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

Métodos de aprendizaje profundo para la súper-resolución y segmentación semántica de imágenes

Autor

Aitor González Marfil

2021

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

Métodos de aprendizaje profundo para la súper-resolución y segmentación semántica de imágenes

Autor

Aitor González Marfil

Directores

Ignacio Arganda Carreras y Gorka Azkune Galparsoro

Resumen

En este proyecto hemos explorado la complementariedad de dos importantes tareas del campo de la visión artificial, como son la súper-resolución y la segmentación semántica. Nuestra hipótesis de partida es que ambas tareas requieren de habilidades parecidas, por lo que las arquitecturas modernas propuestas para cada una de ellas, deberían ser buenas soluciones para la otra tarea. Para validar esta hipótesis, hemos seleccionado una arquitectura típica de segmentación semántica y hemos medido su desempeño en la tarea de súper-resolución. También hemos repetido el mismo procedimiento al revés. Los resultados obtenidos nos muestran que, en efecto, una red diseñada para segmentación semántica obtiene buenos resultados en súper-resolución, y viceversa. Por ello, en este proyecto proponemos una única red multi-tarea que es capaz de realizar las dos tareas simultáneamente, alcanzando resultados comparables a las arquitecturas diseñadas específicamente para cada una de las tareas.

Índice general

Índice de figuras

Índice de tablas

1. CAPÍTULO

Introducción

La digitalización de la sociedad ha traído como consecuencia el aumento exponencial de la cantidad de imágenes y videos generados, ya que entre otras cosas, casi todas las personas contamos con una cámara en nuestros bolsillos. Esta gran cantidad de datos ha estado en el origen del desarrollo acelerado que ha experimentado el campo de la visión artificial en los últimos años. En este proyecto, nos centramos en dos tareas que generan gran interés en la visión artificial: la segmentación semántica (ver ejemplo en la Figura 1.2) y la súper-resolución (ver ejemplo en la Figura 1.1). Mientras en la segmentación semántica lo que se busca es clasificar los distintos elementos presentes en la imagen, en la súper-resolución lo que se busca es aumentar la resolución de las imágenes.

A lo largo de los años se ha estudiado una amplia variedad de algoritmos para ambas tareas [van Ouwerkerk, 2006, Thoma, 2016], pero las propuestas basadas en métodos de aprendizaje profundo están obteniendo los mejores resultados [Sultana et al., 2020, Wang et al., 2020b]. Aun así, incluso los mejores métodos ofrecen un amplio margen de mejora, lo que motiva una mayor investigación en estos campos.

Nuestra hipótesis es que las tareas de segmentación semántica y súper-resolución requieren de habilidades muy parecidas, por lo que se pueden resolver satisfactoriamente con una misma red. Hay estudios que aportan evidencias interesantes en este sentido. Estudios en los que hacen uso de esta similitud para obtener un mejor desempeño en la tarea de segmentación semántica [Wang et al., 2020a]. O hacen uso de una tarea similar, como la supresión de ruido, para entrenar la red de segmentación semántica con un subconjunto reducido de imágenes segmentadas [Tim-Oliver et al., 2020].

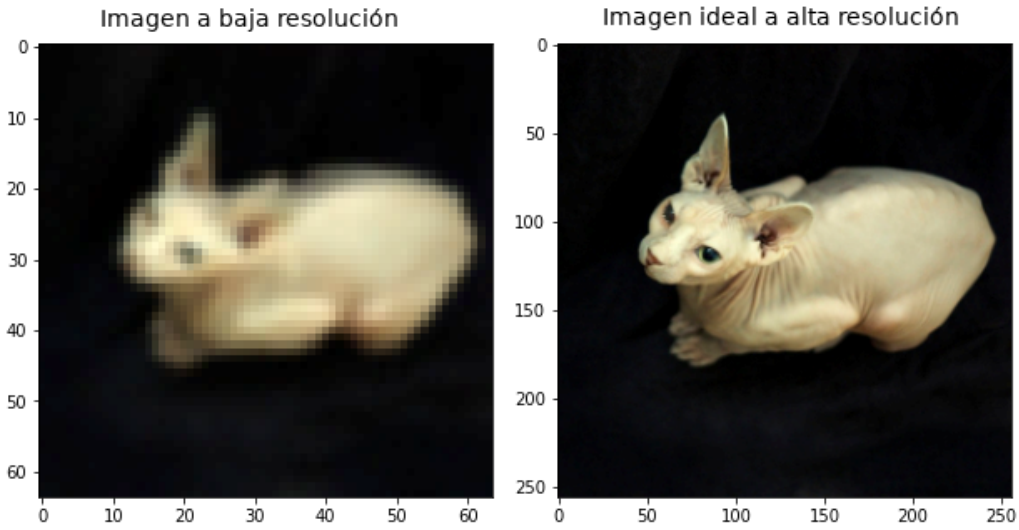

Figura 1.1: Ejemplo de súper-resolución. De izquierda a derecha: imagen a baja resolución (64×64 píxeles) y su correspondiente versión al cuádruple de resolución (256×256 píxeles).

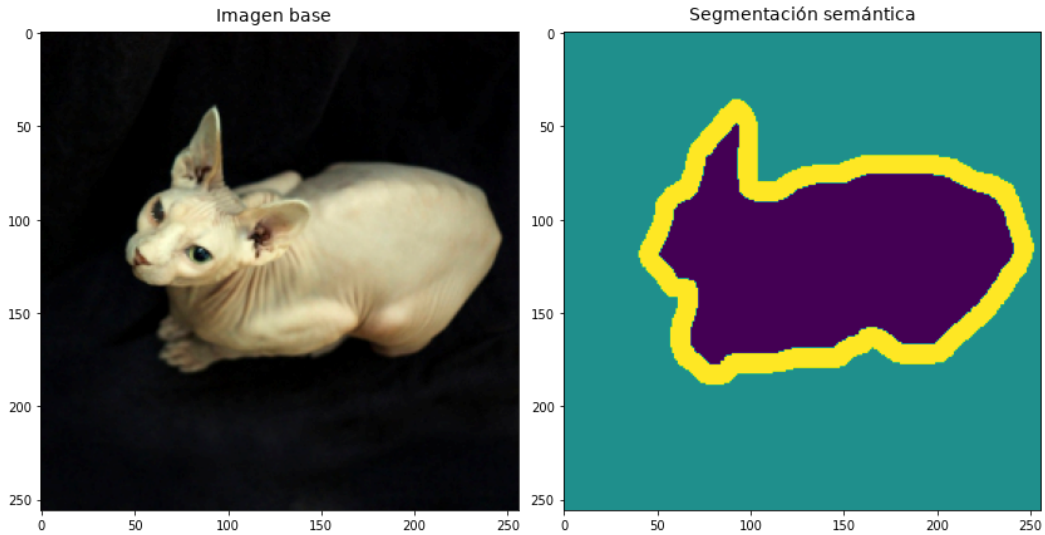

Figura 1.2: Ejemplo de segmentación semántica. De izquierda a derecha: imagen base y su correspondiente segmentación semántica. En la máscara de segmentación cada clase se ve reflejada con un color distinto. Siendo azul marino el animal (perro o gato), amarillo el contorno del animal y turquesa oscuro cualquier otro elemento.

En este proyecto, principalmente se analizará el desempeño de dos arquitecturas. Cada una especializada en una tarea distinta. Y se explorará el desempeño de ambas arquitecturas a la hora de hacer la tarea para la que no están especializadas. En concreto, se analizará la U-Net [Ronneberger et al., 2015], y una versión derivada con conexiones residuales [He et al., 2016], a la que llamaremos ResUNet [Zhang et al., 2018b]. Ambas se analizarán para la tarea de súper-resolución, pese a que su especialidad original es la segmentación. Y por otra parte se analizará la *Residual Channel Attention Networks* (RCAN) [Zhang et al., 2018a] para la tarea de segmentación semántica, pese a que su especialidad es súper resolver imágenes. De forma adicional, se analizará el desempeño de las arquitecturas para realizar ambas tareas simultáneamente. Y se experimentará con combinaciones de estas redes, para ambas tareas, por separado y multitarea.

