por sus siglas en inglés) o L1, y el error cuadrático medio (MSE por sus siglas en inglés) o L2. Se denotan formalmente como en la Definición 3.18.

***Definición* 3.18** *Definimos L1 y L2 como:*

$$L_{L1}(x,y) = \frac{1}{hwc}\sum_{i,j,k}|x_{i,j,k} - y_{i,j,k}|$$

$$L_{L2}(x,y) = \frac{1}{hwc}\sum_{i,j,k}(x_{i,j,k} - y_{i,j,k})^2$$

Donde h, w y c son la altura (height), la anchura (width) y el número de canales, respectivamente.

Además, existe una variante del loss de píxel L1, llamado el loss de Charbonnier [Lai et al., 2017b], [Bruhn et al., 2004], que se muestra en la Definición 3.19.

***Definición* 3.19** *Definimos loss de Charbonnier como:*

$$L_{Charbonnier}(x,y) = \frac{1}{hwc}\sum_{i,j,k}\sqrt{(x_{i,j,k} - y_{i,j,k})^2 + \varepsilon^2}$$

Donde ε es una constante numérica necesaria para estabilidad numérica.

Comparando el loss L1 con el L2, el segundo penaliza mucho los errores grandes, pero es más tolerante con los errores pequeños, por lo que sus resultados suelen ser más suaves. En la práctica, L1 muestra un desempeño mejor [Ahn et al., 2018], [Lim et al., 2017], [Zhao et al., 2015].

Debido a que PSNR está altamente relacionado con la diferencia de píxeles, minimizar esta diferencia (es decir, que los loss de píxeles nos den resultados más bajos) maximiza directamente PSNR. Sin embargo, debido a que el loss de píxel no tiene en cuenta detalles de alto nivel como la calidad perceptual [Johnson et al., 2016] o las texturas [Sajjadi et al., 2016], se pierden detalles de alto nivel y sus resultados a menudo son perceptivamente insatisfactorios con las texturas más suaves [Ledig et al., 2016], [Johnson et al., 2016], [Z. Wang and Simoncelli, 2004], [Wang, 2003].

Loss de contenido

El loss de contenido (del inglés content loss) tiene como objetivo precisamente evaluar la calidad percibida de las imágenes [Johnson et al., 2016], [Dosovitskiy and Brox, 2016]. Específicamente, mide las diferencias semánticas entre las imágenes usando una red de clasificación de imágenes previamente entrenada, como se muestra en la Definicíon 3.20.

***Definición* 3.20** *Si se denota la NN pre-entrenada como θ y las representaciones de alto nivel extraídas en la capa l-ésima como $\theta^{(l)}$, el loss de contenido se define como la distancia euclídea entre las representaciones de alto nivel de dos imágenes:*

$$L_{Content}(x,y) = \frac{1}{h_l w_l c_l}\sqrt{\sum_{i,j,k}(\theta^{(l)}_{i,j,k}(x) - \theta^{(l)}_{i,j,k}(y))^2}$$

Donde h_l, w_l y c_l son la altura (height), la anchura (width) y el número de canales, en una capa l-ésima de la red.

Esencialmente, el loss de contenido transfiere el conocimiento aprendido de las características de las imágenes desde la red de clasificación, θ, a la red SR .

Por tanto, a diferencia de loss de píxel, esta función de pérdida fomenta que la imagen de salida sea perceptualmente similar a la imagen de destino, en lugar de imponer que coincidan exactamente píxel a píxel. Por lo tanto, produce resultados visualmente más realistas o detallados al ojo humano y también se usa ampliamente en SR [Sajjadi et al., 2016], [Ledig et al., 2016], [Johnson et al., 2016], [Bulat and Tzimiropoulos, 2017], [Wang et al., 2018b], siendo VGG [Simonyan and Zisserman, 2014] y ResNet [Kaiming He, 2015] las CNNs pre-entrenadas más comúnmente utilizadas.

Loss de texturas

El tercero es el loss de texturas (del inglés texture loss), que se introduce en la SR debido a que las imágenes reconstruidas han de tener el mismo estilo (colores, texturas, contraste) que la imagen objetivo, pero las nuevas investigaciones de DL han incorporado la transferencia de estilo (style transfer) [Gatys et al., 2015], [Gatys et al., 2016], por lo que ahora existen nuevos elementos a tener en cuenta. Debido a esto, también se le llama style reconstruction loss.

La textura de la imagen se define como las correlaciones entre los diferentes canales de características (obtenidos también de redes pre-entrenadas) y se define como una matriz de Gram [12] como en la Definición 3.21.

***Definición* 3.21** *Si se tiene la siguiente matriz de Gram, se define el loss de texturas de la siguiente manera:*

$$G^l \in \mathbb{R}^{c_l \times c_l},\ G^{(l)}_{i,j}(I) = vec(\theta^{(l)}_i(I))vec(\theta^{(l)}_j(I))$$

$$L_{Content}(x,y) = \frac{1}{c_l^2}\sqrt{\sum_{i,j}(G^{(l)}_{i,j}(x) - G^{(l)}_{i,j}(y))^2}$$

Donde G es es el producto interno entre los mapas de características vectorizados i y j, en la capa j-ésima

Empleando este loss de texturas, se crean texturas más realistas y resultados visualmente más satisfactorios cualitativamente, como se propone en la EnhanceNet [Sajjadi et al., 2016].

3.2.5. Otras técnicas

Además de lo mencionado, se han estudiado (y utilizado) otras técnicas para mejorar el rendimiento de las CNNs en SR como lo son batch normalization (ver Sección 3.2.5) o data augmentation (ver Sección 3.2.5).

Batch normalization

Para acelerar y estabilizar el entrenamiento de las CNN profundas, se propuso la normalización de lotes (batch normalization) [Ioffe and Szegedy, 2015] para reducir el cambio interno covariable de las redes (internal covariate shift [13]).

Concretamente, se realiza una normalización para cada mini lote y se entrenan dos parámetros de transformación adicionales para cada canal, a fin de preservar la capacidad de representación. Ya que el batch normalization calibra la distribución de características intermedias y mitiga los gradientes que desaparecen, permite el uso de mayores learning rates y ser menos cuidadoso con la inicialización. Debido a ello, se utiliza en muchos

[12] https://es.wikipedia.org/wiki/Matriz_de_Gram
[13] https://machinelearning.wtf/terms/internal-covariate-shift/

modelos SR [Ledig et al., 2016], [Zhang et al., 2017], [Tai et al., 2017], [Tai et al., 2017], [Liu et al., 2018], [Sønderby et al., 2016].

Sin embargo, se argumentó que el batch normalization pierde la información de escala de cada imagen [Lim et al., 2017], por lo que se ahorró su uso (alrededor del 40 % del coste en memoria) para desarrollar un modelos mayores y aumentar el rendimiento considerablemente [Wang et al., 2018b], [Wang et al., 2018a], [Chen et al., 2018].

El batch normalization se explica más intuitivamente que en esta revisión en [14] o en [15], pero su proceso es el siguiente: se normaliza la salida de una capa de activación previa restando la media del lote y dividiendo por la desviación estándar del lote. Esto se ilustra en la Figura 3.9.

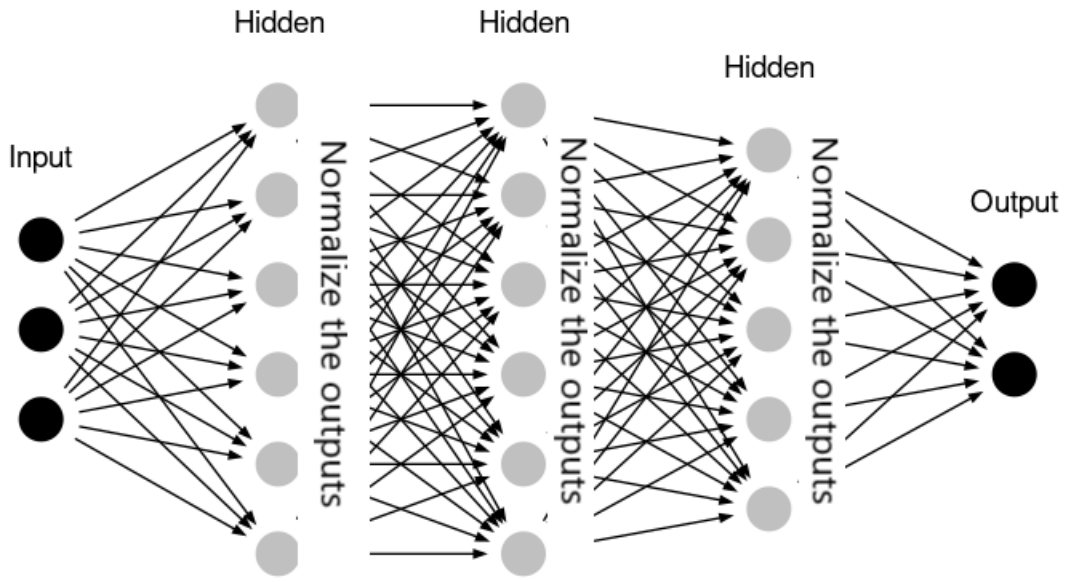

Figura 3.9: Batch normalization intuitivamente: se normaliza la salida de cada capa.
Fuente: https://medium.com/hyunjulie/batch-normalization-simple-summary-d6a873960e5c

Data augmentation

Aumentar el conjunto de datos de entrenamiento artificialmente (lo que se llama data augmentation), es una de las técnicas más utilizadas para aumentar el rendimiento de

[14] https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c
[15] https://medium.com/hyunjulie/batch-normalization-simple-summary-d6a873960e5c

una red neuronal profunda. Consiste en modificar los datos de entrada para lograr más y enriquecer los conjuntos de datos que se utilizan.

En VC, los tipos de data augmentation más típicos son el recorte, volteo, escalado, rotación y espejo [Lai et al., 2017b], [Lim et al., 2017], [Timofte et al., 2015b], [Tai et al., 2017], [Han et al., 2018], [Hui et al., 2018]. Cuando las imágenes son de tipo RGB, también se utiliza el intercambio aleatorio de los canales de color [Bei et al., 2018]. Podemos ver un ejemplo en la Figura 3.10.

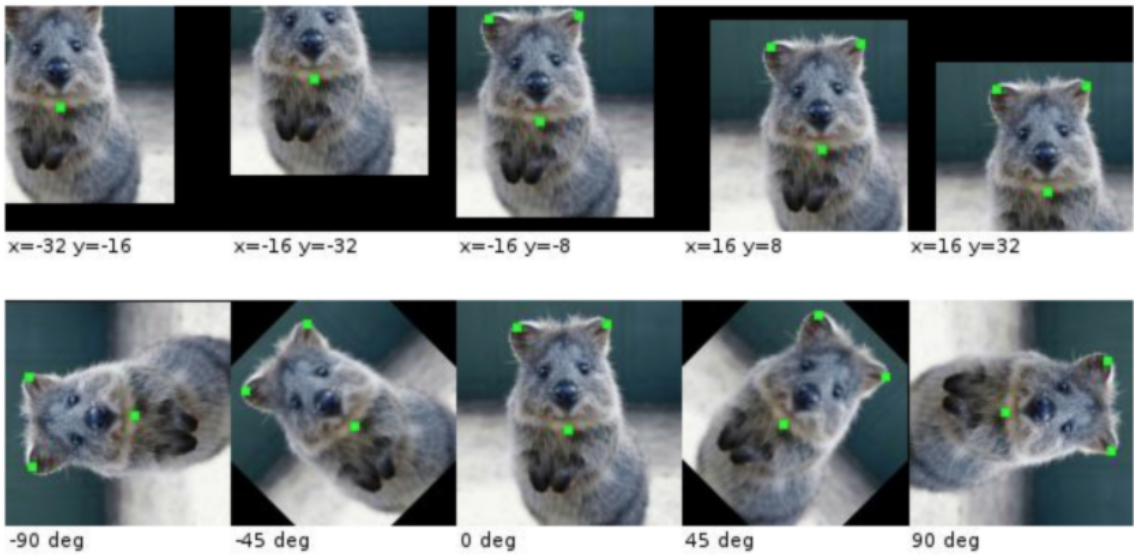

Figura 3.10: Data augmentation con varias traslacione sobre la imagen original (arriba) y varias rotaciones sobre la imagen original (abajo).
Fuente: https://www.yanxishe.com/TextTranslation/1649

3.2.6. CNNs en el estado del arte

Finalmente, la tabla de la Figura 3.11 resume las principales CNNs para SR en el estado del arte, así como sus técnicas y el año de publicación. Las primeras dos columnas describen el nombre y la fecha de publicación; la tercera, el punto en el que se hace upsampling; la cuarta, el tipo de upsampling utilizado; la quinta a la octava hablan de diseño de red que no se ha estudiado para este trabajo; la novena dice si utiliza el loss L1 o no; la décima si utiliza el loss L2 o no; y la undécima y última menciona las palabras clave que definen qué se hace en esos trabajos.

Method	Publication	Fw.	Up.	Rec.	Res.	Dense	Att.	$\mathcal{L}_{L1}$	$\mathcal{L}_{L2}$	Keywords
SRCNN [22]	2014, ECCV	Pre.	Bicubic						✓	
DRCN [82]	2016, CVPR	Pre.	Bicubic	✓	✓				✓	Recursive layers
FSRCNN [43]	2016, ECCV	Post.	Deconv						✓	Lightweight design
ESPCN [156]	2017, CVPR	Pre.	Sub-Pixel						✓	Sub-pixel
LapSRN [27]	2017, CVPR	Pro.	Bicubic		✓			✓		$\mathcal{L}_{\text{pixel\_Cha}}$
DRRN [56]	2017, CVPR	Pre.	Bicubic	✓	✓				✓	Recursive blocks
SRResNet [25]	2017, CVPR	Post.	Sub-Pixel		✓				✓	$\mathcal{L}_{\text{Con.}}$, $\mathcal{L}_{\text{TV}}$
SRGAN [25]	2017, CVPR	Post.	Sub-Pixel		✓					$\mathcal{L}_{\text{Con.}}$, $\mathcal{L}_{\text{GAN}}$
EDSR [31]	2017, CVPRW	Post.	Sub-Pixel		✓			✓		Compact and large-size design
EnhanceNet [8]	2017, ICCV	Pre.	Bicubic		✓					$\mathcal{L}_{\text{Con.}}$, $\mathcal{L}_{\text{GAN}}$, $\mathcal{L}_{\text{texture}}$
MemNet [55]	2017, ICCV	Pre.	Bicubic	✓	✓	✓			✓	Memory block
SRDenseNet [79]	2017, ICCV	Post.	Deconv		✓	✓			✓	Dense connections
DBPN [57]	2018, CVPR	Iter.	Deconv		✓	✓			✓	Back-projection
DSRN [85]	2018, CVPR	Pre.	Deconv	✓	✓				✓	Dual state
RDN [93]	2018, CVPR	Post.	Sub-Pixel		✓	✓		✓		Residual dense block
CARN [28]	2018, ECCV	Post.	Sub-Pixel	✓	✓	✓		✓		Cascading
MSRN [99]	2018, ECCV	Post.	Sub-Pixel		✓			✓		Multi-path
RCAN [70]	2018, ECCV	Post.	Sub-Pixel		✓		✓	✓		Channel attention
ESRGAN [103]	2018, ECCVW	Post.	Sub-Pixel		✓	✓		✓		$\mathcal{L}_{\text{Con.}}$, $\mathcal{L}_{\text{GAN}}$
RNAN [106]	2019, ICLR	Post.	Sub-Pixel		✓		✓	✓		Non-local attention
Meta-RDN [95]	2019, CVPR	Post.	Meta Upscale		✓	✓		✓		Magnification-arbitrary
SAN [105]	2019, CVPR	Post.	Sub-Pixel		✓		✓	✓		Second-order attention
SRFBN [86]	2019, CVPR	Post.	Deconv	✓	✓	✓		✓		Feedback mechanism

Figura 3.11: CNNs significativas para el problema de SR (las referencias de la primera columna se corresponden con las de la revisión [Wang et al., 2020], no con las de este trabajo).
Fuente: [Wang et al., 2020]

3.3. Superresolución en imágenes de microscopía

Dado que el principal tema de este trabajo es la SR en microscopía, lo que trataremos aquí no serán las múltiples opciones que nos da la SR, sino solamente aquellos artículos especializados en SR para microscopía.

3.3.1. PSSR

El primer y más importante de los artículos de dominio específico fue la base original por la que se empezó este TFG, es decir, el gran objetivo fue emular sus resultados.

Se llama Deep Learning-based Point Scanning Super-resolution (Superresolución de escaneo de puntos, basado en Deep Learning), cuyo modelo hemos abreviado como PSSR durante todo el trabajo [Fang et al., 2019].

El hito de esta investigación es utilizar una CNN para hacer SR sobre imágenes de microscopía electrónica y de campo claro. Así, debido a la importancia de este artículo, se ahondará mucho en él; sus porqués, sus resultados y sus técnicas.

Como en todos los sistemas de capturas de imágenes, los microscopios sufren del llamado

triángulo del compromiso, una relación que dicta que la resolución, la intensidad de la iluminación (y, por tanto, el daño de las muestras o samples) y la velocidad para obtener las capturas siempre están en tensión. Es decir, que no se puede optimizar una de ellas sin reducir al menos una de las otras dos. Esto se representa en la Figura 3.12.