Los objetivos concretos planteados para el proyecto son los siguientes:

1. Familiarizarse con las herramientas actuales de desarrollo y evaluación de métodos de aprendizaje profundo.
2. Analizar el estado del arte del aumento de resolución automático de imágenes y segmentación semántica de imágenes.
3. Estudiar el rendimiento de redes especializadas en súper-resolución para segmentación semántica.
4. Estudiar el rendimiento de redes especializadas en segmentación semántica para súper-resolución.
5. Estudiar el rendimiento de redes especializadas en distintas tareas para la realización de ambas tareas simultáneamente.
6. Evaluar los modelos implementados y entrenados usando datos reales.

2. CAPÍTULO

Estado del arte

En este capítulo se introducirán las tareas sobre las que se va a trabajar, se explicará en que consisten, se darán ejemplos de uso y se explicarán las métricas habituales para evaluarlas. Posteriormente se introducirán algunas de las arquitecturas empleadas a lo largo de la historia, junto a sus funciones de pérdida y demás módulos. Y finalmente se presentarán las arquitecturas pertenecientes al estado del arte para estas tareas, con las que se ha trabajado a lo largo del proyecto.

2.1. Segmentación semántica

La tarea de segmentación semántica consiste en dada una imagen, etiquetar los píxeles de la imagen en función de a cuál de las categorías definidas pertenece. En caso de haber múltiples elementos de la misma clase en la imagen, todas ellas se clasificarán con la misma clase. Por ejemplo, en una foto con varias personas, si la tarea es segmentar personas, se clasificarán a todas con la misma clase, no se hace una distinción entre personas.

La tarea encargada de segmentar cada persona de forma individual, tiene cierta complejidad añadida a la tarea de la segmentación. A esta tarea se le denomina segmentación de instancias. En este proyecto se trabajará con la tarea de segmentación semántica.

En la Figura 2.1 podemos ver un ejemplo donde se realizan ambas tareas. Por un lado, en la segmentación semántica se han clasificado todas las personas con la misma etiqueta, representada mediante el mismo color. En cambio, en la segmentación de instancias

Figura 2.1: Segmentación semántica y segmentación de instancias. De izquierda a derecha: segmentación semántica aplicada a una imagen, y segmentación de instancias aplicada a la misma imagen. Fuente: [Arnab et al., 2018].

se hace una distinción entre personas mediante distintos colores. En este ejemplo de forma adicional se segmentan también otros elementos, como pueda ser la mesa. En ambas imágenes se puede ver que la mesa está clasificada con una etiqueta distinta a las demás, representada con un color distinto.

La segmentación semántica es una tarea con una amplia variedad de aplicaciones. Esto se debe a que hay muchas aplicaciones que requieren una cierta comprensión del contenido de la imagen. Ejemplos de ello son: la detección y segmentación de condiciones médicas en células y tejidos [Ronneberger et al., 2015, Wu et al., 2019, Isensee et al., 2020] (ver ejemplo de la Figura 2.2), la navegación en coches autónomos [Alonso et al., 2020, Sagar and Soundrapandiyan, 2020] (ver ejemplo de la Figura 2.3) o el desarrollo de robots que puedan moverse e interactuar con el entorno [Milioto et al., 2018] (ver ejemplo de la Figura 2.4).

Existen varios métodos para realizar esta tarea [Thoma, 2016], como por ejemplo mediante *clustering*. Pero la mayoría de ellos son muy circunstanciales. No fue hasta la llegada de las redes convolucionales [Krizhevsky et al., 2012], que hubo un antes y un después en el área de la visión por computador. Área a la que pertenece esta tarea. Como se indica en su nombre, estas redes deben su nombre a las convoluciones. Ya que la forma que tienen de trabajar, es aplicando convoluciones bidimensionales de forma reiterada. El papel de la red en este proceso, es aprender los filtros necesarios en cada convolución, para poder obtener la salida deseada.

Gracias a las investigaciones realizadas en busca de una mayor comprensión del funcio-

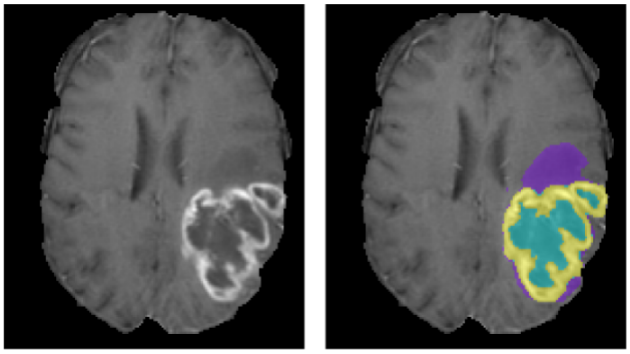

Figura 2.2: Aplicación de la segmentación semántica en medicina. De izquierda a derecha: imagen tomográfica de un cerebro y su correspondiente segmentación semántica, clasificando los tumores presentes en este. Fuente: [Isensee et al., 2020].

Figura 2.3: Aplicación de la segmentación semántica en vehículos autónomos. Ejemplo de segmentación semántica para imágenes obtenidas desde un vehículo. En la segmentación semántica se clasifican los transeúntes, acera, carretera, señales de tráfico, y demás elementos clave de la escena. Fuente: cityscapes-dataset.com.

Figura 2.4: Aplicación de la segmentación semántica en robots agricultores. A la izquierda se encuentra el robot agricultor. En la zona superior derecha se encuentra la imagen obtenida por el robot, y debajo, la segmentación semántica extraída, clasificando diversas plantas. Fuente: [Milioto et al., 2018].

namiento interno de estas redes, se han desarrollado métodos como la maximización de la activación para un filtro dado [Erhan et al., 2009]. Haciendo uso de estos métodos se ha podido entender mejor el funcionamiento interno de estas redes [Qin et al., 2018]. Se ha visto que los primeros filtros aplicados a la imagen de entrada aprenden patrones más generales o superficiales de la imagen, como puedan ser líneas rectas o esquinas. Y a medida que se profundiza en la red, las señales se activan más ante estímulos de cada vez mayor carga semántica. A modo de ejemplo, ante una imagen de una cara, al principio los filtros se encargarían de obtener las distintas líneas que forman la imagen. Posteriormente detectaría que una cierta composición de líneas dan como resultado un ojo o una nariz. Y finalmente reconocería que una composición de ojos, cejas, nariz y demás elementos, forman una cara.

Podemos ver esto reflejado en la Figura 2.5, donde se han escogido dos neuronas aleatorias en distintas capas (*Convolution Layer*, CL). Concretamente, desde la primera hasta la quinta capa. Y se han anulado el resto de neuronas. Las imágenes en la segunda columna de cada capa, son el resultado de este proceso. En la primera columna se encuentran las secciones de la imagen que más han activado cada neurona. Lo podemos interpretar como la sección de la imagen en la que la neurona se ha fijado, o a la que está mirando. Este ejemplo se encuentra más extensamente explicado en [Qin et al., 2018].