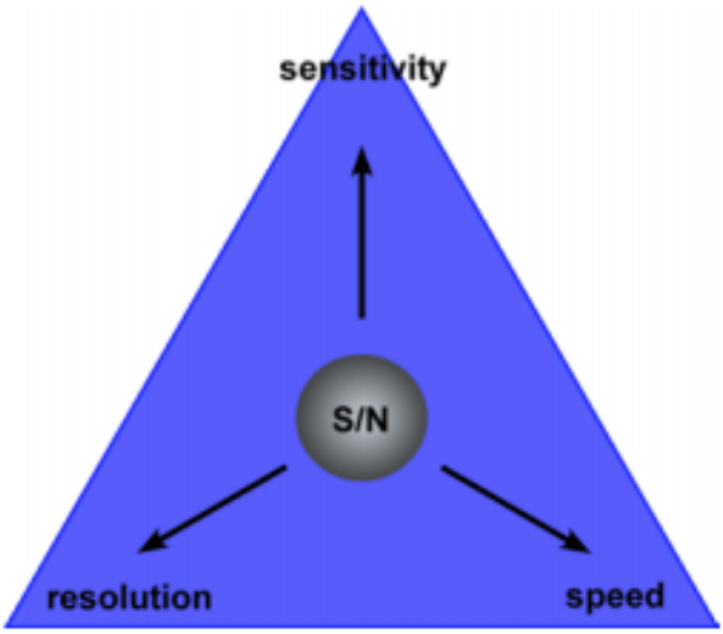

Figura 3.12: Tríangulo de compromiso en VC
Fuente: https://miap.eu/fileadmin/primary/Public/user_uploads/Bilder/MICROSCOPY/LIC/EN_wp_Airyscan-detector.pdf

Este triángulo se acentúa aún más en sistemas de escaneo de puntos, para los que imágenes de mayor resolución requieren un mayor número de píxeles adquiridos secuencialmente para garantizar el correcto muestreo, por lo que el tiempo de obtención de esas imágenes y el daño de la muestra aumentan en proporción directa a la resolución de píxeles.

Así, [Fang et al., 2019] logran, utilizando DL, que la resolución espaciotemporal de las muestras de laboratorio aumente de formas que antes no eran posibles. Hay que aclarar que denomina resolución temporal a la resolución que cambia en función al tiempo, es decir, que cuanto más rápido se obtienen las muestras, menor es la resolución temporal. Por tanto, se llama resolución espaciotemporal a la resolución que cambia con el tiempo y también el espacio.

Debido a la incapacidad de una red, SBFSEM [Denk and Horstmann, 2004], de detectar vesículas presinápticas, [Fang et al., 2019] decidieron probar si una CNN entrenada en pares de imágenes LR-HR (de 2 y 8 nm por píxel respectivamente) podía “super-resolver”

las LR, es decir, pasarlas a resolución de 2 nm. La idea era ver si un modelo podría hacer SRx2 y SRx4, es decir, SR por un factor de escalado de 2 y SR por un factor de escalado de 4.

Así, utilizando imágenes submuestreadas a 16x (o sea, de tamaño 16 veces menor que una de HR), la velocidad de obtención de imágenes HR sería 16 veces más rápida, habría 16 veces menos daño de muestra y, además, cada imagen de las 16 ocuparía 16 veces menos en memoria debido a que tendría 16 veces menos cantidad de píxeles.

Para generar el conjunto de entrenamiento, se adquirió un conjunto de alrededor de 130GB de imágenes a alta resolución con un microscopio electrónico de barrido de transmisión, tSEM [Kuwajima et al., 2013] (ver Sección 4.2.1). Dichas imágenes HR se degradaron por medio de submuestreo y ruido gaussiano para simular las imágenes correspondientes LR. Este conjunto de imágenes HR reales y LR semi-sintéticas se entrenó en una versión de la U-Net, la cual usa una red ResNet34 [Kaiming He, 2015] como base. Esto se puede ver en la Figura 3.13

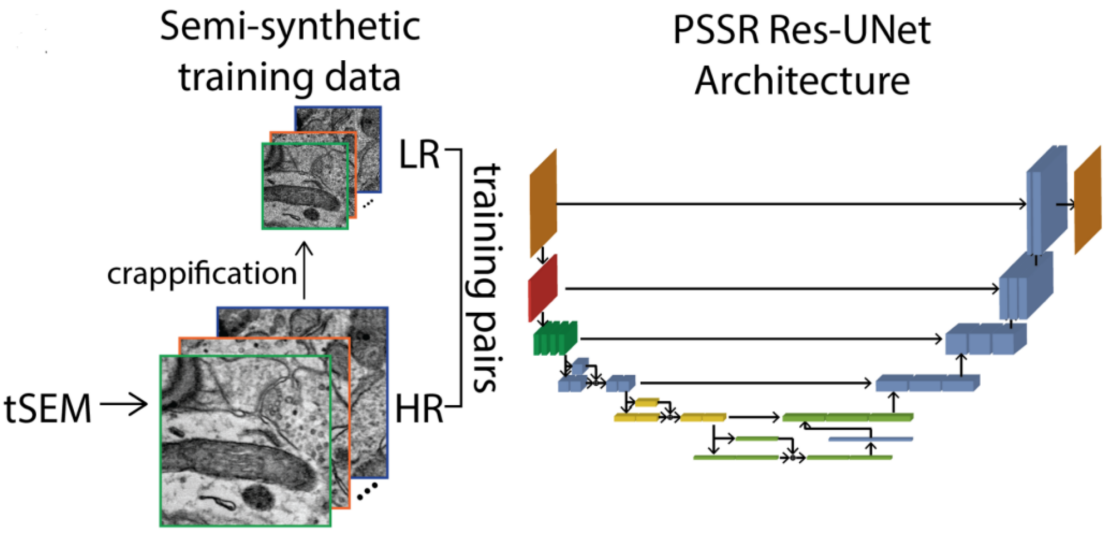

Figura 3.13: Entrenamiento de PSSR: Creación de datos de entrenamiento semi-sintéticos (a la izquierda) y arquitectura de red propuesta (a la derecha).
Fuente: [Fang et al., 2019]

Como puede observarse en la Figura 3.14, los resultados fueron muy positivos en ambos casos, tanto evaluando cualitativamente los resultados haciendo zoom, como evaluando numéricamente los resultados con métricas. Por tanto, [Fang et al., 2019] afirman que el modelo generaliza correctamente, ya que el modelo funciona para muestras de diferentes animales, tomadas en diferentes laboratorios con diferentes microscopios electrónicos.

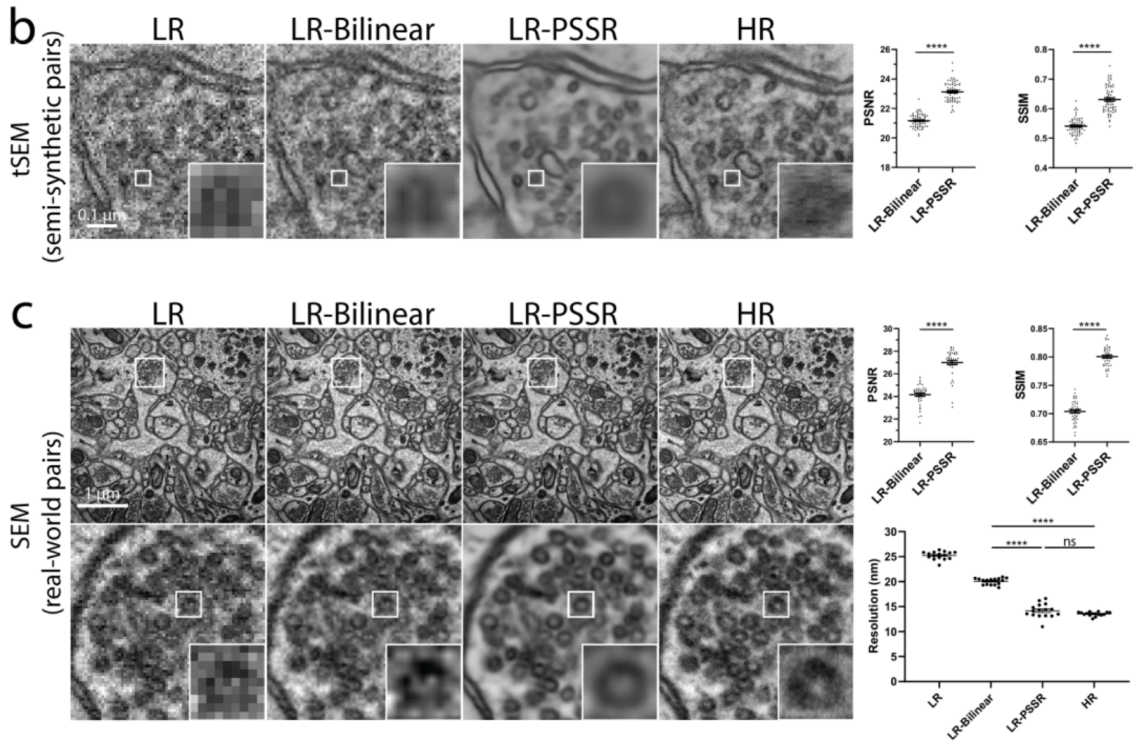

Figura 3.14: En la subfigura b, el dataset semisintético reconstruido de diferentes maneras con el método bilineal y el método propuesto, juntos con sus resultados en las métricas; en la subfigura c, lo mismo con el dataset de datos reales. Los recuadros blancos muestran diferentes zooms de la misma imagen.

Fuente: [Fang et al., 2019]

Sin embargo, un gran problema de las imágenes generadas por modelos de DL son los llamados falsos positivos o alucinaciones (del inglés hallucinations) [Moen et al., 2019], [Weigert et al., 2018], [Chamier et al., 2019], [Belthangady and Royer, 2019], que son objetos distinguibles parecidos a lo que se desea detectar, pero que no estaban en la imagen original. Por ello, para saber si se podía o no confiar en el modelo entrenado en experimentos donde no existan imágenes HR, se adquirió manualmente un conjunto de pares LR y HR, y después se compararon los resultados de aplicar SR por interpolación bilineal o por la red propuesta en PSSR. Los resultados muestran con métricas objetivas que LR a PSSR obtiene números mejores (ver gráficas de la Figura 3.14).

Para probar el modelo en un contexto más real del mundo de la biología, se mezclaron muestras y se pidió a dos expertos que anotaran en ellas vesículas presinápticas. Se encontró que la anotación basada en las reconstrucciones de PSSR fue significativamente más precisa que las de la interpolación bilineal. Curiosamente, mientras que la utilización de PSSR redujo los falsos negativos en 300 %, también tuvo un número ligeramente mayor

de falsos positivos que usando interpolación bilineal. Sobre todo, es importante destacar que la varianza entre los resultados usando PSSR y las imágenes originales HR fue similar a la varianza entre los resultados de dos humanos expertos sólo con datos HR. Es decir, la detección de vesículas hecha sobre los datos PSSR está, probablemente, muy cerca de la máxima precisión posible.

En particular, la CNN usada en PSSR tenía las siguientes características:

- Se utilizó una U-Net basada en ResNet.
- La ResNet se pre-entrenó con datos de ImageNet [16].
- Para el decoder, en vez de utilizar la interpolación bicúbica para el upsampling, se utilizaron capas de upsampling aprendible.

Se utilizó MSE como función de loss y el optimizador SGDR, que es SGD con reinicios (SGD with restarts)[Loshchilov and Hutter, 2016]. SGDR, además de los beneficios típicos de SGD, permite reiniciar el learning rate por cada época de entrenamiento, siguiendo la forma de una función de coseno, lo cual permite obtener un menor loss con una eficiencia computacional mayor.

En cuanto al learning rate, se utilizó uno cíclico acotado entre una cota superior y una inferior para evitar así los puntos de silla [Smith and Topin, 2017].

Como métricas principales de evaluación de los resultados, se utilizaron PSNR y SSIM, sabiendo que PSNR tiene un bajo rendimiento cuando se trata de estimar la calidad perceptiva humana, así como que SSIM tiene más en cuenta factores de distorsión como la iluminación o el contraste.

En este trabajo se destaca el hecho de que cualquier resultado que arroje un modelo de DL es una predicción y no un resultado de perfecta precisión. Si en un experimento se desea utilizar un modelo de DL, primero se tiene que validar si la precisión que el modelo alcanza es óptima o no para ese mismo experimento. Por ejemplo, en el caso de las vesículas, este modelo LR-PSSR sí parece ser útil, puesto que el error entre detectar vesículas con sus resultados o con imágenes HR es mínimo. Pero podría ser que no fuera así con otros experimentos, donde el modelo no restaure las imágenes lo suficientemente bien como para detectar mejor ciertos elementos.

[16] http://www.image-net.org/

Dicho esto, pese a que las imágenes predichas por el modelo no sean idénticas a las imágenes HR objetivo, el hecho de que los sistemas de imágenes tengan esas relaciones directas entre la resolución, el daño y el detalle de las estructuras, PSSR se convierte en una buena opción para obtener muestras rápidas y con suficiente fidelidad con respecto a las HR, sin dañar las imágenes.

Cabe destacar que en este trabajo sólo se ha utilizado como función de loss el MSE, debido a que se querían resultados que dependieran sólo de la información de los píxeles. Por tanto, se anticipa que, si se utilizaran funciones de loss más complejas, probablemente se obtendrían resultados mejores. Para futuros experimentos, se propone adquirir una cantidad pequeña de imágenes HR para afinar el modelo antes de adquirir las imágenes LR. Además, en la actualidad, los modelos de aprendizaje no supervisado o auto-supervisado (self-supervised) parecen estar siendo capaces de generalizar sus resultados mucho mejor.

3.3.2. Otros artículos

Otro artículo muy relacionado con nuestro trabajo fue publicado en 2017 y trata puramente de reconstrucción de imágenes de microscopia por Deep Learning. Se llama Content-Aware Image Restoration: Pushing the Limits of Fluorescence Microscopy (Restauración de imagen consciente del contenido: superando los límites de la microscopía de fluorescencia) [Weigert et al., 2018]. En siete ejemplos concretos, se ilustra cómo se pueden restaurar las imágenes de microscopía incluso si se usan 60 veces menos fotones durante la adquisición (reduciendo la fototoxicidad), cómo se puede lograr la resolución isotrópica con un submuestreo de 10 veces a lo largo de la dirección axial, y cómo las estructuras limitadas por difracción se puede resolver a velocidades de cuadro 20 veces más altas en comparación con los métodos más modernos. En todos estos casos, los modelos utilizados seguían también una arquitectura tipo U-Net.

Por último, otro artículo relacionado más con nuestro trabajo se llama A neural lens for super-resolution biological imaging (Una lente neuronal para SR en imágenes biomédicas) [Grant-Jacob et al., 2019]. En concreto, ofrece una aplicación clara, que es el uso combinado de SR para después hacer detección de granos de polen por segmentación.

4. CAPÍTULO

Investigación

Este capítulo es la contribución más importante del TFG, puesto que cubrirá el trabajo de investigación de Deep Learning que se ha introducido en los anteriores apartados. Concretamente, empezaremos por proponer una solución inicial sobre la que se iterará en función de las investigaciones y experimentos que se han realizado.

4.1. Solución propuesta

La solución que este trabajo propone para el problema de la SR consiste en una combinación de una arquitectura de una red neuronal apropiada (ver Sección 4.1.1) y un algoritmo que haga uso de ella (ver Sección 4.1.2).

4.1.1. Arquitecturas U-Net propuestas

Se ha modificado la función de activación por ELU [1], se ha hecho dropout [2], el pooling ha cambiado a average pooling y las imágenes de entrada son de 256×256. Estos cambios pueden verse en la Figura 4.1, exceptuando que las U-Net utilizadas han tenido 4 niveles.

Además de este cambio básico, se han propuesto (y ensayado sus resultados en la Sección 4.2.3 y 4.2.4) varias arquitecturas más basadas en la U-Net y los paradigmas de la Sección 3.2.2, como:

[1] https://sefiks.com/2018/01/02/elu-as-a-neural-networks-activation-function/
[2] https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

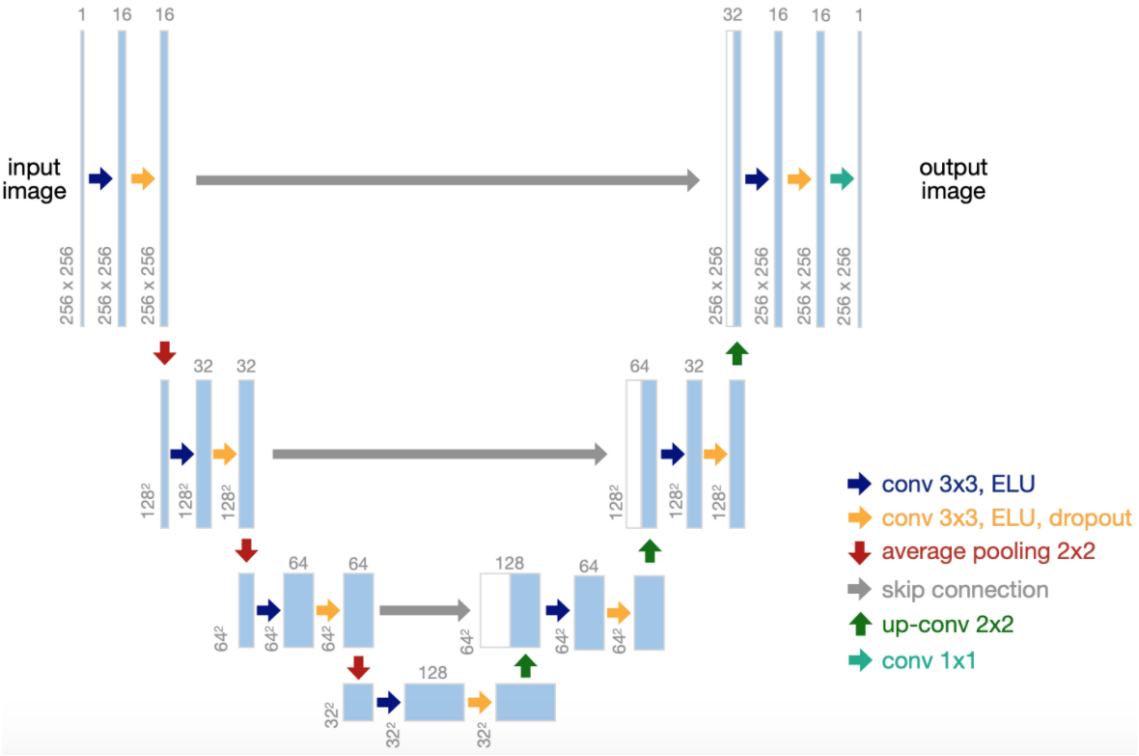

Figura 4.1: Una diferente configuración de la U-Net: con ELU, dropout, average pooling y una entrada de 256×256.

Fuente: Propia.

- preUNet: Una U-Net en la que se hace pre-upsampling.
- postUNet: Una U-Net en la que se hace post-upsampling.
- preResUNet: Una U-Net residual en la que se hace pre-upsampling.
- postResUNet: Una U-Net residual en la que se hace post-upsampling.
- prepostResUNet: Una U-Net residual en la que se hace la mitad del upsampling al principio y la otra mitad al final.