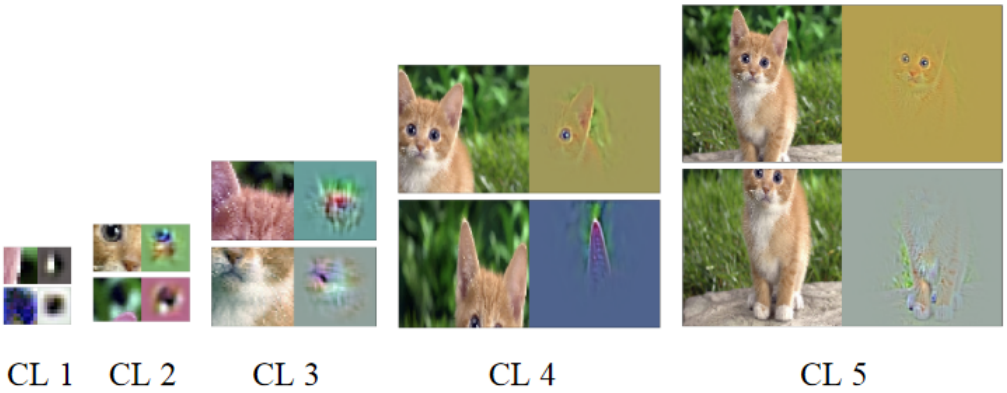

Figura 2.5: Visualización de características extraídas por una red convolucional. De izquierda a derecha podemos ver varios grupos de imágenes, extraídas de capas cada vez más profundas de la red; siendo el primer bloque, la primera capa y el último bloque, la quinta capa. Cada uno de estos bloques cuenta con 4 imágenes. Cada fila refleja de izquierda a derecha: la zona de la imagen en la que se está fijando una neurona y las características extraídas por dicha neurona. De esta manera, cada bloque se compone con imágenes obtenidas a partir de dos neuronas distintas de una cierta profundidad. Fuente: [Qin et al., 2018]

2.1.1. Métrica de evaluación para tareas de segmentación

Para evaluar qué tan buena es la segmentación realizada, se hará uso de la métrica *mean Intersection over Union* (mIoU). También es conocida como índice de Jaccard. La operación realizada por esta métrica se reduce en dividir el número de píxeles bien clasificados, que se solapan entre la segmentación predicha y su solución, entre el número de píxeles que suman en total la segmentación predicha y su solución.

La operación se puede ver reflejada en la Figura 2.6 y vendría definida tal y como podemos ver en la Ecuación 2.1. TP_c es el número de píxeles correctamente clasificados para la clase c. FP_c es el número de píxeles incorrectamente clasificados como pertenecientes a la clase c. FN_c es el número de píxeles incorrectamente clasificados como no pertenecientes a la clase c.

$$IoU(c) = \frac{TP_c}{TP_c + FP_c + FN_c} \tag{2.1}$$

Esta métrica es capaz de recompensar la tasa de acierto alta mientras penaliza los errores. La métrica nos dará como resultado un valor dentro del rango $[0,1]$. En el peor de los casos no habrá ningún píxel correctamente clasificado ($TP = 0$) con lo que se devolverá un 0. Y en el mejor de los casos todos los píxeles estarían correctamente clasificados con

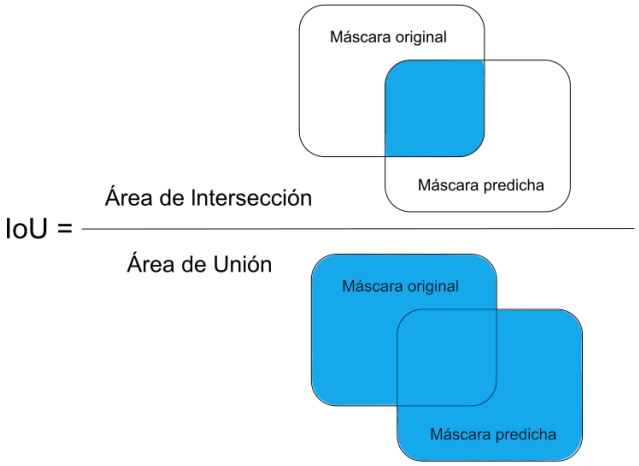

Figura 2.6: Métrica de Intersección sobre Unión. En la figura se ilustra la operación realizada en la métrica IoU para una clase.

lo que la operación devolverá un 1. Puesto que contamos con múltiples clases, se realizará este cálculo para cada una de las clases, y trabajaremos con su media. Se puede ver esta operación en la Ecuación 2.2, donde K indica el número de clases.

$$mIoU = \frac{\sum_{x=1}^{K} mIoU(x)}{K} \tag{2.2}$$

El rango de valores posibles para mIoU e IoU sería la misma, con la diferencia de que IoU nos indica la calidad de la segmentación realizada para una clase, y mIoU nos indicaría la calidad promedio de la segmentación, para todas sus clases.

2.1.2. Métodos de aprendizaje profundo para segmentación

Nos centraremos en aquellos métodos supervisados que cuentan tanto con la imagen como con su correspondiente máscara.

Dentro de la tarea de segmentación semántica, hay una amplia variedad de propuestas basadas en métodos de aprendizaje profundo. Aun así, se ha podido ver una evolución dentro de algunas de las arquitecturas más influyentes.

1. **Red totalmente convolucional**

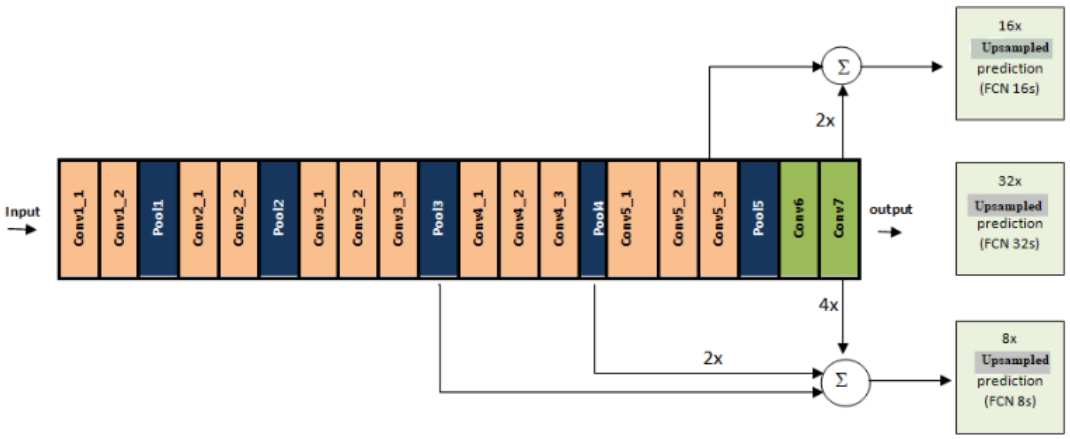

Figura 2.7: Arquitectura totalmente convolucional. Se muestran las arquitecturas de las redes FCN32s, FCN16s y FCN8s. Cada uno de los bloques verdes que se encuentran a la derecha, corresponde a la salida de una de estas redes. Fuente: [Sultana et al., 2020]

Esta arquitectura, traducida al inglés como *Fully Convolutional Network* (FCN), es una de las más simples. Consiste en una serie de convoluciones y módulos de reducción de escala, como pueda ser *max-pooling*. Está inspirada en redes previamente vistas en otras tareas, como clasificación. Pero a diferencia de otras tareas donde finalmente se devuelve un valor, en esta se devuelve una imagen. La imagen final cuenta con un tamaño inferior al de la imagen de entrada por lo que se suele aplicar algún método de aumento de escala, como el reescalado bilineal. Se puede ver esta arquitectura reflejada en la Figura 2.7.

2. **Deconvnet**

   Basada en el enfoque de reducción y aumento de escala, la arquitectura cuenta con dos partes (ver Figura 2.8): una primera mitad compuesta por convoluciones y módulos de reducción de escala, a la que se le llama codificador o *encoder* en inglés; y una segunda mitad simétrica compuesta por convoluciones y módulos de aumento de escala, como puedan ser las convoluciones transpuestas, también llamadas deconvoluciones. Esta segunda mitad deconvolucional también es conocida como decodificador o *decoder* en inglés.

   La imagen se comprime en la primera mitad, donde se extraen las características de la imagen. Y se reconstruye en la segunda mitad, generando el mapa de segmentación. Este tipo de estructuras, compuestas con un codificador seguido de un decodificador, son tradicionalmente conocidas como *autoencoders*.

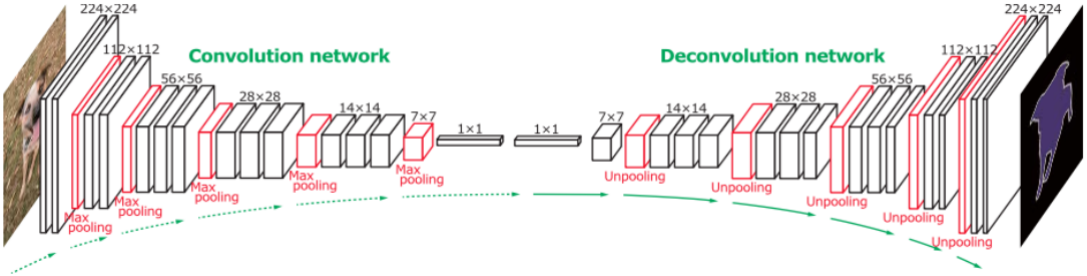

Figura 2.8: Arquitectura Deconvnet. La imagen ilustra la arquitectura deconvnet. Desde el extremo izquierdo, por el que entra la imagen, hasta el extremo derecho, por el que sale la segmentación semántica. El tamaño de cada bloque representa el tamaño de la imagen en esa capa. Encima de cada grupo de bloques se especifica el tamaño de la imagen en esos bloques. Y el grosor de cada bloque representa el número de canales en esa capa. Los bloques negros representan el resultado de aplicar una convolución. Los bloques rojos representan el resultado de aplicar una operación de aumento o reducción de escala, esta se especifica debajo de cada bloque. También se señaliza en verde, encima de cada una de las mitades, que tipo de estructura son: convolucional o deconvolucional. Fuente: [Sultana et al., 2020]

Una de las funciones de pérdida comúnmente utilizada en redes de aprendizaje profundo para esta tarea es la función de entropía cruzada categórica dispersa. En inglés es conocida como *Sparse Categorical Cross Entropy* (SCCE).

Esta función de pérdida hereda la idea tras la entropía cruzada, y se define como:

$$L_{CE}(p,q) = -\sum_{x}^{n} p(x)\log_2 q(x) \tag{2.3}$$

Para n clases, donde $p(x)$ será 1 o 0 en función de si pertenece a la clase x o no. Y $q(x)$ es la probabilidad predicha de pertenecer a la clase x.

La diferencia entre la entropía cruzada categórica y la entropía cruzada categórica dispersa, es que las clases verdaderas no se encuentran codificadas en distintos canales (un canal para cada clase). En su defecto se encuentran codificadas todas en un único canal. A modo de ejemplo, en lugar de tener para 3 clases, los siguientes tres vectores: $[1,0,0]$, $[0,1,0]$ y $[0,0,1]$, se contaría con un único vector con la forma: $[1,2,3]$. Nótese que esto sucede únicamente con los mapas de segmentación verdaderos, los predichos si se encuentran en distintos canales.

2.1.3. Arquitectura Res/U-Net

Dentro de la tarea de segmentación semántica, una de las redes que mejor desempeño ha obtenido estos últimos años es la arquitectura U-Net [Ronneberger et al., 2015]. Debe su nombre a la estructura de la red en forma de "U".

La arquitectura está formada por una primera mitad que reduce el tamaño de la imagen, pero aumenta el número de canales. Y una segunda mitad simétrica, que aumenta el tamaño de la imagen y reduce el número de canales. Está parte se asemeja a la arquitectura Deconvnet.

Comunicando las dos mitades de la red se añaden varias conexiones en distintas profundidades. Pasando las imágenes resultantes de la primera mitad de la red, a la segunda mitad de la red. Esto se realiza en función de la profundidad, de manera que el tamaño de la imagen concuerde. En el ejemplo de la Figura 2.9 la red posee 4 niveles de profundidad. Tal y como podemos observar en la Figura 2.9, el resultado es una red cuya forma se podría ver como una U.

Una de las variantes de la red U-Net que más ha destacado obteniendo también un muy buen rendimiento ha sido la red ResUNet. Es una arquitectura compuesta a base de bloques residuales, como los que podemos ver en la Figura 2.10. La primera mitad está compuesta por un bloque residual en cada profundidad, y entre bloques, módulos de reducción de escala. La segunda mitad es simétrica, con la diferencia de que entre cada bloque en lugar de reducir la escala de la imagen, se aumenta. Y por último al igual que en la U-Net se añaden varias conexiones que comunican las dos mitades de la red, en función de la profundidad.

Tal y como se puede apreciar en la Figura 2.9, las conexiones residuales o comunicaciones entre ambas mitades no siempre están a la misma distancia. Algunas están más lejos que otras. Y esto es importante, pues ayuda no solo a converger más rápido, sino también de forma más estable a la red [Drozdzal et al., 2016]. Para más inri, posteriormente se ha añadido una última conexión, no presente en la figura, que parte desde la primera convolución y llega a la última convolución de la red. Esta conexión es conocida como *long skip connection*.

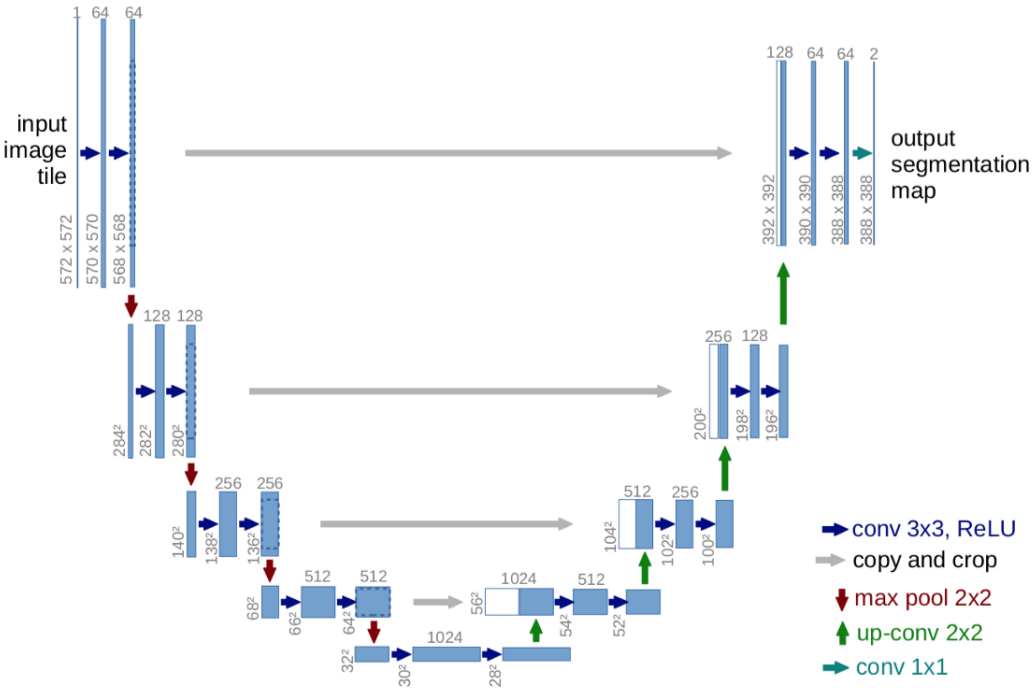

Figura 2.9: Arquitectura U-Net. La imagen ilustra la arquitectura U-Net. Desde el extremo izquierdo, por el que entra la imagen, hasta el extremo derecho, por el que sale la segmentación semántica. La altura de cada bloque representa el tamaño de la imagen en esa capa. Al costado izquierdo de cada bloque se especifica el tamaño de la imagen en ese bloque. La anchura de cada bloque representa el número de canales en esa capa. Encima de cada bloque se especifica el número de canales en ese bloque. Las flechas azules representan una convolución con un filtro de tamaño 3×3, seguido de la aplicación de la función de activación ReLU. Las flechas rojas representan una operación de reducción de escala. Las flechas verdes representan una operación de aumento de escala. La flecha turquesa oscuro final representa una convolución con un filtro de tamaño 1×1. Por último las flechas grises representan las *skip connections*. El resultado de copiar el bloque señalizado con una línea discontinua azul, en la base de cada flecha gris, e inyectarlo en otro punto, se señaliza con un bloque blanco, junto al bloque azul que señaliza la flecha. Fuente: [Ronneberger et al., 2015]