Los tres últimos son una U-Net donde el encoder es una ResNet34 (ver 2.2.2), como la de la Figura 3.13, y todas ellas se han probado con 4 niveles.

4.1.2. Algoritmo prototipo

Se introduce aquí la descripción del algoritmo que se ha utilizado en el proyecto. Se ha modificado en numerosas ocasiones, pero el esqueleto del algoritmo se ha mantenido (es

decir, se han cambiado sus parámetros, pero no los pasos).

Obtener el modelo

Son siete los pasos generales seguidos para entrenar y conseguir un modelo:

1. Leer y cargar las imágenes como grises.
2. Dividir las imágenes en recortes más pequeños de potencias de 2; por ejemplo, dividir una imagen de 1024×1024 en sus cuatro partes de 256×256.
3. Implementar una función llamada crappifier que añada ruido a las imágenes, como ruido gaussiano o S&P (ver Sección 2.1.2).
4. Para cada recorte crear uno igual pasado por el crappifier, es decir, lograr pares de imágenes recortadas LR-HR, con y sin crappifier.
5. Diseñar la U-Net con la configuración deseada, si es necesario con un upsampling al principio y escogiendo las funciones de activación correctas y el tamaño de entrada.
6. Preparar las imágenes para el entrenamiento: normalizarlas entre 0 y 1, y ponerles una tercera dimensión de modo que sean de dimensiones $N \times N \times 1$.
7. Entrenar el modelo con los hiperparámetros deseados, que se pueden escoger en la propia función de entrenamiento (fit [3]).

Evaluar el modelo

Una vez obtenido el modelo, para evaluar sus resultados se han seguido los siguientes cuatro pasos:

1. Coger el modelo y visualizar sus gráficas de sus funciones de loss como MSE y MAE en entrenamiento y validación.
2. Evaluar el modelo cuantitativamente con las métricas PSNR, SSIM o MS-SSIM (para la última, ver Sección 3.1.2).

[3] https://keras.rstudio.com/reference/fit.html

3. Evaluar el modelo cualitativamente comparando la imagen predicha por el modelo y la imagen objetivo.

4. Los modelos se guardan para poder replicar los experimentos si se necesita.

4.2. Experimentos

4.2.1. Conjuntos de datos y puntos de referencia

Han sido dos los datasets (utilizaremos esta palabra como sinónimo de conjunto de datos) utilizados: uno que se ha utilizado más y otro que se ha utilizado para probar los resultados contra el estado del arte.

Dataset de microscopía electrónica

El dataset que más se ha utilizado, fue creado por Lichtman Lab en la Universidad de Harvard, utilizado en [Kasthuri, 2015]. Se compone de imágenes obtenidas por un tipo de microscopio electrónico (concretamente ssEEM, de Scanning Electron Microscopy [4]) de córtex cerebral de ratón.

El dataset son dos conjuntos (uno de entrenamiento y otro de test) con 100 imágenes cada uno, siendo las imágenes de 8 bits, con dimensiones de 1024×1024 píxeles y resolución de $6 \times 6 \times 30$ nm.

Como ejemplo, en la Figura 4.2 se muestran dos imágenes de este dataset.

Este dataset se ha utilizado para la gran mayoría de experimentos (incluyendo todos los experimentos de la Sección 4.2.2), debido a que no eran demasiadas imágenes y a su vez eran suficientes para que la red aprendiera.

Con ellas, ha resultado sencillo evaluar cualitativamente los resultados obtenidos en cada experimento y ha sido el dataset más relevante. Su único problema es que, debido a que no se había utilizado con anterioridad para el problema de SR, no existía estado del arte contra el que compararse utilizando este dataset.

[4] https://es.wikipedia.org/wiki/Microscopio_electr%C3%B3nico_de_barrido

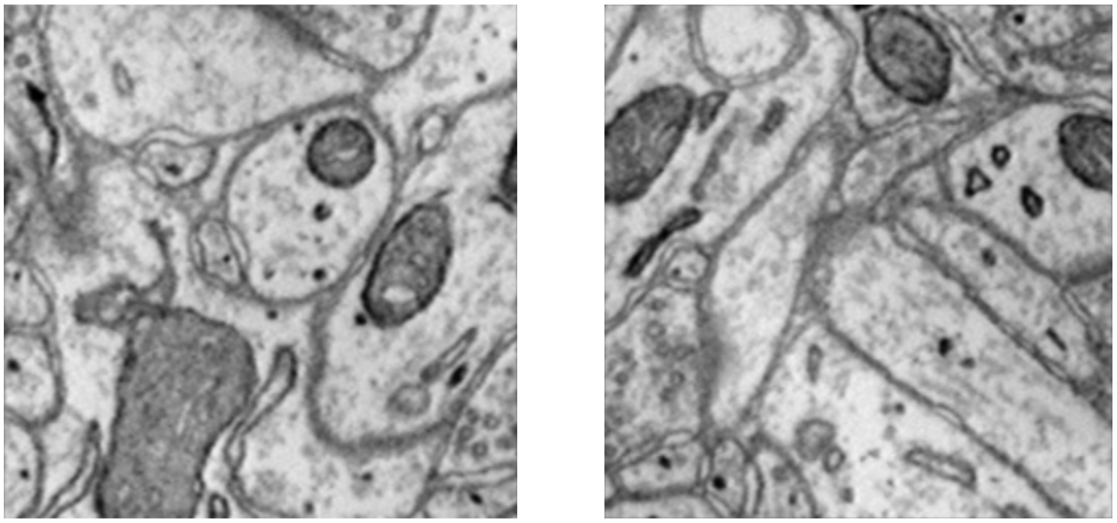

Figura 4.2: Dos imágenes del conjunto de datos EM (son en realidad el recorte de una misma imagen más grande).

Fuente: Propia.

Dataset ManorEM2019

El segundo dataset, de alrededor de 130GB, se compone de imágenes electrónicas de hipocampo de rata [5].

Los resultados obtenidos en PSNR y SSIM en [Fang et al., 2019], que se pueden ver en la Figura 4.3 o en la Figura 3.14b, han sido sobre este dataset, y son el punto a batir. Se trata de SRx4, concretamente, hacer SR pasando las imágenes de 128×128 a 512×512.

Concretamente, la media de los resultados en test de PSSR con respecto a PSNR ha sido entorno al 23 y de SSIM entorno al 0.65, aunque también se pueden observar outliers (valores atípicos) tanto por encima como por debajo de la media.

4.2.2. Búsqueda de hiperparámetros

Se trata de una búsqueda de hiperparámetros (grid search [6]) sobre el algoritmo original, es decir, se ha investigado cómo varía el rendimiento del modelo ante cambios sencillos de hiperparámetros como número de épocas o condición de parada.

Para ello, la primera tanda de experimentos (aproximadamente 20) ha consistido en variar ligeramente los hiperparámetros de la NN con criterios básicos sobre su elección, y se han

[5] https://dataverse.tdl.org/dataset.xhtml?persistentId=doi:10.18738/T8/YLCK5A
[6] https://scikit-learn.org/stable/modules/grid_search.html

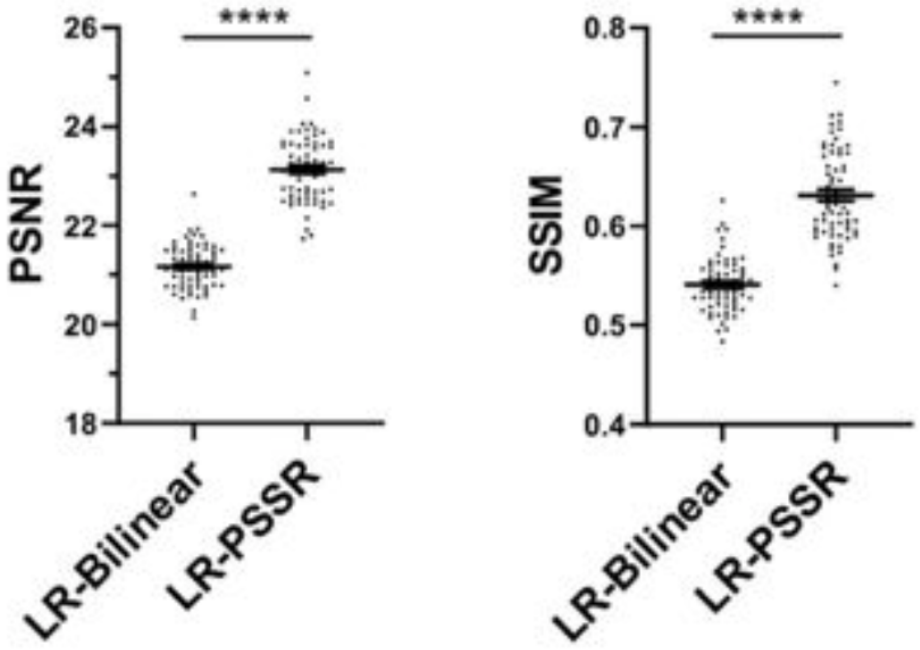

Figura 4.3: Resultados en PSNR y SSIM; se muestran sus resultados estadísticos con la media y valores atípicos.

Fuente: [Fang et al., 2019]

iterado los experimentos solamente si resultaban interesantes. Por otro lado, en la segunda tanda (otros 10 experimentos), se han realizado experimentos mediante a la construcción de nuevas combinaciones de hiperparámetros en base a las conclusiones de la primera parte.

En cada experimento, los datos se han recogido de la siguiente manera:

- Objetivo del experimento.
- Descripción de ideas a priori.
- Gráficas de MSE y MAE en aprendizaje y validación.
- Resultados de PSNR y SSIM.
- Tiempo de ejecución del aprendizaje.
- Evaluación cualitativa de los resultados.
- Otroos detalles importantes.

Imágenes	200
Imágenes en train	100
Imágenes en validación	0
Imágenes en test	100
Tamaño de crops	1024×1024
Crops en train	1440
Crops en validación	160
Crops en test	1600

Tabla 4.1: Características comunes de los experimentos en la Sección 4.2.2 y 4.2.3.

En cuanto a los hiperparámetros, estos han sido los que se han considerado, basándose en las opciones de la función fit de Keras:

- Épocas.
- Tamaños de batch.
- Learning rates.
- Optimizadores.
- Condiciones de parada.

En esta fase sólo se ha ensayado SRx4 (concretamente, de 64×64 a 256×256), sólo se han cambiado los hiperparámetros y no se ha tocado la red, que siempre ha sido la U-Net de la Figura 4.1 con 4 niveles. Además, en todos los experimentos son comunes las características de la Tabla 4.1, todos se han corrido en la GPU de Colab [7] y el dataset utilizado ha sido el de la Sección 4.2.1.

Resultados

Los resultados recopilados de todos los experimentos pueden verse en los anexos (ver Sección C).

[7] https://colab.research.google.com/notebooks/gpu.ipynb

Análisis de resultados

Los primeros 20 experimentos han sido búsqueda exploratoria de los resultados, pero para los últimos 10 experimentos los criterios para realizarse se han ido refinando: se han valorado los primeros experimentos y después, basándose en sus resultados se han planteado los nuevos.

Sin embargo, se debe decir que estos experimentos tenían un fallo de diseño: el conjunto de validación. En vez de escogerse por semilla (es decir, en vez de haber hecho que los conjuntos de entrenamiento y validación siempre fueran exactamente las mismas imágenes), el conjunto de validación se ha especificado en un porcentaje (10 % del conjunto de entrenamiento) y parece ser que en cada ejecución cada modelo ha entrenado y validado con diferentes imágenes, por lo que los resultados no son perfectos y, sobre todo, no es del todo lícito comparar un experimento con el otro, puesto que uno ha podido aprender con mayor dificultad que otro, resultando esto en mayor tiempo de ejecución pero mejor aprendizaje.

En cuanto a los resultados, se ha visto que ningún experimento (es decir, ninguna variación de hiperparámetros o combinación de variaciones) ha resultado en una mejora demasiado significativa con respecto al primer prototipo.

Sin embargo, sí se obtienen ciertas conclusiones, relacionadas con los hiperparámetros:

- Épocas: Los modelos obtienen resultados muy buenos con muy pocos epochs (alrededor de 20) y, pese a que sigan mejorando ya más ligeramente incluso hasta los 50, el tiempo de ejecución quizás no lo justifique.

- Condición de parada: Puesto que el entrenamiento no es extremadamente largo, puede no utilizarse. Si se hiciera, se recomienda utilizar una stop condition con paciencia alrededor de 5, puesto que, a partir de los 30 epochs, a menudo le cuesta mejorar los resultados MSE y MAE.

- Optimizador: Sin duda, los mejores modelos se han conseguido con Adam [8] (y sus variaciones cercanas como Adamax [9]) y RMSprop [10], siendo esta segunda la más estable, puesto que con Adam pueden surgir muchos saltos en el entrenamiento.

[8] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
[9] https://keras.io/api/optimizers/adamax/
[10] https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a

- Learning rate: Los cambios bruscos de learning rate empeoran significativamente los modelos y los cambios cortos no los hacen mejores, por lo que se recomienda utilizar los valores por defecto de learning rate de cada optimizador.

- Tamaño de batch: Todos los experimentos con tamaños de batch mayores de 5-8 han arrojado resultados negativos, mientras que con tamaños menores no existe diferencia considerable. Con todo, los mejores resultados los dan los tamaños 1 o 2.

4.2.3. Cambios en la arquitectura, el upsampling y la función de loss

El objetivo de la segunda fase evalúa cómo cambia el rendimiento de la red ante cambios en su arquitectura y también ante cambios en algunos hiperparámetros más complejos como funciones de loss o también aumento de datos (ver Sección 3.2.5).

Las ideas para desarrollar esta fase provienen directamente del artículo [Wang et al., 2020], el cual ofrecía una manera sistemática de probar y desarrollar modelos de DLSR. Se han seguido sus directrices como se muestra en la Figura 4.4, investigando partes de la taxonomía ofrecida en el artículo.

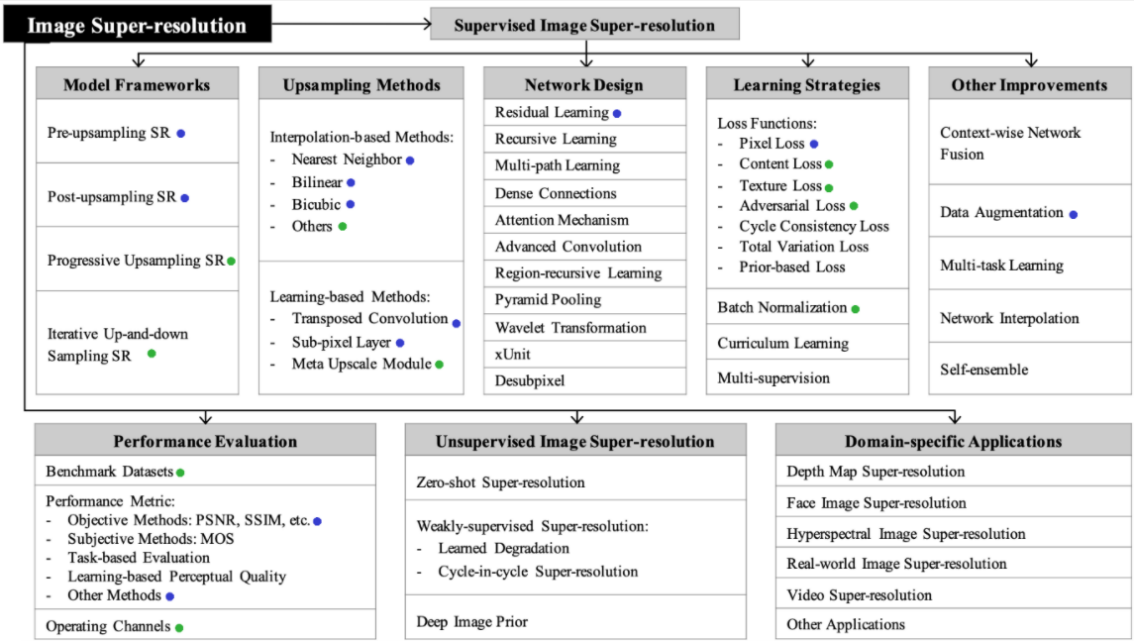

Figura 4.4: Experimentos de 4.2.3 en base a la taxonomía de [Wang et al., 2020]; en verde los apartados investigados y en azul los apartados investigados e implementados.
Fuente: [Wang et al., 2020] modificado.

Network	Learn upsamp	Data Augmen.	Loss	Optimizer	Learning rate	Batch size	PSNR	SSIM
pre U-Net 4	No	Flips	MSE	RMSprop	0.001	1	30.08536175474726	0.940639494363673
pre U-Net 4	No	Flips	MSE	RMSprop	0.001	4	27.97525171468447	0.9339049484419669
pre U-Net 4	No	Flips	MSE	RMSprop	0.001	8	27.871971056948922	0.9265746547472511
pre U-Net 4	Conv2DTranspose	Flips	MSE	RMSprop	0.001	1	31.346568504666962	0.9495415906097375
pre U-Net 4	SubpixelConv2D	Flips	MSE	RMSprop	0.001	1	31.47840056104983	0.950078727145805
pre U-Net 4	Conv2DTranspose	Flips	MAE	RMSprop	0.001	1	31.135559951104707	0.9489651833382099
pre U-Net 4	Conv2DTranspose	Flips	Multiscale SSIM	RMSprop	0.001	1	28.18823892107196	0.946146713443402
post U-Net 4	Conv2DTranspose	Flips	MSE	RMSprop	0.001	1	26.934498416141185	0.8996287572349664
post U-Net 4	SubpixelConv2D	Flips	MSE	RMSprop	0.001			
preResUNet 4	Conv2DTranspose	Flips	MSE	RMSprop	0.001	1	31.87002804644723	0.952240913729224
preResUNet 4	SubpixelConv2D	Flips	MSE	RMSprop	0.001	1	**32.02967881742632**	**0.9543686935925512**
preResUNet 4	SubpixelConv2D	Flips + Rots	MSE	RMSprop	0.001	1	31.972609624960043	0.9531994300778652
preResUNet 4	SubpixelConv2D	Flips + Rots	Multiscale SSIM	RMSprop	0.001	1	30.691666497552724	0.9535240685548629
preResUNet 4	Conv2DTranspose	Flips	Multiscale SSIM	RMSprop	0.001	1	30.74552561575781	0.9504236807004993
postResUNet 4	Conv2DTranspose	Flips	MSE	RMSprop	0.001	1	29.795946775725735	0.9263929519631147
postResUNet 4	SubpixelConv2D	Flips	MSE	RMSprop	0.001	1	30.357075404750876	0.9348586176944836
pre U-Net 4	No	Flips	MSE	Adam	0.001	1	27.097530320658123	0.887516776684588
pre U-Net 4	No	Flips	MSE	Adam	0.0001	1	25.94659519983512	0.897565223647682
pre U-Net 4	Conv2DTranspose	Flips	Multiscale SSIM	Adam	0.001	1	27.292037354313134	0.9381449717799708

Tabla 4.2: Resultados en SRx2.

Nuevamente, todos los experimentos han tenido las características de la Tabla 4.1, se han corrido en Colab y sobre el dataset de la Sección 4.2.1, pero esta vez se ha probado el funcionamiento de los modelos para SRx2 y SRx4 (es decir, de 128×128 a 256×256 y de 64×64 a 256×256).

En cada experimento, se han recogido menos datos de resultados que en la anterior fase, puesto que esta se ha centrado principalmente en optimizar las métricas PSNR y SSIM, y no tanto en las curvas de aprendizaje o en el tiempo de ejecución.

Por otro lado, el problema de la semilla mencionado en la Sección 4.2.2 se ha solucionado y todos los experimentos utilizan las mismas particiones de conjuntos, por lo que los resultados son más fiables y comparables entre ellos.

Resultados

En la Tabla 4.2 se muestran los resultado haciendo SRx2. Los parámetros iniciales han sido 16, el número de épocas de 50 y la condición de parada ha sido de 10 épocas sin cambios.

Network	Learn upsamp	Data Augmen.	Loss	Optimizer	PSNR	SSIM
pre U-Net 4	Conv2DTranspose	Flips	MSE	RMSprop	26.72883848437704	0.8497668978019084
preResUNet 4	Conv2DTranspose	Flips	MSE	RMSprop	26.661484804957592	0.8533425785164003
preResUNet 4	SubpixelConv2D	Flips	MSE	RMSprop	26.963524827177043	0.8591994030651623
preResUNet 4	SubpixelConv2D	Flips + Rots	MSE	RMSprop	**26.98195689368552**	0.8578117304724304
preResUNet 4	SubpixelConv2D	Flips	MAE	RMSprop	26.93888243518849	0.8577101388502509
preResUNet 4	SubpixelConv2D	Flips	Multiscale SSIM	RMSprop	26.702280712181764	0.8606867852945526
preResUNet 4	SubpixelConv2D	Flips + Rots	MIX (alpha=0.5)	RMSprop	26.972478395407023	0.8616802999963804
preResUNet 4	SubpixelConv2D	Flips + Rots	MIX (alpha=0.75)	RMSprop	26.95213826974417	0.8615363529196247
preResUNet 4	SubpixelConv2D	Flips + Rots	MIX (alpha=0.84)	RMSprop	26.916436520077404	**0.8618897536980452**
prePostResUNet 4	SubpixelConv2D	Flips	MSE	RMSprop	26.76728020998355	0.8524503827526808

Tabla 4.3: Resultados en SRx4.

Por otro lado, en la Tabla 4.3, se muestran los resultado haciendo SRx4. Los parámetros iniciales han sido 16, el número de épocas de 50, la condición de parada ha sido de 10 épocas sin cambios, y en este caso el tamaño de batch ha estado estático en 1, así como el learning rate en 0.001.

En ambos casos, el loss llamado Mix es una mezcla parametrizada de los loss MS-SSIM (ver Sección 3.1.2) y PSNR (ver Sección 3.1.1), propuesta en [Zhao et al., 2015]. El upsampling *No* (aparece en rojo) se corresponde con un upsampling tradicional (concretamente, la función Upsampling2D [11]), en este caso, la interpolación por vecinos más cercanos [12]; mientras las interpolaciones aprendidas (Conv2DTranspose y SubpixelConv2D), se corresponden con las interpolaciones transpuesta y subpíxel de la Sección 3.2.3. Por último, los nombres de las redes utilizadas son los mismos que en la Sección 4.1.1.

Análisis de resultados

Como se puede observar, en el caso de la SR de 2x, los resultados son muy positivos, y se consigue reconstruir las imágenes a HR de manera que es difícil distinguir cualitativamente la imagen predicha por el modelo y la imagen original.

Los resultados en la métrica PSNR parecen ser más sensibles que los de SSIM, puesto que se mueven desde alrededor de 25 hasta alrededor 32, mientras que los de SSIM oscilan de alrededor de 0.88 a alrededor de 0.95. De hecho, las conclusiones más claras son,

[11] https://www.tensorflow.org/api_docs/python/tf/keras/layers/UpSampling2D
[12] https://es.wikipedia.org/wiki/Interpolación_por_el_vecino_más_cercano

precisamente, visibles en PSNR: que el mejor tamaño de batch es el 1 y que el mejor learning rate es 0.001 (ambos fenómenos también pasaban en la Sección 4.2.2).

En el resto de factores, las ideas a extraer resultan más discutibles:

- Red utilizada: En condiciones iguales, la preU-Net obtiene resultados considerablemente mejores que los de la postU-Net, ligeramente mejores que los de la postResU-Net y ligeramente peores que los de la preResU-Net, que es la red que mejores resultados obtiene y menor variabilidad ha mostrado.

- Tipo de upsampling: Pese a que el UpSampling2D puede obtener resultados positivos, parece ser que existe un techo por lo alto al que sólo los dos upsamplings aprendidos llegan y además estos oscila siempre entre resultados más altos. Por otro lado, tanto el UpSampling2D como el Conv2DTranspose obtienen a veces resultados muy bajos hasta los que SubpixelConv2D no baja, pero, en condiciones iguales, no se aprecia una gran diferencia entre los upsampling aprendidos.

- Data augmentation: Muy probablemente, su uso ha mejorado los resultados de los modelos significativamente. Sin embargo, no parece que haya diferencia entre el data augmentation con flips (voltear) o con flips y rots (rotar). El mejor resultado se ha conseguido con el primero, pero las dos únicas veces que se ha utilizado se han conseguido muy buenos resultados (aunque creemos que esto es debido a que se ha utilizado SubpixelConv2D en esos experimentos).

- Función de loss: Al contrario de lo que se podría creer, el loss MSE sigue mostrándose como el más fiable incluso en las medidas de SSIM, donde se podría creer que el loss MS-SSIM debería optimizarlo más. MS-SSIM, de hecho, optimiza SSIM con muy buenos resultados, pero decae ligeramente en sus medidas en PSNR, por lo que su uso no parece ser necesario. En cuanto a MAE, los resultados son prácticamente idénticos (muy ligeramente peores) a los de MSE en el único experimento en el que se ha probado, por lo que se podría usar indistintamente o investigar más su uso.

- Optimizador: Se evidencia una gran diferencia entre Adam y RMSprop, siendo la primera la que peores resultados ha arrojado. Pese a que Adam no se ha probado con SubpixelConv2D, dos experimentos en los que la única variante ha sido el optimizador (Adam frente a RMSprop), es decir, en condiciones iguales, han arrojado resultados muy diferentes a favor de RMSprop.

En cuanto a los resultados en SR de 4x, se pueden aplicar todas las conclusiones esta lista y añadir dos apuntes:

- El loss MIX parece optimizar SSIM correctamente, siendo el que mejores resultados ha obtenido en SSIM a medida que se le subía el peso del parámetro alpha (que indica cuánto de valor se le da a MS-SSIM en su cálculo interno). Sin embargo, subir este parámetro implica directamente bajar sus resultados en PSNR, porque se le quita peso al cálculo de MSE.
- La arquitectura prepostResU-Net no mejora los resultados de preResU-Net en su experimento en condiciones iguales, pero tampoco lo empeora en demasía. Se requeriría más investigación.

4.2.4. Comparación con el estado del arte

Como prueba final, se ha utilizado el dataset de la Sección 4.2.1 para poder compara directamente los resultados de nuestras redes con las del estado del arte en [Fang et al., 2019]. Para ello, no se ha podido entrenar en Colab y se han tenido que utilizar máquinas con la GPU Titan Xp.

Resultados

Estas pruebas se han realizado dos veces; la primera arrojó unos resultados (que se muestran en la Tabla 4.4), pero resultó que en test los resultados de las métricas empeoraban debido a que las imágenes no eran del mismo tamaño y se tuvo que solucionar mediante solapamiento de imágenes.

En cambio, en TensorFlow2 [13], ese problema se solucionaba, ya que las redes no piden como parámetro de entrada los tamaños. Los resultados se muestran en la Tabla 4.5.

En ambas tablas se han mantenido las siguientes características:

- Red: preResUnet4.
- Data augmentation: Flips+rots.
- Learning-rate: 0.001.

[13] https://www.tensorflow.org/guide/effective_tf2

Loss function	Optimizer	Batch size	# Epochs	PSNR (val)	SSIM (val)	PSNR (test)	SSIM (test)	Time per epoch (s)
MSE	RMSprop	1	20	24.98378092	0.6751944638	23.47808601	0.6422273663	2637
MSE	RMSprop	16	20	24.92387176	0.6743562141	23.38547814	0.6428640466	2107
MAE	RMSprop	16	20	24.93147937	0.6730207042	23.45027998	0.6430451343	2082
MAE	Adam	16	20	24.93607278	0.6724600733	23.45058547	0.6428346297	2050
MAE	Adam	16	30	24.95234531	0.6734499427	23.48251954	0.6435962757	2141
MSE	Adam	16	30	24.94351912	0.6732394508	**23.54762377**	**0.6453462033**	2080
MsSSIM	Adam	16	30	24.57341533	0.6757932266	23.12806765	0.6425516618	2498
MAE	Adam	8	20	24.98899962	0.674566391	23.50644071	0.6440590027	4304

Tabla 4.4: Primeros resultados de nuestros modelos para el dataset de la Sección 4.2.1.

Loss function	Optimizer	Batch size	# Epochs	PSNR (val)	SSIM (val)	PSNR (test)	SSIM (test)	Time per epoch (s)
MSE	RMSprop	1	20	**24.98378092**	0.6751944638	24.10370309	0.660285592	2637
MSE	RMSprop	16	20	24.92387176	0.6743562141	24.05572476	0.663296492	2107
MAE	RMSprop	16	20	24.93147937	0.6730207042	24.06842261	0.6615858892	2082
MAE	Adam	16	20	24.93607278	0.6724600733	24.08371757	0.6613059881	**2050**
MAE	Adam	16	30	24.95234531	0.6734499427	24.11410239	0.6624970804	2141
MSE	Adam	16	30	24.94351912	0.6732394508	**24.12194525**	**0.6628168596**	2080
MsSSIM	Adam	16	30	24.57341533	**0.6757932266**	23.7315578	0.6653553727	2498

Tabla 4.5: Resultados definitivos de nuestros modelos para el dataset de la Sección 4.2.1.

- Condición de parada: 10 épocas.

Análisis de resultados

Como se puede ver en la Tabla 4.5, los resultados obtenidos superan al estado del arte [Fang et al., 2019]. Concretamente, en test, se han conseguido (ver Tabla 5.1 para más información):

- PSNR: 24.12194525097956.
- SSIM: 0.6628168595878045.

En ambos casos, se superan los objetivos de la Figura 4.3; aproximadamente 24.1 frente a aproximadamente 23 en PSNR y aproximadamente 0.66 frente a aproximadamente 0.65 en SSIM. Además, la arquitectura que ofrecemos es más sencilla que en [Fang et al., 2019]. Se pueden medir los resultados obtenidos de forma cualitativa, viendo las imágenes de los anexos (ver Sección B).

En cuanto a los hiperparámetros, surgen nuevas conclusiones:

- Tamaño de batch: Los tamaños de batch más grandes (16 o 32) hacen a los modelos entrenar más rápido sin empeorar los resultados. Esto puede ser debido a que el dataset utilizado (ver Sección 4.2.1) es mucho más pesado.

- Función de loss: No se aprecian diferencias entre MSE y MAE, y MS-SSIM optimiza ligeramente más SSIM en detrenimiento de sus resultados en PSNR.
- Optimizador: Tanto RMSprop como Adam han funcionado bien, sin diferencias evidentes entre ellos.

5. CAPÍTULO

Conclusiones

En este último capítulo, se da un cierre al trabajo, exponiendo sus conclusiones más importantes.

5.1. Sobre el trabajo

Como resumen final, se comenta lo que se ha conseguido, en lo que se ha trabajado paralelamente y lo que se planea hacer en el futuro.

5.1.1. Hitos del trabajo

En esta investigación se han intentado en principio emular los resultados de un artículo sobre SR en microscopía [Fang et al., 2019]. Para ello, se ha investigado el estado del arte del problema (ver Sección 3), principalmente con el artículo [Wang et al., 2020] y se ha experimentado (ver Sección 4.2) sobre una solución (ver Sección 4.1).

Los resultados obtenidos en SR por factor de 2 son muy positivos, mientras que los resultados en SR por un factor de 4 (lo cual es un reto mayor) superan el artículo que se quería emular (ver Sección 4.2.4, ver Tabla 5.1), resultados que se pueden comprobar cualitativamente en los anexos (ver Sección B). Además, y sobre todo, se ofrece una solución considerablemente más sencilla para el mismo problema.

Tipo de red	preResU-Net4
Data augmentation en train	Flips + rots
Función de loss	MSE
Optimizador	Adam
Tamaño de batch	16
Learning rate	0.001
Número de épocas	160
Criterio de parada	10 épocas
PSNR en validación	2494351912
SSIM en validación	0.6732394508
PSNR en test	24.12194525
SSIM en test	0.6628168596
Tiempo por época (segundos)	2080
GPU	TitanXp

Tabla 5.1: Mejor resultado sobre el dataset de la Sección 4.2.1 .

Debido a esto, se está trabajando en una publicación que recoja de manera adecuada estos resultados.

5.1.2. Líneas paralelas

Para comprender bien el contexto de este trabajo, es importante destacar que, pese a que no han sido mencionados en el resto de la memoria, este de TFG (su investigación tanto en estado del arte como con los experimentos) ha abierto numerosas otras investigaciones que se han y se están desarrollando todavía.

Para empezar, el estudio profundo del estado del arte en SR general en combinación con el SR para microscopía, nos ha permitido entender cuáles son los trabajos hechos y cuáles son las tendencias futuras, haciendo que empecemos a trabajar en una futura revisión del tema.

Por otro lado, el estudio de las métricas comenzó como una parte más del estudio del estado del arte, pero derivó en una investigación exhaustiva sobre sus propiedades y cómo se podrían combinar y modificar para utilizarlas como funciones de loss más eficaces. Esto ha derivado en otra revisión sobre las métricas.

Por último, descubrir las propiedades no tan conocidas de PSNR y SSIM han terminado por un estudio en métodos formales donde se han demostrado matemáticamente esas

propiedades y después se han verificado automáticamente por computador, mediante a la herramienta Dafny [1].

5.1.3. Trabajo futuro

Pese a que el trabajo se haya desarrollado durante un curso entero, existen varios inconclusos que se terminarán en unos pocos meses:

- Trabajar en la publicación y refinar los modelos con nuevos cambios.
- Trabajar en el resto de líneas paralelas.
- Investigar las tendencias en SR para microscopía y en las métricas.

Además de esto, cabe destacar que la SR por DL es un campo realmente nuevo y que, por tanto, se puede esperar que todavía haya importantes hitos que lo hagan encaminarse hacia un lado o hacia otro en función de sus propuestas. Por ello, es más difícil que en otros campos de estudio saber con certeza qué trabajo futuro deparará el DLSR.

5.2. Conclusiones personales

En lo personal, este TFG ha supuesto un reto en diferentes aspectos como el académico o el de organización.

En lo académico, el trabajo ha estado estrechamente relacionado con lo que se enseña en asignaturas de Aprendizaje Automático y Visión por Computador, temas sobre los cuales he tenido opción de ampliar mis conocimientos y contra los que he tenido que batallar para no perderme en ellos. Por otro lado, todo el conocimiento teórico se ha visto reforzado por la práctica, que ha sido documentarse e implementar programas en Python, resultando en que algunos de esos algoritmos no fueran triviales de encontrar o hacer. Asimismo, quisiera destacar que asignaturas formales como Matemática Discreta, Estructuras de Datos y Algoritmos, Métodos Formales o toda la especialidad de Computación me han dado una capacidad de abstracción esencial para facilitarme la tarea de entender cada concepto nuevo que afrontaba.

[1] microsoft.com/en-us/research/project/dafny-a-language-and-program-verifier-for-functional-correctness/

Además del conocimiento académico, el TFG ha aportado numerosos beneficios transversales, especialmente en organización. Pienso firmemente que no habría sido posible compaginar este TFG con el resto de quehaceres como el trabajo laboral, de no ser por una gestión plena como proyecto, que me ha permitido conocer poco a poco cómo yo trabajo conmigo mismo y con otros para así optimizar la consecución de mis objetivos. Por lo que al TFG le agradezco haberme ayudado también en esa formación tan importante que es difícil integrar como asignatura y que toda la carrera lleva enseñándome.

En el ámbito más personal, estoy satisfecho porque considero que se han conseguido los objetivos, se han superado, y además me llevo enseñanzas muy importantes de cara a futuro, tanto si escojo el camino de la investigación como si no.

Anexos

A. ANEXO

Gestión del proyecto

Este anexo sirve como documentación a nivel de memoria, donde se podrá ver por encima cuál ha sido la gestión del trabajo a nivel de proyecto. Para desarrollar esta parte se ha basado en las directrices del PMI en su libro PMBoK [1], fundamentado en la tabla de su Figura A.1, utilizando las notas de la web [2].

A.1. Inicio

A.1.1. Descripción del proyecto

Este es un proyecto de investigación del área de las ciencias de la computación, cuyos objetivos son, por un lado, investigadr la teoría y aplicaciones de SR para imágenes de microscopía y, por otro lado, redactar una memoria de TFG basado en dicha investigación.

Concretamente, es un proyecto de naturaleza remota, donde los tutores (en la Facultad de Informática de Donostia) y el alumno (en Madrid), tratarán de implementar una arquitectura de NN capaz de reconstruir imágenes LR de microscopía en imágenes HR. Para ello, se estudiarán los fundamentos de la SR y se implementarán algoritmos prototípicos que evolucionarán en función de lo estudios del estado del arte y los experimentos que se realicen.