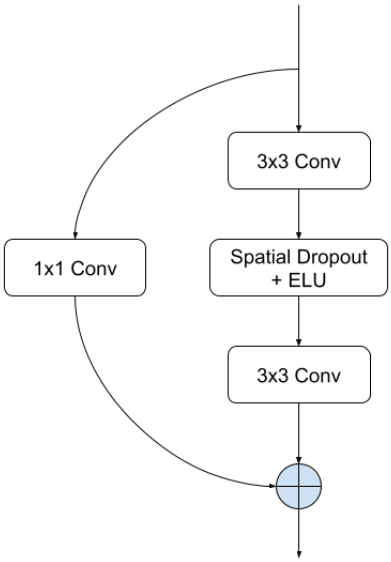

Figura 2.10: Bloque Residual. Este es el bloque residual que hemos usado. Cuenta con una primera bifurcación donde se encuentra la conexión residual con una convolución con un tamaño de filtro de 1×1 que genera la señal que se inyectara al final del bloque residual. El otro camino de la bifurcación se compone de dos convoluciones con un tamaño de filtro de 3×3, y entre ellas una combinación de la función de activación ELU y la técnica de regularización *Spatial Dropout*.

2.2. Súper resolución de una sola imagen

La tarea de súper resolver una sola imagen consiste en, dada una imagen a una cierta resolución, estimar la imagen con una resolución superior (ver Figura 1.1). Es una tarea de especial interés en áreas donde la obtención de imágenes de alta resolución supone un incremento en el coste. Haciendo la obtención de imágenes más cara o más lenta; pudiendo llegar a hacer que la imagen sea inviable de obtener. A modo de ejemplo, esto sucede tanto en microscopía electrónica como en telescopios especializados. También puede darse el caso de que directamente la imagen no sea reproducible. Como puedan ser las imágenes históricas. Imágenes que no se pueden repetir, y poseen un gran valor por ser únicas. Como pueda ser la imagen del congreso Solvay de 1927, donde se reunieron varios grandes nombres de la historia de la ciencia, como Erwin Schrödinger, Max Planck, Niels Bohr, Marie Curie o Albert Einstein.

Los métodos numéricos tradicionalmente utilizados para escalar una imagen (interpolación bilineal, por vecinos próximos, etc.), únicamente hacen uso de los valores adyacentes de cada píxel. Dando como resultado una imagen borrosa, con artefactos o dientes de sierra. Además de que los métodos no se adaptan bien al contenido de la imagen.

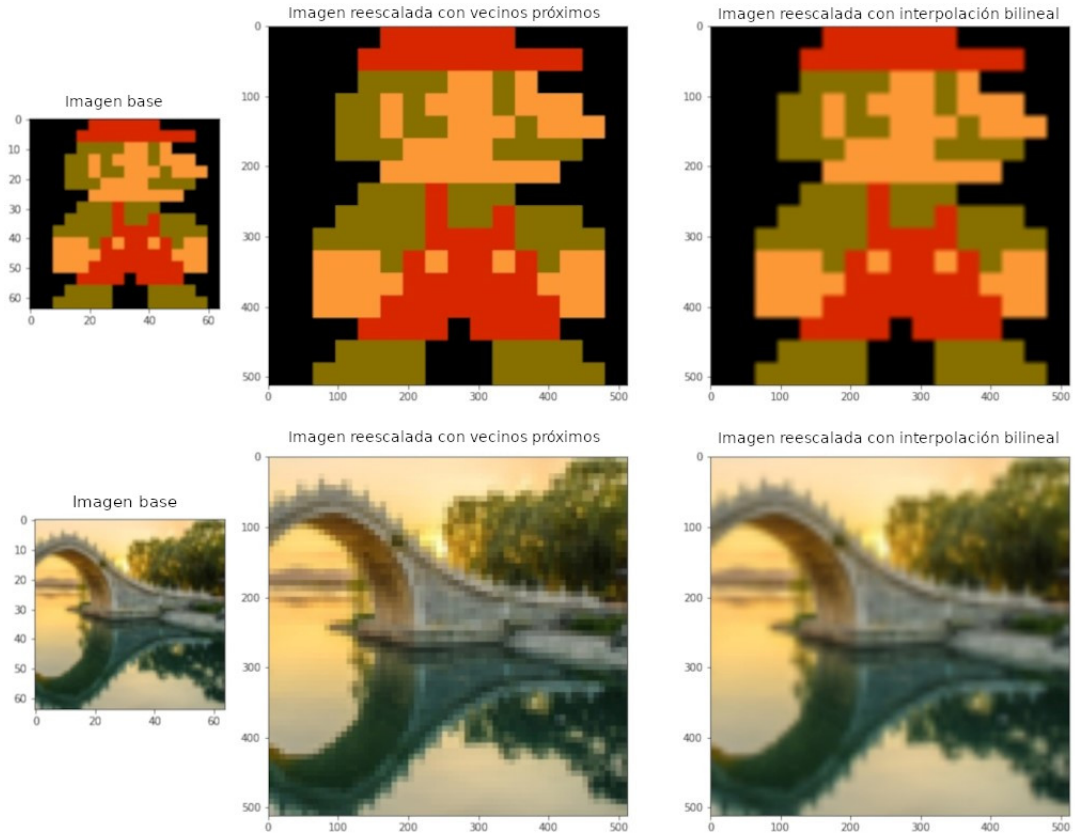

Figura 2.11: Ejemplo de distintos métodos de escalado. Por columnas, de izquierda a derecha: la imagen a baja resolución, y sus respectivas imágenes reescaladas con un aumento de factor 8, primeramente con el método de vecinos próximos y después con el método de interpolación lineal. La primera fila corresponde a un ejemplo con una imagen *pixel-art* de Mario, y la segunda fila corresponde a una imagen de la puesta de sol sobre un puente. Fuente: webstockreview.net (Mario) y wallup.net (paisaje), ambas empleadas para generar los ejemplos.

Por ejemplo, para escalar una fotografía de un paisaje, el escalado bilineal obtiene un resultado considerable. En cambio, aplicar este mismo algoritmo para una imagen *pixel-art*, únicamente difuminaría la imagen. Para las imágenes *pixel-art* sería más conveniente utilizar el algoritmo de vecinos próximos; ya que mantendría la relación de píxeles, dando una imagen mucho más nítida. Pero por otro lado, este mismo algoritmo aplicado a la imagen del paisaje, únicamente nos daría como resultado una imagen con artefactos y dientes de sierra muy marcados. Se puede ver este ejemplo reflejado en la Figura 2.11, con una imagen *pixel-art* de Mario, y una imagen de un paisaje con un puente.

En cualquier caso, ningún método es capaz de desempeñar su papel con una amplia variedad de imágenes, obteniendo como resultado imágenes foto-realistas. Y los mejores resultados obtenidos siguen dejando un amplio margen de mejora. Es este margen lo que

ha generado una amplia investigación en el área.

De las propuestas presentadas, aquellas que mejores resultados está obteniendo en esta tarea son las que hacen uso de redes de aprendizaje profundo [Wang et al., 2020b]. Las redes de aprendizaje profundo permiten reconstruir la imagen, haciendo uso de los patrones aprendidos en las imágenes otorgadas durante la etapa de entrenamiento. Esto permite hacer una reconstrucción de la imagen basándose no solo en los píxeles de la imagen, sino además en la información adquirida del resto de imágenes asimiladas por la red. Por ejemplo, para súper resolver la cara de un gato, se ayudará del resto de gatos previamente vistos por la red. Este enfoque se asemeja un poco más a la forma de trabajar de un humano. Esta forma de trabajar nos permite eliminar el ruido inherente a la obtención de la imagen, eliminar los artefactos presentes en la imagen, o reconstruir zonas dañadas. Dando como resultado imágenes de muy buena calidad.