[1] https://www.edu.xunta.gal/centros/cfrpontevedra/aulavirtual2/pluginfile.php/13688/mod_folder/content/0/libros_pmbok_guide5th_spanish.pdf?forcedownload=1

[2] https://www.gladysgbegnedji.com/

El proyecto consta de dos hitos principales [3]:

- Hito externo: Es la última fecha posible de entrega del TFG (subirlo a la plataforma Addi), 2 de febrero del 2020.
- Hito interno: Es la última fecha que internamente se ha marcado como fecha para la primera revisión entregable del TFG, el 15 de enero del 2020.

Asimismo, se prevee una posible segunda iteración del proyecto (con su consiguiente incremento del alcance) en caso de que se reciba la beca de investigación Ikasiker.

A.1.2. Descripción de interesados

***Definición* A.1** *El propósito de este proceso es, además de identificar a todas las personas y organizaciones afectadas por el proyecto, documentar cualquier información relevante acerca de su interés, influencia, actitud y compromiso con el éxito del mismo* [4].

Los interesados internos se han identificado rápidamente:

- Alumno: Es el interesado principal, puesto que, además de servir para formarse en SR y en el mundo de la investigación, la realización del TFG es imprescindible para terminar el grado. Sus roles serán investigar e implementar. Su nivel de participación será el más alto.
- Turores: Doctores en informática y directores de este y otros proyectos. Sus roles serán asistir al alumno, guiarlo en caso de problemas y contactar con otros colaboradores. Su nivel de participación será muy alto.
- Ricardo Andrade [5]: Director de microscopía analítica en SGIker. Sus roles serán aportar conjuntos reales de datos y realizar consultas como posible consumidor. Su nivel de participación será bajo.
- Otros colaboradores: Podría ser que se necesitara de nuevos colaboradores en momentos puntuales del proyecto.

[3] https://www.sinnaps.com/blog-gestion-proyectos/hito-la-gestion-proyectos
[4] https://www.gladysgbegnedji.com/identificar-a-los-interesados/
[5] https://www.ehu.eus/es/web/sgiker/mikroskopia-analitikoa-kontaktua

Además, se han identificado algunos interesados externos relevantes:

- Grupo CVPD [6]: Interesados porque esta es una investigación de VC en la facultad, y sobre un tema puntero como lo es la SR.
- Laboratorios biomédicos: Interesados porque las contribuciones en SR suponen una ventaja instantánea a la hora de abaratar costes en laboratorios de microscopía.
- Investigadores en SR: Interesados porque, si la investigación del proyecto resulta en conclusiones interesantes, se habrá contribuido en el campo.

A.2. Planificación

A.2.1. Planificación del alcance

Recopilación de requisitos

***Definición* A.2** *El proceso recopilar requisitos, consiste en documentar las necesidades, deseos y expectativas cuantificadas y documentadas de los interesados para convertirlas en requisitos del proyecto* [7].

A nivel de proyecto, se tienen los siguientes requisitos:

- Requisitos de organización: Mantener una comunicación activa debido a que se va a trabajar en remoto.
- Requisitos de entrega: Fecha de entrega marcada en febrero.

A nivel de producto, se tienen los siguientes requisitos:

- Requisitos técnicos: Documentarse sobre el estado del arte e implementar código funcional que realice SR sobre imágenes de microscopía.
- Requisitos de seguridad: No revelar código o datasets privados.

[6] https://www.ehu.eus/es/web/enpresa/transferentzia-eta-berrikuntza-eskaintza-arloaren-arabera/-/asset$_{p}ublisher/L5fcxCxBETI1/content/cvpd?inheritRedirect = false$s

[7] https://www.gladysgbegnedji.com/recopilar-requisitos/

Definición del alcance

***Definición* A.3** *Definir el alcance del proyecto es el proceso que consiste en desarrollar una descripción detallada del proyecto y del producto* [8].

Sobre el proyecto, es fundamental establecer que este es un trabajo de investigación en el marco de la visión por computador y el aprendizaje automático, de apróximadamente seis meses de duración y prorrogable a otra iteración en caso de obtenerse una beca de investigación.

Por otro lado, se generan dos productos principales:

- La implementación de una NN que, a la que, dándosele un conjunto de entrenamiento de imágenes de microscopía electrónica a baja y a alta resolución, aprende a aumentar la resolución de las imágenes; hecho que se pueda probar contrastando el modelo contra un conjunto de test.
- Una memoria de TFG que describa de manera científica los pasos principales y resultados más destacables del proyecto.

Creación de la EDT/WBS

***Definición* A.4** *La estructura de desglose de trabajo (EDT) o Work Breakdown Structure (WBS), es una herramienta que se utiliza para describir el alcance de un proyecto según sus entregables. Los cuales dividiremos en componentes lo suficientemente pequeños y manejables que nos permita planificar de manera fácil el proyecto. Estos componentes del último nivel de descomposición se denominan Paquetes de Trabajo* [9].

Para poder crear la EDT (ver Figura A.2), se debe tener un mapa claro de qué patas y subpatas va a tener el proyecto, patas que se basan en los productos mencionados en la Sección A.2.1, además de en la gestión.

- Memoria: Serán una serie de bocetos que evolucionarán incrementalmente el uno del otro.

[8] https://www.gladysgbegnedji.com/definir-el-alcance/
[9] https://www.gladysgbegnedji.com/crear-la-edt-wbs/

- Investigación:
  - Bases: Son una serie de lecturas que hay que comprender y aprender de ellas.
  - Algoritmo prototipo: Diferentes versiones de un algoritmo que evolucionará hasta ser funcional.
  - Estado del arte: Estudio y lectura del artículo principal [Fang et al., 2019] y su código con el que comprender mejor el estado del arte.
  - Algoritmo mejorado: Nuevas versiones del algoritmo prototipo reforzadas por el conocimiento del estado del arte.
  - Experimentos: Se ensayarán diferentes experimentos sobre el algoritmo definitivo, para ver qué tal funcionan los modelos.
- Gestión:
  - Planificación: Un estudio detallado sobre el proyecto, sus objetivos y cómo alcanzarlos.
  - Control: El control integral del proyecto en todos los ámbitos como tiempo, comunicación, replanificación...

A.2.2. Planificación del tiempo

Definición de las actividades

***Definición* A.5** *Definir las actividades consiste en identificar las acciones que deben ser llevadas a cabo para conseguir los entregables del proyecto. Después de crear la EDT, obtenemos el nivel más bajo de esta descomposición, el cual denominamos Paquetes de trabajo. La descomposición de éstos, en componentes más pequeños nos proporciona las actividades necesarias para ejecutar los paquetes de trabajo* [10].

Para cada entregable de (ver Figura A.2), se han definido una serie de actividades conocidas (aunque se espera que vayan a ser más):

- En Memoria: Cada boceto es una actividad.

[10] https://www.gladysgbegnedji.com/definir-las-actividades-del-proyecto/

- En Investigación:
  - Bases: Cada lectura es una actividad.
  - Algoritmo prototipo: Cada versión es una actividad.
  - Estado del arte: Cada apartado estudiado del artículo es una actividad.
  - Algoritmo mejorado: Cada versión es una actividad.
  - Experimentos: Cada experimento es una actividad.
- En Gestión:
  - Planificación: Cada apartado de la planificación es una actividad.
  - Control: Cada sesión de control es una actividad.

Secuenciación de las actividades

***Definición* A.6** *Secuenciar las actividades del proyecto, consiste en determinar las dependencias entre actividades, es decir, qué relación de ejecución existe entre ella, en qué secuencia se ejecutan. Cada una de las actividades o hitos del cronograma tiene al menos una actividad sucesora o predecesora, a excepción de la primera y la última.*

Cuando se ha presentado el EDT en la Sección A.2.1, ya se ha tenido en mente una división temporal y las dependencias entre entregables se ven de arriba abajo en la Figura A.2; es decir, los entregables de la abajo dependen del los que se encuentran inmediatamente encima de ellos.

Concretamente, tomando las siguientes actividades como no definitivas:

- En Memoria: Cada boceto es una nueva iteración del anterior.
- En Investigación:
  - Bases: Existen unas lecturas principales (qué es la U-Net -ver Sección 2.2.2-, cómo usar Keras[11]...), pero después aparecerán nuevas derivadas de esas primeras.
  - Algoritmo prototipo: Cada versión es una iteración del anterior.

[11] https://keras.rstudio.com/

  - Estado del arte: Se debe comenzar por comprender sus logros, después los métodos y después las implementaciones.
  - Algoritmo mejorado: Cada versión es una iteración del anterior.
  - Experimentos: Al igual que en las bases, los primeros experimentos no tienen relación entre sí, pero los siguientes serán derivados de los anteriores.

- En Gestión:
  - Planificación: Cada apartado de la planificación es una actividad.
  - Control: Cada sesión de control es una actividad.

Estimación del tiempo actividades

***Definición* A.7** *El proceso de estimar la duración de las actividades es utilizar información sobre el alcance del trabajo de cada actividad y sus relaciones para estimar el cronograma* [12].

Si nos centramos en las actividades de ambos productos:

- En Memoria: Cada boceto tendrá un mes de desarrollo.
- En Investigación:
  - Bases: Las primeras lecturas deberían ser de dos semanas, el resto todavía no se saben.
  - Algoritmo prototipo: Cada versión se desarrollará durante una semana o dos.
  - Estado del arte: El entendimiento de los métodos tendrán tres semanas de duración y el estudio de la implementación dos semanas.
  - Algoritmo mejorado: Cada versión se desarrollará durante una a tres semanas.
  - Experimentos: Cada semana se habrán realizado unos 10 experimentos.

Recopilando estos tiempos, nos queda que el proyecto tendrá una duración estimada que se muestra en la Tabla A.1.

[12] https://www.gladysgbegnedji.com/estimar-la-duracion/

Paquete de trabajo	Duración (horas)
Gestión	**40**
Planificación	10
Monitoreo y control	30
Investigación	**210**
Bases	30
Algoritmo prototipo	50
Estado del arte	40
Algoritmo mejorado	40
Experimentos	50
Memoria	**50**
Total	**300**

Tabla A.1: Estimación de la duración del proyecto en base a sus paquetes de trabajo.

Desarrollo del cronograma

***Definición* A.8** *Desarrollar el cronograma del proyecto consiste en integrar los procesos anteriores para crear el cronograma del proyecto, el cual determina las fechas de comienzo y de fin para las actividades planificadas. Suele ser iterativo porque muy probablemente requiera de una o varias revisiones de las estimaciones para resultar realista* [13].

Integrando la Sección A.2.2, A.2.2 y A.2.2, se presenta un cronograma del proyecto en la Figura A.3, hecho en la herramienta VisualParadigm [14].

A.2.3. Planificación de riesgos

Identificación de riesgos

***Definición* A.9** *Identificar los riesgos es un proceso iterativo que se actualiza en cada nueva fase, ya que se pueden descubrir nuevos riesgos o pueden evolucionar conforme el proyecto avanza a lo largo de su ciclo de vida* [15].

Los riesgos encontrados para el proyecto no son muchos ni se preveen graves, pero se describen aquí:

[13] https://www.gladysgbegnedji.com/desarrollar-el-cronograma-del-proyecto/
[14] https://online.visual-paradigm.com/es/
[15] https://www.gladysgbegnedji.com/identificar-los-riesgos-del-proyecto/

- Necesidad demasiado alta de computación: Si las máquinas locales o las máquinas en la nube no fueran capaces de entrenar con pesados conjuntos de datos como en la Sección 4.2.1.
- Resultados negativos o difusos en experimentación: Si los experimentos no arrojaran resultados concluyentes, ya fuera porque no se logran resultados positivos o porque descifrar estos resultados resulta no ser trivial.
- Dificultad teórica o práctica: Si comprender ciertos conceptos o implementarlos exijiera más esfuerzos de los previstos como más documentación o reuniones.
- Pérdida de información: Si se perdiera parcial o totalmente el avance (ficheros, código, ideas...) hecho a lo largo de un tiempo.

Análisis de riesgos

***Definición* A.10** *El análisis de riesgos incluye los métodos para priorizar los riesgos identificados, mediante la definición de niveles de probabilidad e impacto* [16] [17].

Sobre los riesgos identificados en la Sección A.2.3, se hará un análisis sobre sus probabilidades e impactos.

Sobre las probabilidades:

- Necesidad demasiado alta de computación: Sobre los datos de microscopía electrónica (ver Sección 4.2.1) la probabilidad es baja, mientras que sobre los datos del artículo [Fang et al., 2019] es probable que se necesiten máquinas más potentes de alguna manera.
- Resultados negativos o difusos en experimentación: Es altamente probable que los resultados no sean lo suficientemente parecidos al artículo que se pretende emular, debido a que es una publicación reciente y puntera.
- Dificultad teórica o práctica: La probabilidad de que entender los conceptos suponga un riesgo es baja, debido a que ya se cuenta con la base para ello; pero la probabilidad de que las implementaciones cuesten es mediana, puesto que nunca antes se ha manejado este tipo de redes (ver Sección 2.2.2 y 2.2.2).

[16] https://www.gladysgbegnedji.com/realizar-el-analisis-cualitativo-de-riesgos/
[17] https://www.gladysgbegnedji.com/realizar-el-analisis-cuantitativo-de-riesgos/

- Pérdida de información: Es muy baja la probabilidad de que información mínimamente relevante se pierda, debido a que todo se comparte en la nube de una manera u otra; sin embargo, la probabilidad de que ideas interesantes queden en el olvido es alta, porque se mantendrán muchas comunicaciones sobre muchos temas.

Sobre los impactos que supondrían:

- Necesidad demasiado alta de computación: Si surgiera el caso de una necesidad de máquinas más potentes, implicaría dificultades para entrenar los modelos o comparar nuestros resultados, pero no supondrían un punto muerto en el proyecto.
- Resultados negativos o difusos en experimentación: Obtener resultados negativos no es de por si malo, puesto que permiten tanto averiguar qué no funciona, como trazar nuevas líneas experimentales; mientras que los resultados difusos sí suponen un gran problema, debido a que dificultan la realización de nuevos experimentos, al no saber interpretar los anteriores.
- Dificultad teórica o práctica: Una dificultad para entender la teoría podría retrasar el inicio de las implementaciones, mientras que una dificultad en las implementaciones podría retraras el inicio de la evaluación de los modelos (y, por tanto, de los trabajos), por lo que ambos casos se deben tener en cuenta.
- Pérdida de información: Una pérdida relevante de información sería catastrófica en términos de retrabajo y cumplir con las entregas y con el cronograma de la Figura A.3, mientras la pérdida de ideas interesantes no afectaría en demasía a este proyecto.

Respuesta a los riesgos

***Definición* A.11** *La respuesta a los riesgos desarrolla las opciones y acciones para mejorar las oportunidades y reducir las amenazas a los objetivos del proyecto, teniendo en cuenta que las respuestas a los riesgos deben ser congruentes con la importancia del riesgo y tener un coste efectivo relacionado con su desafío* [18].

Debido a que los riesgos propuestos en la Sección A.2.3 no son especialmente complejos, sus métodos de mitigación y prevención son también sencillos:

[18] https://www.gladysgbegnedji.com/planificar-la-respuesta-a-los-riesgos-2/

- Necesidad demasiado alta de computación: En caso de necesitarse más poder de cómputo, se prevee en uso de máquinas en línea o máquinas más potentes en la facultad. Como mitigación también existe la posibilidad de trabajar con conjuntos de datos menos pesados.

- Resultados negativos o difusos en experimentación: Para prevenir que esto suceda, se basarán las implementaciones en fuentes contrastadas, y se leerá el estado del arte para comprender las salidas. Como mitigación está el hecho de que un resultado malo o difuso no deja de ser un resultado más que no desmerece el trabajo realizado.

- Dificultad teórica o práctica: En caso de estancamiento en teoría o código, se podrá consultar con los tutores, diferentes fuentes o expertos. Como mitigación, existe la posibilidad de pasar a otras tareas mientras se resuelven los atascos.

- Pérdida de información: La manera de evitar las pérdidas importantes es mediante el manejo de varios repositorio diferentes, al igual que la pérdida de información menos relevante. Como mitigación, se debe tener en cuenta que los datos perdidos se pueden recuperar (ya que no se partiría desde el principio de nuevo) y que las ideas olvidadas no son ideas esenciales, al menos para este proyecto.

A.2.4. Otras planificaciones

Planificación de calidad

***Definición* A.12** *Para planificar la calidad, se deben establecer métricas para medirla* [19].

Para evitar el retrabajo, cada producto tendrá sus propios criterios o métricas de calidad.

- En Memoria: Los bocetos se evaluarán subjetivamente en base al criterio de los tutores.

- En Investigación:

    - Bases: La comprensión de las lecturas se medirá en base a la cantidad de inconclusos que surjan.

[19] https://www.gladysgbegnedji.com/gestion-de-la-calidad/

- Algoritmo prototipo: Mediante métricas para evuluar el rendimiento de las implementaciones y los modelos.
- Estado del arte: De la misma manera que en las bases.
- Algoritmo mejorado: De la misma manera que el algoritmo prototipo.

Los algoritmos (sus implementaciones) son el elemento que métricas más sofisticadas require. Por un lado, existen las métricas sobre sus características (velocidad, documentación, elegancia...); y por otro lado, métricas (ver Sección 3.1) sobre los modelos que se generan en las implementaciones.

Planificación de comunicaciones

***Definición* A.13** *Se debe asegurar que los interesados correctos reciban la información adecuada en tiempo y forma* [20].

Para definir un plan de comunicaciones, se debe hablar primeramente sobre cómo se van a comunicar los interesados internos del proyecto:

- Reuniones: Esencial para hacer las puestas en común, las reuniones físicas se efectuarán una vez por mes. Se debatirá sobre lo hecho y, sobre todo, funcionará para planificar los siguientes pasos. En caso de no poder reunirse, estarán abiertas las vías online como Skype o Hangouts. En cualquier caso, las reuniones seguirán la metodología PCWF [21] por parte del alumno, para aumentar su utilidad.
- Correo electrónico: El canal más utilizado, debido a la naturaleza en remoto del proyecto. La mayoría de consultas técnicas se realizarán por este medio, así como resolución de dudas o contacto con otros interesados externos.
- Teléfono: En casos de consultas rápidas o de urgencia, se podrá contactar vía teléfono.

Además de ello, es indispensable establecer un sistema de información (en este caso, digital) con el que trabajar:

[20] https://www.gladysgbegnedji.com/gestionar-las-comunicaciones/
[21] https://www.youtube.com/watch?v=xW0CjH95K3Q

- Máquinas particulares: Para almacenar algunos documentos de poco interés, para trabajar las implementaciones sin necesidad de conexión o para entrenar redes con un peso demasiado grande, se utilizarán las máquinas particulares.
- Google Drive: Toda la información de mayor interés (las implementaciones de Google Collaboratory [22], los documentos informativos, las tablas excel de los experimentos...) se publicará en las carpetas compartidas de Google Drive, de manera que interactivamente se puedan ver y revisar.
- Overleaf [23]: En caso de que se decida escribir la memoria en el sistema de composición LaTex [24], se utilizará este repositorio online que funciona de manera similar a Google Drive.

A.3. Control

Sobre el control del proyecto, sólo se ha considerado interesante destacar el hecho de que se ha realizado una segunda iteración del proyecto, debido a que la beca de investigación fue concedida. Por tanto, como parte del monitoreo en alcance y tiempo, se modificó tanto la EDT de la Figura A.2 como el cronograma de la Figura A.3, resultando en nuevos EDT y cronograma que se muestran en las Figura A.4 y A.5 respectivamente.

A.4. Finalización

***Definición* A.14** *Cerrar el proyecto es el proceso que consiste en finalizar todas las actividades para completar formalmente el proyecto; se deben asegurar de que todo el trabajo del proyecto está completo y de que se han alcanzado los objetivos* [25].

Pese a que la investigación en sí misma no se haya concluído, se decidió a finales de mayo que la parte perteneciente a este proyecto se acabaría.

Por tanto, se recopilaron todos los resultados de la investigación (así como sus repositorios más importantes), se clasificaron y se redactó la memoria de TFG, que sirve también como memoria muy resumida del proyecto.

[22] https://colab.research.google.com/notebooks/welcome.ipynb?hl=es
[23] https://en.wikipedia.org/wiki/Overleaf
[24] https://es.wikipedia.org/wiki/LaTeX
[25] https://www.gladysgbegnedji.com/cerrar-el-proyecto-o-fase/

En cuanto a los interesados, el contacto se mantiene, debido a que se seguirá trabajando en los mismos objetivos fuera del proyecto.

Procesos de un Área de Conocimiento	Grupos de Procesos de Dirección de Proyectos				
	Grupo de Procesos de Iniciación	Grupo de Procesos de Planificación	Grupo de Procesos de Ejecución	Grupo de Procesos de Seguimiento y Control	Grupo de Procesos de Cierre
4. Gestión de la Integración del Proyecto	Desarrollar el Acta de Constitución del Proyecto 3.2.1.1 (4.1) Desarrollar el Enunciado del Alcance del Proyecto Preliminar 3.2.1.2 (4.2)	Desarrollar el Plan de Gestión del Proyecto 3.2.2.1 (4.3)	Dirigir y Gestionar la Ejecución del Proyecto 3.2.3.1(4.4)	Supervisar y Controlar el Trabajo del Proyecto 3.2.4.1 (4.5) Control Integrado de Cambios 3.2.4.2 (4.6)	Cerrar Proyecto 3.2.5.1 (4.7)
5. Gestión del Alcance del Proyecto		Planificación del Alcance 3.2.2.2 (5.1) Definición del Alcance 3.2.2.3 (5.2) Crear EDT 3.2.2.4 (5.3)		Verificación del Alcance 3.2.4.3 (5.4) Control del Alcance 3.2.4.4 (5.5)	
6. Gestión del Tiempo del Proyecto		Definición de las Actividades 3.2.2.5 (6.1) Establecimiento de la Secuencia de las Actividades 3.2.2.6 (6.2) Estimación de Recursos de las Actividades 3.2.2.7 (6.3) Estimación de la Duración de las Actividades 3.2.2.8 (6.4) Desarrollo del Cronograma 3.2.2.9 (6.5)		Control del Cronograma 3.2.4.5(6.6)	
7. Gestión de los Costes del Proyecto		Estimación de Costes 3.2.2.10 (7.1) Preparación del Presupuesto de Costes 3.2.2.11 (7.2)		Control de Costes 3.2.4.6 (7.3)	
8. Gestión de la Calidad del Proyecto		Planificación de Calidad 3.2.2.12 (8.1)	Realizar Aseguramiento de Calidad 3.2.3.2 (8.2)	Realizar Control de Calidad 3.2.4.7 (8.3)	
9. Gestión de los Recursos Humanos del Proyecto		Planificación de los Recursos Humanos 3.2.2.13 (9.1)	Adquirir el Equipo del Proyecto 3.2.3.3 (9.2) Desarrollar el Equipo del Proyecto 3.2.3.4 (9.3)	Gestionar el Equipo del Proyecto 3.2.4.8 (9.4)	
10. Gestión de las Comunicaciones del Proyecto		Planificación de las Comunicaciones 3.2.2.14 (10.1)	Distribución de la Información 3.2.3.5 (10.2)	Informar el Rendimiento 3.2.4.9 (10.3) Gestionar a los Interesados 3.2.4.10 (10.4)	
11. Gestión de los Riesgos del Proyecto		Planificación de la Gestión de Riesgos 3.2.2.15 (11.1) Identificación de Riesgos 3.2.2.16 (11.2) Análisis Cualitativo de Riesgos 3.2.2.17 (11.3) Análisis Cuantitativo de Riesgos 3.2.2.18 (11.4) Planificación de la Respuesta a los Riesgos 3.2.2.19 (11.5)		Seguimiento y Control de Riesgos 3.2.4.11 (11.6)	

Figura A.1: Correspondencia entre grupos de procesos y áreas de conocimiento de la dirección de proyectos, según PMBok.

Fuente: https://manolosangoquiza.wordpress.com/2008/03/10/correspondencia-de-los-procesos-de-direccin-de-proyectos/

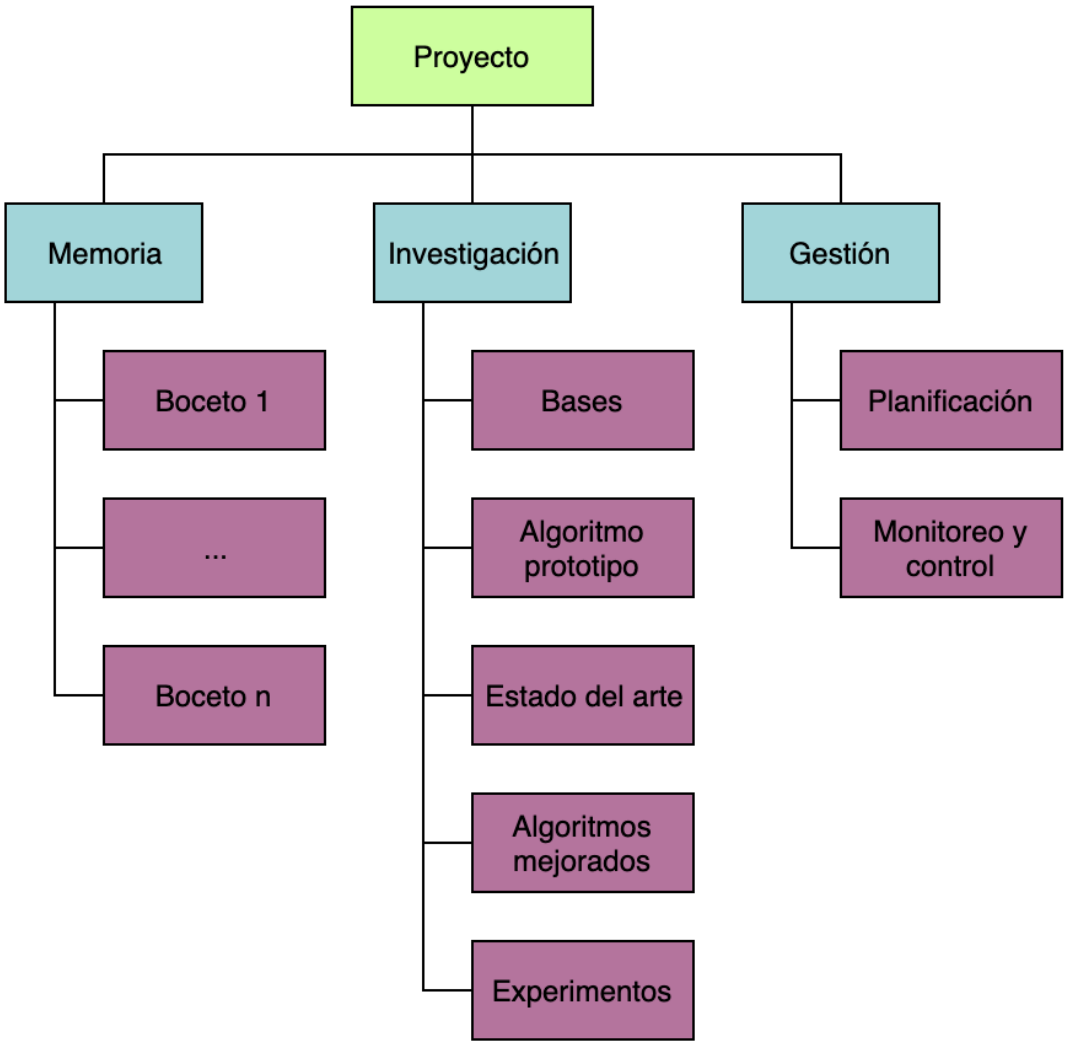

Figura A.2: EDT/WBS simplificado del proyecto

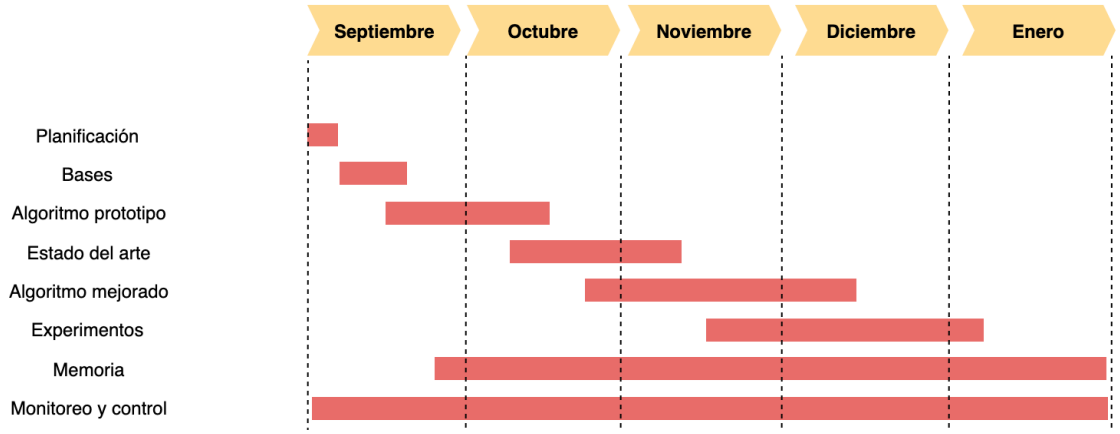

Figura A.3: Cronograma simplificado del proyecto

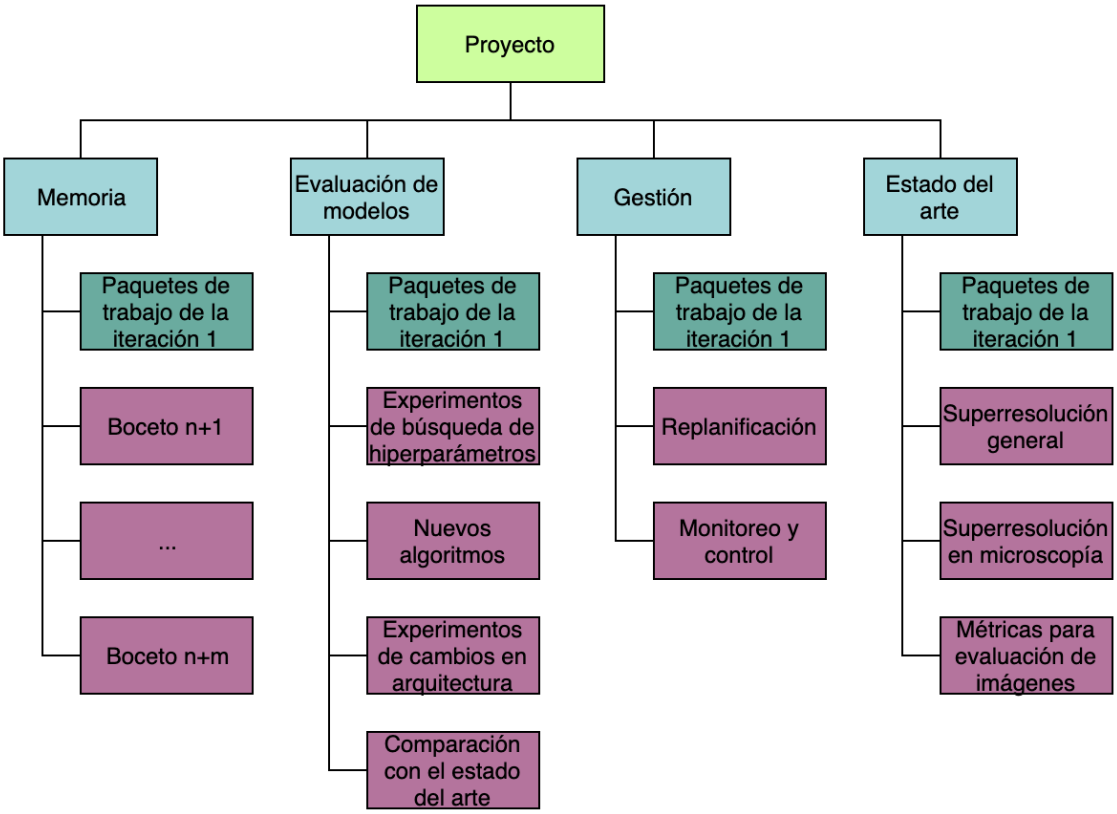

Figura A.4: EDT/WBS simplificado de la segunda iteración del proyecto

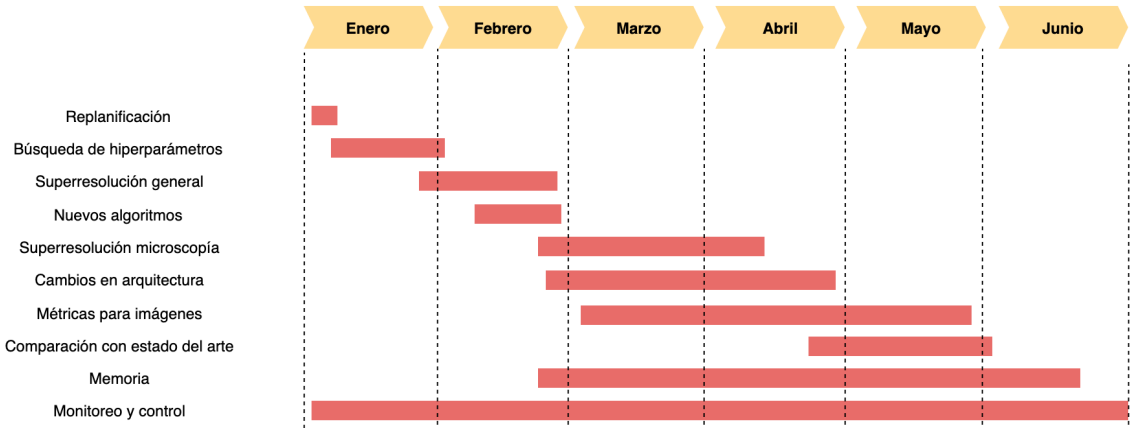

Figura A.5: Cronograma simplificado de la segunda iteración del proyecto

B. ANEXO

Comparación de imágenes

En este segundo anexo se muestran las imágenes generadas por los modelos, para que puedan ser contrastadas con la imagen LR+ruido y la imagen HR objetivo. Para compararlas mejor, se puede usar la herramienta ImageDiff [1].

Se debe destacar que, pese a que las imágenes a LR si están en su tamaño original, las imágenes HR (tanto las de los modelos como las originales) están submuestreadas para que entren en las páginas.

Las siguientes imágenes en SR4x se corresponden con los experimentos en la Sección 4.2.4.

B.1. Imagen 1

La Figura B.1 es la imagen 1 a LR con ruido, la Figura B.2 es la reconstrución de nuestro modelo y la Figura B.3 es la imagen objetivo.

B.2. Imagen 2

La Figura B.4 es la imagen 2 a LR con ruido, la Figura B.5 es la reconstrución de nuestro modelo y la Figura B.6 es la imagen objetivo.

[1] https://www.diffchecker.com/image-diff

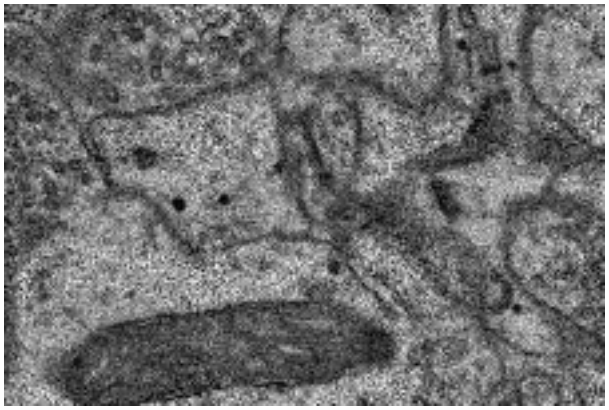

Figura B.1: Imagen 1 a LR con ruido.