2.2.1. Métricas de evaluación para súper resolución

La evaluación cuantitativa de qué tan buena es la imagen generada, es un tema difícil y ampliamente debatido por la comunidad [Wang et al., 2004]. Pese a que sigue siendo un tema abierto y en constante investigación, existe un conjunto de métricas que mantienen una correlación considerable con los valores obtenidos de participantes humanos en diversas pruebas. Este conjunto se compone de las siguientes métricas:

- **Proporción máxima de señal a ruido**

  En inglés se traduciría como *Peak Signal to Noise Ratio* (PSNR). Es una expresión que relaciona el máximo valor posible de una señal, y la potencia del ruido en la señal. Debido a que las señales cuentan con un amplio rango dinámico, generalmente se expresa en una escala logarítmica. En decibelios más concretamente.

  Para una definición más simple, primeramente se define el error cuadrático medio como en la Ecuación 2.4. También conocido como MSE por sus siglas en inglés, *Mean Squared Error*, donde dadas dos imágenes de tamaño $m \times n$, la original x y la predicha y, se mide la diferencia entre ambas imágenes. Que sería el ruido no deseado de la imagen.

$$MSE(x,y) = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [x(i,j) - y(i,j)]^2 \tag{2.4}$$

Una vez contamos con el MSE, podemos definir el PSNR en decibelios tal y como se indica en la Ecuación 2.6, donde MAX_f indica el valor máximo posible de la imagen. Se puede calcular mediante la Ecuación 2.5, donde B indica el número de bits por píxel. En nuestro caso con 8 bits por píxel, será de 255.

$$MAX_f = 2^B - 1 \tag{2.5}$$

$$PSNR(x,y) = 20\log_{10}\left(\frac{MAX_f}{\sqrt{MSE(x,y)}}\right) \tag{2.6}$$

- **Similitud estructural**

  Conocido en inglés como *Structural Similarity Index Measure* (SSIM). Es una métrica que sirve para medir la similitud estructural de dos imágenes (x e y) [Wang et al., 2004].

  A diferencia de métodos de error absoluto cómo MSE o PSNR, SSIM es una métrica basada en la percepción. Podemos dividir la percepción considerada por esta métrica en 3 importantes fenómenos: enmascaramiento de luminancia, enmascaramiento de contraste y estructura. La estructura sigue la idea de que los píxeles tienen una mayor interdependencia cuanto más cerca se encuentren. El enmascaramiento de luminancia por otro lado, indica que las distorsiones de la imagen suelen ser menos notorias en regiones brillantes. Y por último, el enmascaramiento de contraste mantiene la idea de que las distorsiones son menos perceptibles en zonas de gran actividad como puedan ser las texturas.

  Considerando que:

  - μ_x es el promedio de x.
  - μ_y es el promedio de y.
  - σ_x^2 es la varianza de x.
  - σ_y^2 es la varianza de y.
  - σ_{xy} es la covarianza de x e y.
  - MAX_f es el rango dinámico de los píxeles. Habitualmente definido como en la Ecuación 2.5.
  - $k_1 << 1$ y $k_2 << 1$ son constantes cuyos valores habitualmente se establecen en $k_1 = 0{,}01$ y $k_2 = 0{,}03$.

- c_1, c_2 y c_3 son variables para estabilizar las divisiones con un denominador muy pequeño, donde:
  - $c_1 = (k_1 MAX_f)^2$.
  - $c_2 = (k_2 MAX_f)^2$.
  - $c_3 = \frac{c_2}{2}$.

Tal y como se ha comentado previamente, podemos dividir la métrica en tres apartados: luminancia 2.7, estructura 2.8 y contraste 2.9.

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{2.7}$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{2.8}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{2.9}$$

SSIM es la combinación ponderada por α, β y γ de los elementos mencionados previamente. Tal y como podemos ver en la Ecuación 2.10.

$$SSIM(x,y) = [l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma] \tag{2.10}$$

Si establecemos todos los pesos (α, β y γ) a 1 podemos reducir la ecuación, hasta obtener la Ecuación 2.11

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.11}$$

Los valores de SSIM están dentro del rango $[0,1]$. El valor 1 únicamente se obtendrá cuando se evalúen 2 imágenes idénticas, lo que señaliza una similitud estructural perfecta. Por otra parte, 0 indica que no hay similitud alguna.

2.2.2. Métodos de aprendizaje profundo para súper resolución

En los métodos supervisados se cuenta tanto con la imagen a baja resolución, como su correspondiente solución, la imagen a alta resolución. Nosotros nos centraremos en este

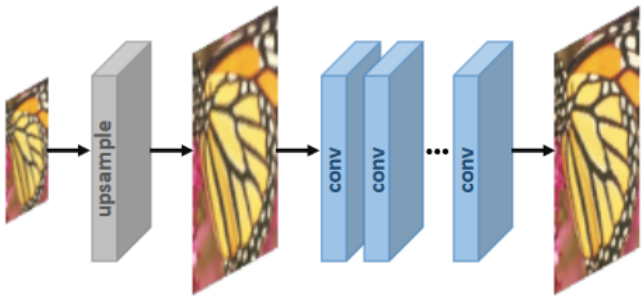

Figura 2.12: Método de súper-resolución basado en aumentar primero de escala. El módulo de aumento de escala se refleja en la figura con un bloque gris. A continuación le sigue una retahíla de convoluciones representadas con diversos bloques azules, dando como resultado la imagen súper-resuelta final. Fuente: [Wang et al., 2020b].

método de trabajo. Una forma fácil de adquirir las imágenes a baja resolución sería añadiendo ruido a las imágenes a alta resolución y reduciendo la resolución mediante alguna técnica como reescalado bilineal.

Existen varias propuestas de arquitecturas para redes de aprendizaje profundo, con una amplia variedad de diferencias. Pero en esencia son una combinación de componentes, métodos de aumento de escala, diseño de la arquitectura y estrategias de aprendizaje.

Los componentes principales son: convoluciones, módulos de aumento de escala y módulos de reducción de escala. Es habitual ver casos en los que la imagen resultante de una convolución no solo pasa al siguiente bloque, sino que además también se inyecta en componentes posteriores [He et al., 2016]. Este tipo de conexiones son conocidas como *skip connections*.

Las distribuciones de componentes más habituales son las siguientes:

- ***Pre-Upsampling***

  Este método consiste en empezar aumentando la escala de la imagen. Una vez tenemos una imagen súper-resuelta de forma burda, la tarea de la red es aprender a pulir los detalles de la imagen, hasta obtener una versión refinada (ver Figura 2.12).

- ***Post-Upsampling***

  En este método, primero se pasa la imagen a baja resolución por una serie de convoluciones y finalmente se aumenta su escala (ver Figura 2.13). De esta manera la extracción de características [Qin et al., 2018] realizada por las convoluciones, se realiza sobre un espacio de menor dimensión, ya que las imágenes tienen un

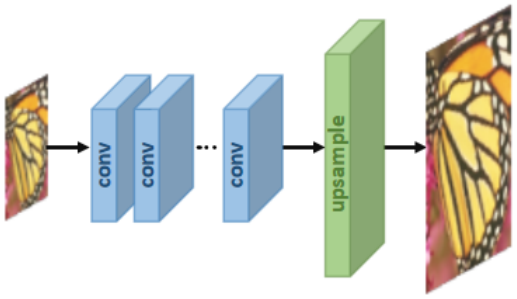

Figura 2.13: Método de súper-resolución basado en aumentar de escala al final. La imagen pasa primero por una retahíla de convoluciones representadas con diversos bloques azules. Y finalmente pasa por el módulo de aumento de escala representado con un bloque verde. Fuente: [Wang et al., 2020b].