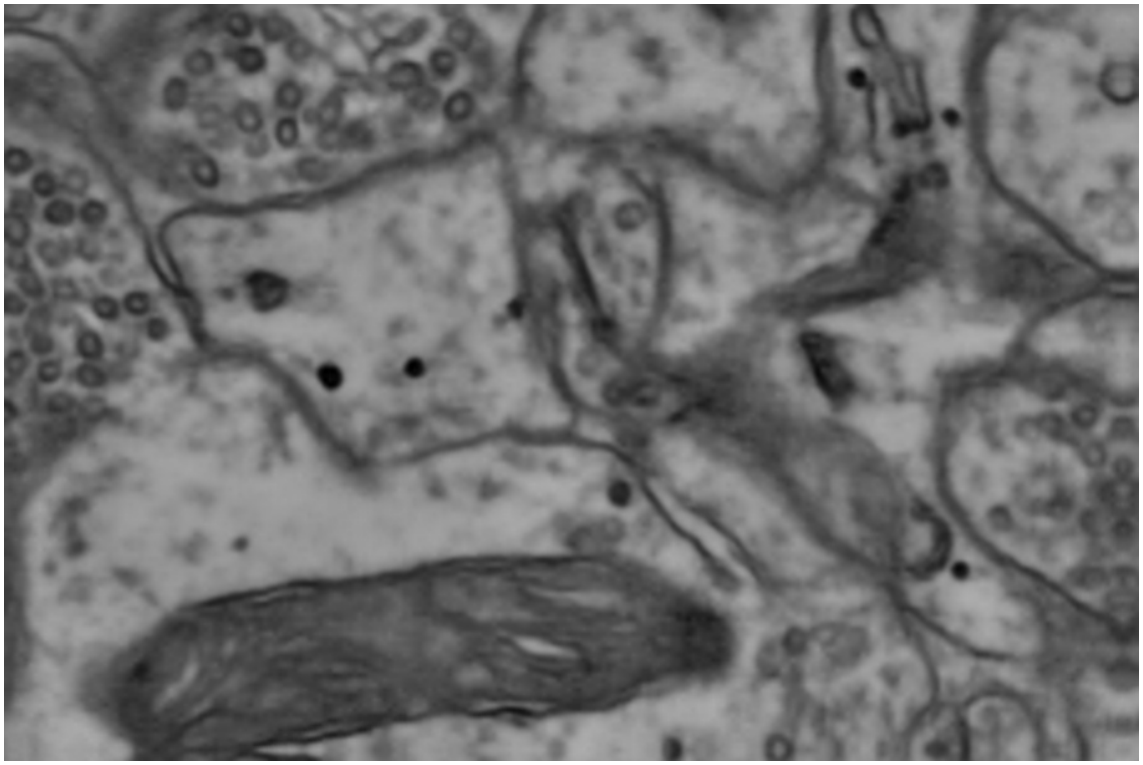

Figura B.2: Imagen 1 reconstruida a HR por el modelo.

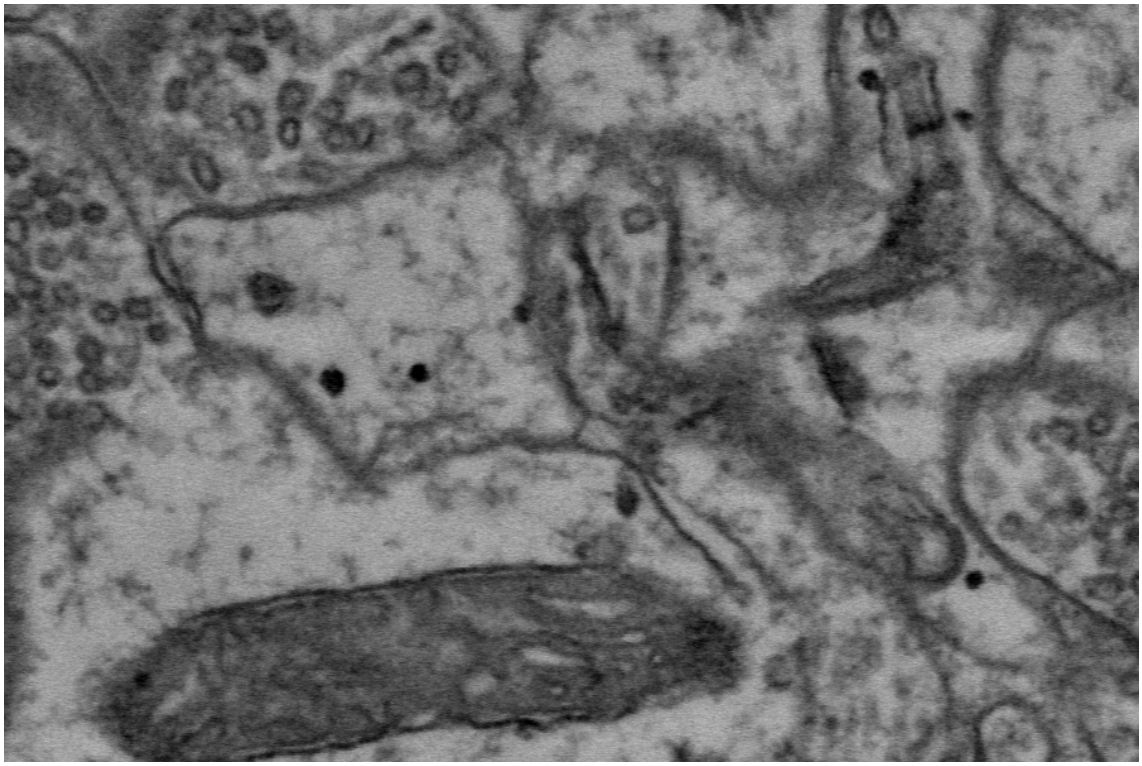

Figura B.3: Imagen 1 original a HR (el objetivo).

Figura B.4: Imagen 2 a LR con ruido.

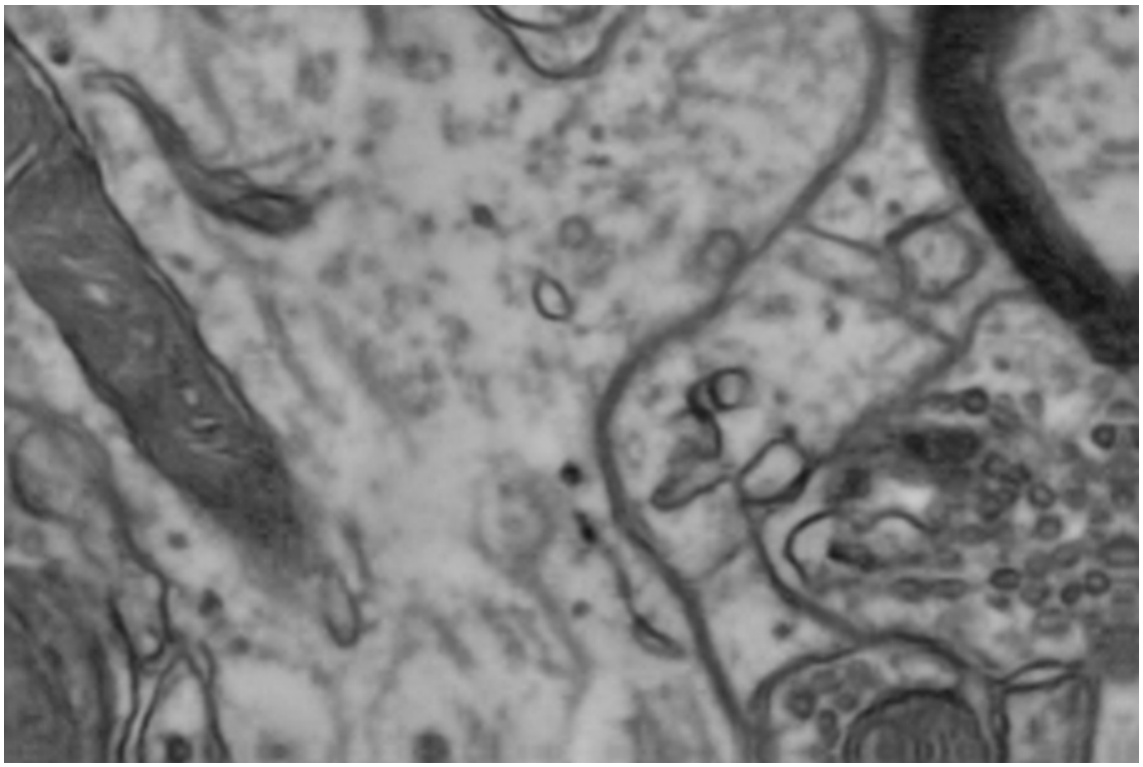

Figura B.5: Imagen 2 reconstruida a HR por el modelo.

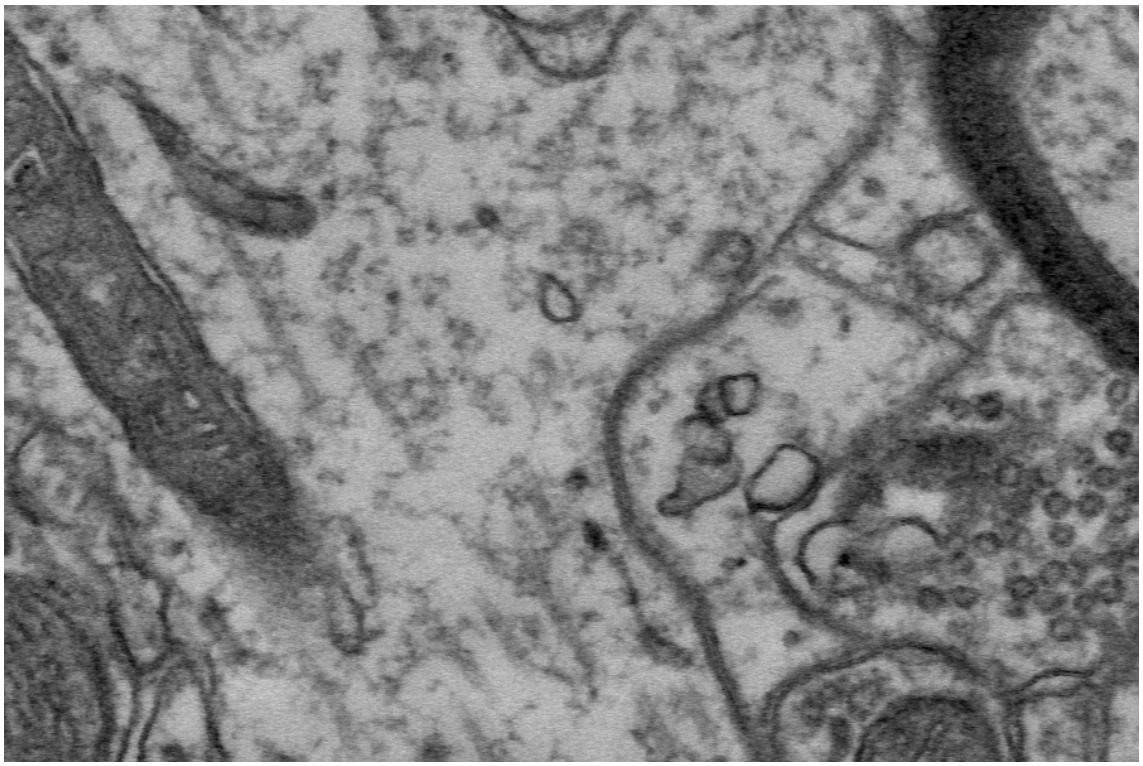

Figura B.6: Imagen 2 original a HR (el objetivo).

C. ANEXO

Tabla de experimentos de búsqueda de hiperparámetros

Este último anexo recopila los resultados de los experimentos de la Sección 4.2.2, debido a que era una tabla de Excel muy larga.

ID	Comments	# of epoch	Stop condition	Optimizer	Learning rate	Batch size	AVERAGE Train loss	AVERAGE Validation loss	AVERAGE Test loss	AVERAGE Train MAE	AVERAGE Validation MAE	AVERAGE Test MAE	AVERAGE PSNR score	AVERAGE SSIM score	AVERAGE Time per epoch (s)	AVERAGE Train total time (s)
Prototipo	**Versión prototipo.**	20	loss patience=20, ==numEpochs	ADAM	??	1	-	-	-	-	-	-	-		-	-
Exp 1	**Prototipo con epochs = 50**	50	loss patience=50 = numEpochs	ADAM	0.001; Adam por defecto	1	**1,10E-03**	**8,87E-04**	**1,12E-03**	**2,55E-02**	**2,31E-02**	**2,55E-02**	**29,6915**	**9,23E-01**	2,60E+01	1,30E+03
Exp 2	**Prototipo con epochs = 30**	30	loss patience=30, ==numEpochs	ADAM	0.001; Adam por defecto	1	1,20E-03	9,47E-04	1,38E-03	2,67E-02	2,38E-02	2,81E-02	28,8114	9,06E-01	2,70E+01	8,10E+02
Exp 3	**Prototipo con batch size = 5**	20	loss patience=20 = numEpochs	ADAM	0.001; Adam por defecto	5	2,10E-03	1,60E-03	2,10E-03	3,52E-02	3,04E-02	3,55E-02	27,0162	8,73E-01	1,80E+01	3,60E+02
Exp 4	**Prototipo con batch size = 32**	20	loss patience=20, ==numEpochs	ADAM	0.001; Adam por defecto	32	4,00E-03	3,20E-03	5,22E-03	4,87E-02	4,40E-02	5,71E-02	23,1373	7,60E-01	**1,60E+01**	3,20E+02
Exp 5	**Prototipo con batch size = 64 y epochs = 50**	50	loss patience=50 = numEpochs	ADAM	0.001; Adam por defecto	64	4,60E-03	3,30E-03	5,16E-03	5,24E-02	4,44E-02	5,81E-02	23,1894	7,36E-01	**1,60E+01**	8,00E+02
Exp 6	**Prototipo con batch size = 128 y epochs = 100**	100	loss patience=100, ==numEpochs	ADAM	0.001; Adam por defecto	128	3,10E-03	2,30E-03	3,81E-03	4,26E-02	3,68E-02	4,90E-02	24,5145	8,00E-01	**1,60E+01**	1,60E+03
Exp 7	**Prototipo con learning rate = 0.0001**	20	loss patience=20 = numEpochs	ADAM	0.0001	1	2,10E-03	1,50E-03	1,99E-03	3,48E-02	3,04E-02	3,44E-02	27,2219	8,74E-01	2,40E+01	4,80E+02
Exp 8	**Prototipo con learning rate = 0.000001**	20	loss patience=20 = numEpochs	ADAM	0.000001	1	7,00E-03	4,20E-03	8,36E-03	6,53E-02	5,02E-02	7,33E-02	21,2984	6,59E-01	2,50E+01	5,00E+02
Exp 9	**Prototipo con learning rate = 0.0000001 y epochs = 50**	50	loss patience=50 = numEpochs	ADAM	0.0000001	1	1,48E-02	2,07E-02	2,74E-02	9,64E-02	1,25E-01	1,47E-01	15,6596	5,51E-01	2,40E+01	1,20E+03
Exp 10	**Prototipo con learning rate = 0.01**	20	loss patience=20 = numEpochs	ADAM	0.01	1	2,77E-01	3,25E-01	2,45E-01	4,90E-01	5,41E-01	4,49E-01	6,141	2,07E-02	2,40E+01	4,80E+02
Exp 11	**Prototipo con learning rate = 0.1**	20	loss patience=20 = numEpochs	ADAM	0.1	1	3,01E-01	2,08E-01	3,92E-01	5,14E-01	4,24E-01	5,96E-01	4,2773	1,09E-03	2,40E+01	4,80E+02
Exp 12	**Prototipo con optimizer = adadelta**	20	loss patience=20 = numEpochs	Adadelta	1.0; Adadelta por defecto	1	2,60E-03	1,80E-03	2,73E-03	3,90E-02	3,28E-02	4,13E-02	25,8768	8,52E-01	2,50E+01	5,00E+02
Exp 13	**Prototipo con optimizer = adagrad**	20	loss patience=20 = numEpochs	Adagrad	0.01; Adagrad por defecto	1	2,87E-01	2,85E-01	2,96E-01	5,00E-01	5,00E-01	5,00E-01	5,3018	1,26E-02	2,20E+01	4,40E+02
Exp 14	**Exp 13 con learning rate = 0.001**	20	loss patience=20 = numEpochs	Adagrad	0.001	1	3,70E-03	3,50E-03	5,78E-03	4,67E-02	4,73E-02	6,26E-02	22,641	7,85E-01	2,20E+01	4,40E+02
Exp 15	**Prototipo con optimizer = RMSprop**	20	loss patience=20 = numEpochs	RMSprop	0.001; RMSprop por defecto	1	1,70E-03	1,20E-03	1,56E-03	3,16E-02	2,63E-02	3,02E-02	28,2486	9,00E-01	2,20E+01	4,40E+02

ID	Comments	# of epoch	Stop condition	Optimizer	Learning rate	Batch size	AVERAGE Train loss	AVERAGE Validation loss	AVERAGE Test loss	AVERAGE Train MAE	AVERAGE Validation MAE	AVERAGE Test MAE	AVERAGE PSNR score	AVERAGE SSIM score	AVERAGE Time per epoch (s)	AVERAGE Train total time (s)
Exp 16	**Prototipo con optimizer = SGD**	20	loss patience=20 = numEpochs	SGD	0.01; SGD por defecto	1	6,30E-03	5,00E-03	6,70E-03	6,15E-02	5,60E-02	6,54E-02	21,9037	6,70E-01	2,10E+01	4,20E+02
Exp 17	**Exp 16 con learning rate = 0.001**	20	loss patience=20 = numEpochs	SGD	0.001	1	9,80E-03	8,00E-03	8,99E-03	7,76E-02	7,26E-02	7,58E-02	20,6094	6,08E-01	2,40E+01	4,80E+02
Exp 18	**Prototipo con optimizer = Adamax**	20	loss patience=20 = numEpochs	Adamax	0.002; Adamax por defecto	1	1,70E-03	1,30E-03	1,65E-03	3,16E-02	2,77E-02	3,10E-02	28,0317	8,90E-01	2,40E+01	4,80E+02
Exp 19	**Exp 12 con learning rate = 10**	20	loss patience=20 = numEpochs	Adadelta	10	1	2,86E-01	2,85E-01	2,96E-01	5,00E-01	5,00E-01	5,00E-01	5,3074	1,27E-02	2,50E+01	5,00E+02
Exp 20	**Exp 12 con learning rate = 0.01**	20	loss patience=20 = numEpochs	Adadelta	0.01	1	6,90E-03	4,90E-03	6,41E-03	6,50E-02	5,43E-02	6,26E-02	22,1616	6,62E-01	2,30E+01	4,60E+02
Exp 21	**Prototipo con batch-size = 2**	20	loss patience=20 = numEpochs	ADAM	0.001; Adam por defecto	2	1,70E-03	1,30E-03	1,69E-03	3,20E-02	2,81E-02	3,11E-02	27,2219	8,74E-01	**1,50E+01**	**3,00E+02**
Exp 22	**Exp 21 con epochs = 50**	50	loss patience=50 = numEpochs	ADAM	0.001; Adam por defecto	2	**1,10E-03**	**8,65E-04**	**1,12E-03**	**2,60E+02**	**2,29E-02**	**2,56E-02**	**29,695**	**9,23E-01**	2,00E+01	4,00E+02
Exp 23	**Prototipo con batch-size = 3**	20	loss patience=20 = numEpochs	ADAM	0.001; Adam por defecto	3	1,90E-03	1,50E-03	1,83E-03	3,33E-02	2,97E-02	3,27E-02	27,5771	8,86E-01	**1,20E+01**	**2,40E+02**
Exp 24	**Exp 23 con epochs = 50**	50	loss patience=50 = numEpochs	ADAM	0.001; Adam por defecto	3	1,50E-03	1,20E-03	1,58E-03	3,01E-02	2,64E-02	3,03E-02	28,2234	8,94E-01	2,40E+01	1,20E+03
Exp 25	**Prototipo con batch-size = 4**	20	loss patience=20 = numEpochs	ADAM	0.001; Adam por defecto	4	1,90E-03	1,60E-03	2,08E-03	3,39E-02	3,09E-02	3,53E-02	27,1112	8,78E-01	1,80E+01	3,60E+02
Exp 26	**Prototipo con learning rate = 0.003**	20	loss patience=20 = numEpochs	ADAM	0.003	1	3,01E-01	2,08E-01	3,92E-01	5,14E-01	4,24E-01	5,96E-01	4,2773	1,09E-03	2,40E+01	4,80E+02
Exp 27	**Prototipo con learning rate = 0.004**	20	loss patience=20 = numEpochs	ADAM	0.004	1	2,72E-01	3,61E-01	2,01E-01	4,85E-01	5,76E-01	4,04E-01	7,423	3,83E-01	2,50E+01	5,00E+02
Exp 28	**Exp 27 con patience = numEpochs / 4 = 5**	20	loss patience= numEpochs / 4 = 5	ADAM	0.004	1	2,20E-03	1,60E-03	1,87E-03	3,61E-02	3,14E-02	3,24E-02	27,5097	8,77E-01	3,40E+01	3,60E+02
Exp 29	**Prototipo con learning rate = 0.002**	20	loss patience=20 = numEpochs	ADAM	0.002	1	2,00E-03	1,50E-03	2,00E-03	3,41E-02	3,00E-02	3,47E-02	27,2132	8,77E-01	2,70E+01	5,40E+02
Exp 30	**Experimento 15 con optimizer = RMSprop**	20	loss patience=20 = numEpochs	RMSprop	0.002	1	1,60E-03	1,20E-03	2,88E-02	3,14E-02	2,63E-02	2,88E-02	**28,6215**	**9,03E-01**	6,60E+01	1,32E+03
Exp 31	**Experimento 15 con optimizer = RMSprop**	50	loss patience=20 = numEpochs	RMSprop	0.001; RMSprop por defecto	1	**1,20E-03**	**1,10E-03**	**1,29E+15**	**2,69E-02**	**2,57E-02**	**3,47E-02**	**29,0324**	**9,24E-01**	3,20E+01	1,60E+03

Bibliografía

[Ahn et al., 2018] Ahn, N., Kang, B., and Sohn, K.-A. (2018). Fast, accurate, and lightweight super-resolution with cascading residual network.

[Bai et al., 2018] Bai, Y., Zhang, Y., Ding, M., and Ghanem, B. (2018). *SOD-MTGAN: Small Object Detection via Multi-Task Generative Adversarial Network: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, pages 210–226.

[Bei et al., 2018] Bei, Y., Damian, A., Hu, S., Menon, S., Ravi, N., and Rudin, C. (2018). New techniques for preserving global structure and denoising with low information loss in single-image super-resolution.

[Belthangady and Royer, 2019] Belthangady, C. and Royer, L. A. (2019). Applications , promises , and pitfalls of deep learning for fluorescence image reconstruction.

[Bruhn et al., 2004] Bruhn, A., Weickert, J., and Schnörr, C. (2004). Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61:211–231.

[Brunet et al., 2012] Brunet, D., Vrscay, E. R., and Wang, Z. (2012). On the mathematical properties of the structural similarity index. *IEEE Transactions on Image Processing*, 21(4):1488–1499.

[Bulat and Tzimiropoulos, 2017] Bulat, A. and Tzimiropoulos, G. (2017). Super-fan: Integrated facial landmark localization and super-resolution of real-world low resolution faces in arbitrary poses with gans.

[Chamier et al., 2019] Chamier, L., Laine, R., and Henriques, R. (2019). Artificial intelligence for microscopy: What you should know. *Biochemical Society Transactions*, 47:BST20180391.

[Chen et al., 2018] Chen, R., Qu, Y., Zeng, K., Guo, J., Li, C., and Xie, Y. (2018). Persistent memory residual network for single image super resolution. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 922–9227.

[Chen et al., 2017] Chen, Y., Tai, Y., Liu, X., Shen, C., and Yang, J. (2017). Fsrnet: End-to-end learning face super-resolution with facial priors.

[Dai et al., 2015] Dai, D., Wang, Y., Chen, Y., and Gool, L. V. (2015). Is image super-resolution helpful for other vision tasks?

[Denk and Horstmann, 2004] Denk, W. and Horstmann, H. (2004). Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS biology*, 2:e329.

[Dong et al., 2014] Dong, C., Loy, C. C., He, K., and Tang, X. (2014). Learning a deep convolutional network for image super-resolution. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 184–199, Cham. Springer International Publishing.

[Dong et al., 2016] Dong, C., Loy, C. C., and Tang, X. (2016). Accelerating the super-resolution convolutional neural network.

[Dosovitskiy and Brox, 2016] Dosovitskiy, A. and Brox, T. (2016). Generating images with perceptual similarity metrics based on deep networks.

[Dosselmann and Yang, 2005] Dosselmann, R. and Yang, X. (2005). Existing and emerging image quality metrics. volume 2005, pages 1906 – 1913.

[Duchon, 1979] Duchon, C. (1979). Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology - J APPL METEOROL*, 18:1016–1022.

[Fang et al., 2019] Fang, L., Monroe, F., Novak, S. W., Kirk, L., Schiavon, C. R., Yu, S. B., Zhang, T., Wu, M., Kastner, K., Kubota, Y., Zhang, Z., Pekkurnaz, G., Mendenhall, J., Harris, K., Howard, J., and Manor, U. (2019). Deep learning-based point-scanning super-resolution imaging. *bioRxiv*.

[Fookes et al., 2012] Fookes, C., Lin, F., Chandran, V., and Sridharan, S. (2012). Evaluation of image resolution and super-resolution on face recognition performance. *J. Visual Communication and Image Representation*, 23:75–93.

[Freedman and Fattal, 2011] Freedman, G. and Fattal, R. (2011). Image and video upscaling from local self-examples. *ACM Trans. Graph.*, 30(2).

[Freeman et al., 2002] Freeman, W. T., Jones, T. R., and Pasztor, E. C. (2002). Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65.

[Gao et al., 2020] Gao, H., Yuan, H., Wang, Z., and Ji, S. (2020). Pixel transposed convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(5):1218–1227.

[Gatys et al., 2015] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). Texture synthesis using convolutional neural networks.

[Gatys et al., 2016] Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423.

[Glasner et al., 2009] Glasner, D., Bagon, S., and Irani, M. (2009). Super-resolution from a single image. pages 349 – 356.

[Goodfellow et al., 2014] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

[Grant-Jacob et al., 2019] Grant-Jacob, J. A., Mackay, B. S., Baker, J. A. G., Xie, Y., Heath, D. J., Loxham, M., Eason, R. W., and Mills, B. (2019). A neural lens for super-resolution biological imaging. *Journal of Physics Communications*, 3(6):065004.

[H. R. Sheikh and Bovik, 2006] H. R. Sheikh, M. F. S. and Bovik, A. C. (2006). A statistical evaluation of recent full reference image quality assessment algorithms.

[Han et al., 2018] Han, W., Chang, S., Liu, D., Yu, M., Witbrock, M., and Huang, T. S. (2018). Image super-resolution via dual-state recurrent networks.

[Haris et al., 2018] Haris, M., Shakhnarovich, G., and Ukita, N. (2018). Deep back-projection networks for super-resolution. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1664–1673.

[Haris et al., 2018] Haris, M., Shakhnarovich, G., and Ukita, N. (2018). Task-driven super resolution: Object detection in low-resolution images.

[Haris et al., 2019] Haris, M., Shakhnarovich, G., and Ukita, N. (2019). Recurrent back-projection network for video super-resolution.

[Hong Chang et al., 2004] Hong Chang, Dit-Yan Yeung, and Yimin Xiong (2004). Super-resolution through neighbor embedding. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I.

[Horé and Ziou, 2010] Horé, A. and Ziou, D. (2010). Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369.

[Huang et al., 2017] Huang, Y., Shao, L., and Frangi, A. F. (2017). Simultaneous super-resolution and cross-modality synthesis of 3d medical images using weakly-supervised joint convolutional sparse coding.

[Hui et al., 2018] Hui, Z., Wang, X., and Gao, X. (2018). Fast and accurate single image super-resolution via information distillation network.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

[Irani and Peleg, 1991] Irani, M. and Peleg, S. (1991). Improving resolution by image registration. *CVGIP: Graphical Models and Image Processing*, 53:231–239.

[Isaac and Kulkarni, 2015] Isaac, J. S. and Kulkarni, R. (2015). Super resolution techniques for medical image processing. In *2015 International Conference on Technologies for Sustainable Development (ICTSD)*, pages 1–6.

[Jian Sun et al., 2008] Jian Sun, Zongben Xu, and Heung-Yeung Shum (2008). Image super-resolution using gradient profile prior. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[Jianchao Yang et al., 2008] Jianchao Yang, Wright, J., Huang, T., and Yi Ma (2008). Image super-resolution as sparse representation of raw image patches. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[Johnson et al., 2016] Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution.

[Kaiming He, 2015] Kaiming He, Xiangyu Zhang, S. R. J. S. (2015). Deep residual learning for image recognition.

[Kasthuri, 2015] Kasthuri, N., H. K. J. B. D. R. S. R. L. C. J. A. K.-B. S. [U+FFFD]. J. W. (2015). Deep residual learning for image recognition.

[Keys, 1981] Keys, R. (1981). Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160.

[Kim and Kwon, 2010] Kim, K. I. and Kwon, Y. (2010). Single-image super-resolution using sparse regression and natural image prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1127–1133.

[Kotevski and Mitrevski, 2010] Kotevski, Z. and Mitrevski, P. (2010). *Experimental Comparison of PSNR and SSIM Metrics for Video Quality Estimation*, pages 357–366.

[Kouame and Ploquin, 2009] Kouame, D. and Ploquin, M. (2009). Super-resolution in medical imaging : An illustrative approach through ultrasound. In *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 249–252.

[Kuwajima et al., 2013] Kuwajima, M., Mendenhall, J. M., Lindsey, L. F., and Harris, K. M. (2013). Automated transmission-mode scanning electron microscopy (tsem) for large volume analysis at nanoscale resolution. *PLoS ONE*, 8.

[Lai et al., 2017a] Lai, W.-S., Huang, J.-B., Ahuja, N., and Yang, M.-H. (2017a). Deep laplacian pyramid networks for fast and accurate super-resolution.

[Lai et al., 2017b] Lai, W.-S., Huang, J.-B., Ahuja, N., and Yang, M.-H. (2017b). Fast and accurate image super-resolution with deep laplacian pyramid networks.

[Ledig et al., 2016] Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., and Shi, W. (2016). Photo-realistic single image super-resolution using a generative adversarial network.

[Li et al., 2019] Li, Z., Yang, J., Liu, Z., Yang, X., Jeon, G., and Wu, W. (2019). Feedback network for image super-resolution.

[Lim et al., 2017] Lim, B., Son, S., Kim, H., Nah, S., and Lee, K. M. (2017). Enhanced deep residual networks for single image super-resolution.

[Liu et al., 2018] Liu, P., Zhang, H., Zhang, K., Lin, L., and Zuo, W. (2018). Multi-level wavelet-cnn for image restoration.

[Loshchilov and Hutter, 2016] Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts.

[Mao et al., 2016] Mao, X.-J., Shen, C., and Yang, Y.-B. (2016). Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections.

[Mittal et al., 2013] Mittal, A., Soundararajan, R., and Bovik, A. C. (2013). Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212.

[Moen et al., 2019] Moen, E., Bannon, D., Kudo, T., Graf, W., Covert, M., and Valen, D. (2019). Deep learning for cellular image analysis. *Nature Methods*, 16.

[Odena et al., 2016] Odena, A., Dumoulin, V., and Olah, C. (2016). Deconvolution and checkerboard artifacts. *Distill*, 1.

[Pambrun and Noumeir, 2015] Pambrun, J. and Noumeir, R. (2015). Limitations of the ssim quality metric in the context of diagnostic imaging. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 2960–2963.

[Rasti et al., 2016] Rasti, P., Uiboupin, T., Escalera, S., and Anbarjafari, G. (2016). Convolutional neural network super resolution for face recognition in surveillance monitoring. volume 9756, pages 175–184.

[Sajjadi et al., 2016] Sajjadi, M. S. M., Schölkopf, B., and Hirsch, M. (2016). Enhancenet: Single image super-resolution through automated texture synthesis.

[Schulter et al., 2015] Schulter, S., Leistner, C., and Bischof, H. (2015). Fast and accurate image upscaling with super-resolution forests. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3791–3799.

[Shi et al., 2016] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

[Smith and Topin, 2017] Smith, L. N. and Topin, N. (2017). Super-convergence: Very fast training of neural networks using large learning rates.

[Sønderby et al., 2016] Sønderby, C. K., Caballero, J., Theis, L., Shi, W., and Huszár, F. (2016). Amortised map inference for image super-resolution.

[Tai et al., 2017] Tai, Y., Yang, J., and Liu, X. (2017). Image super-resolution via deep recursive residual network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2790–2798.

[Tai et al., 2017] Tai, Y., Yang, J., Liu, X., and Xu, C. (2017). Memnet: A persistent memory network for image restoration.

[Timofte et al., 2015a] Timofte, R., De Smet, V., and Van Gool, L. (2015a). A+: Adjusted anchored neighborhood regression for fast super-resolution. volume 9006, pages 111–126.

[Timofte et al., 2015b] Timofte, R., Rothe, R., and Gool, L. V. (2015b). Seven ways to improve example-based single image super resolution.

[Tong et al., 2017] Tong, T., Li, G., Liu, X., and Gao, Q. (2017). Image super-resolution using dense skip connections. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4809–4817.

[Tosheva, 2020] Tosheva, K. L., Y. Y. P. P. M. C. S. . H. R. (2020). Between life and death: strategies to reduce phototoxicity in super-resolution microscopy.

[Wang, 2003] Wang, Z., S. E. P. A. B. (2003). Multiscale structural similarity for image quality assessment.

[Wang et al., 2018a] Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, C. C., Qiao, Y., and Tang, X. (2018a). Esrgan: Enhanced super-resolution generative adversarial networks.

[Wang et al., 2018b] Wang, Y., Perazzi, F., McWilliams, B., Sorkine-Hornung, A., Sorkine-Hornung, O., and Schroers, C. (2018b). A fully progressive approach to single-image super-resolution.

[Wang and Bovik, 2009] Wang, Z. and Bovik, A. C. (2009). Mean squared error: Love it or leave it? a new look at signal fidelity measures.

[Wang et al., 2020] Wang, Z., Chen, J., and Hoi, S. C. H. (2020). Deep learning for image super-resolution: A survey.

[Weigert et al., 2018] Weigert, M., Schmidt, U., Boothe, T., Müller, A., Dibrov, A., Jain, A., Wilhelm, B., Schmidt, D., Broaddus, C., Culley, S., Rocha-Martins, M., Segovia-Miranda, F., Norden, C., Henriques, R., Zerial, M., Solimena, M., Rink, J., Tomancak, P., Royer, L., and Myers, E. (2018). Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature Methods*, 15.

[Xiong et al., 2010] Xiong, Z., Sun, X., and Wu, F. (2010). Robust web image/video super-resolution. *IEEE Transactions on Image Processing*, 19(8):2017–2028.

[Yang et al., 2010] Yang, J., Wright, J., Huang, T. S., and Ma, Y. (2010). Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873.

[Z. Wang, 2002] Z. Wang, A. C. B. (2002). A universal image quality index.

[Z. Wang and Lu, 2002] Z. Wang, A. C. B. and Lu, L. (2002). Why is image quality assessment so difficult?

[Z. Wang and Simoncelli, 2004] Z. Wang, A. C. Bovik, H. R. S. and Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity.

[Zeiler and Fergus, 2013] Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks.

[Zeiler et al., 2010] Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2528–2535.

[Zhang et al., 2018a] Zhang, K., Zhang, Z., Cheng, C.-W., Hsu, W. H., Qiao, Y., Liu, W., and Zhang, T. (2018a). Super-identity convolutional neural network for face hallucination.

[Zhang et al., 2017] Zhang, K., Zuo, W., and Zhang, L. (2017). Learning a single convolutional super-resolution network for multiple degradations.

[Zhang et al., 2010] Zhang, L., Zhang, H., Shen, H., and Li, P. (2010). A super-resolution reconstruction algorithm for surveillance images. *Signal Process.*, 90:848–859.

[Zhang et al., 2011] Zhang, L., Zhang, L., Mou, X., and Zhang, D. (2011). Fsim: A feature similarity index for image quality assessment. *IEEE Transactions on Image Processing*, 20(8):2378–2386.

[Zhang et al., 2018b] Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., and Fu, Y. (2018b). Image super-resolution using very deep residual channel attention networks.

[Zhang et al., 2018c] Zhang, Y., Tian, Y., Kong, Y., Zhong, B., and Fu, Y. (2018c). Residual dense network for image super-resolution.

[Zhao et al., 2015] Zhao, H., Gallo, O., Frosio, I., and Kautz, J. (2015). Loss functions for neural networks for image processing.