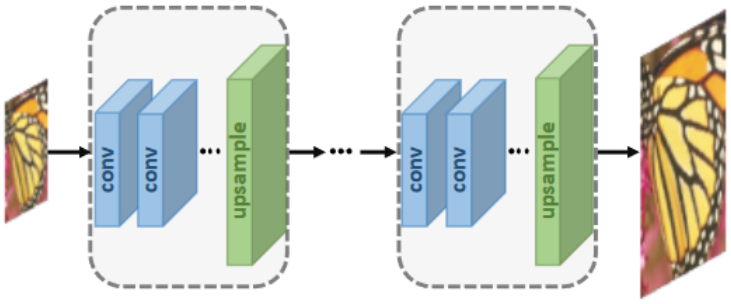

Figura 2.14: Método de súper-resolución basado en aumentar la escala de forma progresiva. La imagen pasa primero por una retahíla de convoluciones representadas con diversos bloques azules. Y después pasa por el módulo de aumento de escala representado con un bloque verde. Esta estructura se repite varias veces, hasta obtener la resolución deseada. Fuente: [Wang et al., 2020b].

menor tamaño. Así pues, este método sería computacionalmente más liviano que *Pre-Upsampling*. Pero en contraparte, a medida que se aumenta el factor de aumento de escala, la tarea se vuelve cada vez más difícil de realizar con una única capa de aumento de escala.

- ***Progressive Upsampling***

  Solucionando el problema del factor de aumento de escala alto, surgió este método. La idea es dividir una tarea compleja en varias más simples. Es decir, en lugar de aumentar la escala de la imagen mediante una única capa, se aumenta de forma progresiva, hasta obtener la resolución deseada (ver Figura 2.14).

- ***Iterative Up and Down Sampling***

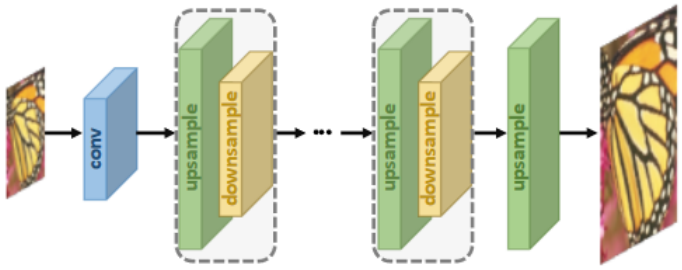

Figura 2.15: Método de súper-resolución basado en aumentar y reducir la escala de forma iterativa. La imagen pasa por una primera convolución representada con un bloque azul. Y después pasa por el módulo de aumento de escala representado con un bloque verde seguido de otro módulo de reducción de escala. Esta estructura se repite varias veces, hasta finalmente con un módulo de aumento de escala obtener la resolución deseada. Fuente: [Wang et al., 2020b].

Este método se basa en alternar entre distintas escalas de la imagen, aumentando y reduciéndola repetidas veces. Este esquema es capaz de extraer mejor las relaciones profundas entre las imágenes de alta y baja resolución, pudiendo proporcionar una reconstrucción de mayor calidad [Wang et al., 2020b] (ver Figura 2.15).

Para aumentar la escala de las imágenes también existen varias alternativas:

- **Convolución de subpíxeles**

  La técnica es más conocida como *subpixel convolution*. En esta técnica propuesta por [Shi et al., 2016], para aumentar la escala de la imagen por un factor de 2, primeramente se debe de aumentar el número de canales de la imagen por un factor de 4. Esto se puede realizar mediante una capa convolucional. Posteriormente combinando distintos canales aumentamos la escala de la imagen mientras reducimos el número de canales. Más concretamente, cada uno de los nuevos canales estará formado por una combinación entrelazada de los píxeles que forman 4 canales distintos. Dejando una imagen con 2 veces el tamaño inicial y el mismo número de canales. Esta operación la podemos ver reflejada en la Figura 2.16.

- **Convolución transpuesta**

  También conocida como deconvolución.

- **Interpolación**

  Dentro de esta categoría existen varios métodos. Algunos de los métodos tradicionales son:

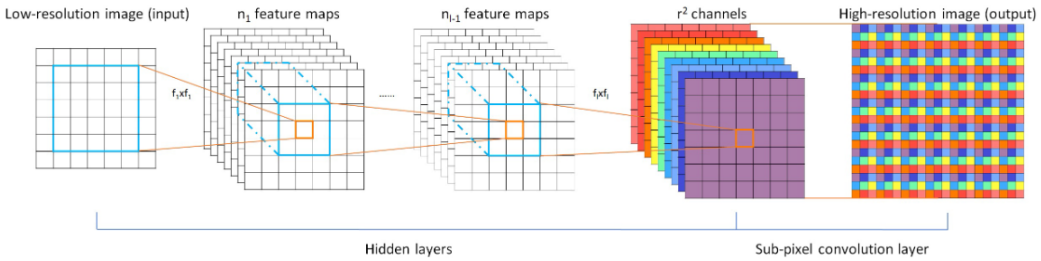

Figura 2.16: Técnica de convolución de subpíxeles. De izquierda a derecha, a partir de la imagen de entrada, podemos ver representado en diversos cuadrados blancos, uno detrás de otro, los mapas de características extraídas por las diversas convoluciones aplicadas. Cada sub-cuadrado, corresponde a un píxel de ese mapa de profundidad. Después, podemos ver la operación de convolución de subpíxeles. A partir de los mapas de profundidad, primeramente se aumenta el número de canales mediante una convolución, en la figura cada canal está coloreado con un color diferente. Y posteriormente podemos ver la combinación entrelazada de los píxeles de cada canal, esta será la imagen de salida, imagen que cuenta con una resolución mayor a la de entrada. Fuente: [Shi et al., 2016].

- Reescalado por vecinos próximos.
- Reescalado bilineal.
- Reescalado bicúbico.

Dentro de las estrategias de aprendizaje profundo nos encontramos con varias funciones de pérdida. Dadas dos imágenes de tamaño $m \times n$, la original x y la predicha y, las funciones de pérdida habituales se asemejan a las funciones utilizadas en la etapa de evaluación: error cuadrático medio (Ecuación 2.4), error absoluto medio (Ecuación 2.12) y función de perdida SSIM (Ecuación 2.13).

El error absoluto medio, traducido al inglés como *Mean Absolute Error* (MAE), es similar a MSE. Pero con el matiz de que en lugar de elevar al cuadrado la diferencia entre valores, MAE trabaja con su valor absoluto. Por otro lado, la función de perdida SSIM, simplemente se traduce de forma que cuanto menor sea el valor, mejor sea la predicción. Para ello se trabajará con la diferencia entre una predicción con similitud estructural perfecta, y la similitud estructural de la imagen predicha.

$$MAE(x,y) = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}|x(i,j) - y(i,j)| \tag{2.12}$$

$$L_{SSIM}(x,y) = 1 - SSIM(x,y) \tag{2.13}$$

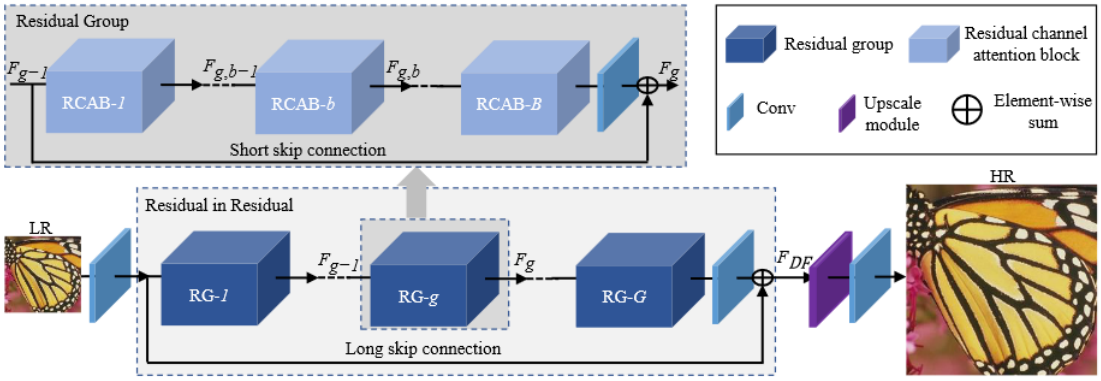

Figura 2.17: Arquitectura RCAN. En la zona inferior podemos ver la arquitectura de la red RCAN, compuesta por varios grupos residuales. El grupo residual, compuesto de bloques RCAB, está definido en la zona superior. Para ver el bloque RCAB, ver Figura 3.5. Los diversos elementos presentes en la arquitectura se encuentran definidos en la zona superior derecha. En azul marino se muestran los bloques de los grupos residuales. En azul claro se muestran los bloques RCAB. En bloques azules de menor grosor, se muestran las convoluciones. En bloques morados de menor grosor, se muestran los módulos de aumento de escala. Y finalmente, con una cruz sobre un círculo se define la operación de suma, elemento a elemento, aplicada en las *skip connections*. Fuente: [Zhang et al., 2018a].

2.2.3. Arquitectura RCAN

Entre las arquitecturas que mejores resultados están obteniendo para la tarea de súper-resolución, se encuentra la arquitectura RCAN [Zhang et al., 2018a]. El nombre proviene del acrónimo de *Residual Channel Attention Networks*, haciendo referencia a la forma de trabajar que tiene la red. Como bien dice su nombre, la red hace uso de bloques residuales, en los que se ha añadido atención por canales. Esta atención sirve para indicar a qué canal darle más importancia sobre el resto. Siendo el canal la tercera dimensión de las imágenes.

Esta arquitectura hace uso del método de *post-Upsampling*. Y cuenta con conexiones residuales a distintas profundidades tal y como se puede ver en la Figura 2.17. Cada una en una de las secciones de la red. Cuenta con una primera profundidad en la sección *"Residual in Residual"* donde se encuentran los grupos residuales. Una segunda profundidad, en la sección de los grupos residuales, donde se encuentran los bloques *Residual Channel Attention Block* (RCAB), se pueden ver los bloques en la Figura 3.5. Una tercera profundidad, en la sección de los RCAB, donde se encuentra la atención. Y una cuarta y última profundidad en la sección de atención por canales.

La atención por canales se calcula partiendo de la señal tratada en cada bloque residual. Esta operación se puede ver representada en la Figura 2.18. Primero se aplica la opera-

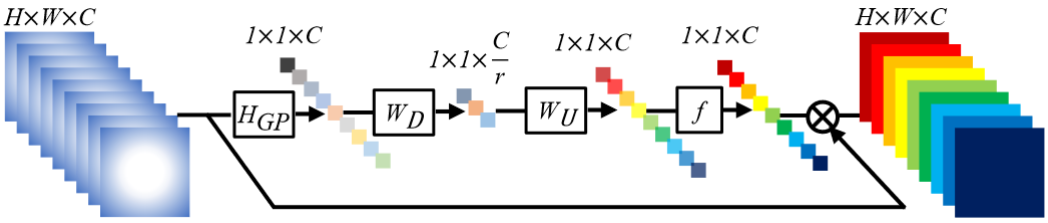

Figura 2.18: Cálculo de atención por canales. De izquierda a derecha podemos ver el proceso del cálculo de atención por canales. El tamaño de los datos con los que se está trabajando en cada momento está reflejado encima de cada etapa. Se representa con H_{GP} la operacion de *GlobalAveragePooling2D*. Con W_D se señala una convolución que da como resultado un menor número de canales. Con W_U se señala una convolución que da como resultado un mayor número de canales. Con f se señala la función de activación sigmoidal. Y con una X sobre un círculo se representa la operación de multiplicación elemento a elemento, aplicado a la conexion residual, donde cada mapa de características se multiplica por la atención calculada para dicho canal. Fuente: [Zhang et al., 2018a].

ción (H_{GP}) de *GlobalAveragePooling2D* a las imágenes, donde en cada canal se extrae el valor promedio de ese canal, reduciendo las imágenes del tamaño $H \times W \times C$ al tamaño $1 \times 1 \times C$. Seguidamente se aplica una convolución (W_D) para reducir el número de canales por un cierto ratio de reducción (r). Se aplica la función de activación *Rectified Linear Unit* (ReLU). Después se hace uso de una segunda convolución (W_U) con la que se recuperan el número de canales. Y se aplica la función de activación sigmoidal (f). Esto da como resultado un valor entre 0 y 1 para cada canal, que representa el nivel de atención a cada canal. Finalmente se multiplica la atención, a la señal tratada en el bloque residual, multiplicando así cada canal por su atención.

Un punto a tener en cuenta es que exceptuando el módulo de atención por canales, la arquitectura RCAN mantiene en todo momento el número de canales. La primera convolución y la última son aquellas que adaptan el número de canales con los que trabaja la red al tipo de imágenes con las que trabajamos, en formato RGB.

Los canales pertenecientes a aquellas imágenes en etapas intermedias no tienen por qué mantener una relación como las que se pueden encontrar habitualmente en los formatos conocidos. Formatos como puedan ser RGB o RGBA. La red descompone la imagen en distintos canales, y en cada uno almacena cierta información de la imagen. El contenido varía en función de lo que la red haya aprendido por lo que no se puede definir el significado de cada canal, a priori. Es por ello que estas imágenes son difícilmente interpretables. A lo largo del proyecto se trabajará únicamente haciendo uso de la imagen resultante al final de la red. Esto también se aplica a las redes U-Net y ResUNet.

Puesto que la salida de la red se corrige buscando una mayor semejanza con la imagen deseada, y esta mantiene la distribución de canales RGB, implícitamente la red aprende a dar como resultado imágenes RGB. Así pues, la imagen de salida es más fácilmente interpretable.

3. CAPÍTULO

Arquitecturas desarrolladas

3.1. Conjunto de datos

Puesto que se va a trabajar con la tarea de segmentación semántica, el conjunto de datos, o *dataset*, debe contener imágenes y sus correspondientes anotaciones. Para la tarea de súper-resolución, hemos optado por reducir artificialmente el tamaño de la imagen para simular las imágenes a baja resolución. Así pues, no será necesario que el conjunto de datos disponga de imágenes reales a baja y alta resolución. Por otro lado, considerando la capacidad de cómputo de la que contamos, la profundidad de las redes con las que se van a trabajar, y el tiempo del que disponemos, no hemos querido optar por conjuntos de datos excesivamente grandes.

Tras considerar varios conjuntos de datos, entre aquellos que cumplían los requisitos mencionados, se ha optado por usar *"The Oxford-IIIT Pet dataset"* [Parkhi et al., 2012]. Un conjunto de datos con una amplia variedad de imágenes de gatos y perros en distintas situaciones y entornos, obtenidas con distintos dispositivos. Este conjunto de datos consta de imágenes anotadas con 3 clases: animal, contorno del animal y fondo. El hecho de contar con una clase para el contorno es interesante, ya que ayuda a la red a la hora de realizar la tarea [Chen et al., 2016].

El conjunto de datos está preparado para poder realizar tareas de clasificación. Es por eso que tiene una distribución muy balanceada de la cantidad de imágenes por cada raza presente, aproximadamente 200 imágenes por raza. La distribución de animales, en cambio, no está tan balanceada, puesto que hay cerca del doble de razas de perros que de gatos,

también hay casi el doble de imágenes de perros que de gatos (4978 imágenes de perros y 2371 imágenes de gatos). Tras evaluar los modelos usando un conjunto de test únicamente con perros y otro únicamente con gatos, hemos visto que si existe una diferencia entre los resultados obtenidos para unos y otros. Pero aun así, se obtienen buenos resultados para ambos animales en ambas tareas, por lo que no tendremos esta diferencia en cuenta.

La distribución de razas y animales se mantiene tanto en el conjunto de entrenamiento como en el de test, que vienen ya preparados. Esto es especialmente interesante a la hora de evaluar el modelo entrenado, ya que el modelo podría ser sensible a esa distribución y los resultados de validación o test no serían tan representativos si la misma cambiase.

Para evaluar el modelo a medida que se entrena, hemos creado nuestro propio conjunto de evaluación a partir del 20 % de los datos de entrenamiento, manteniendo una distribución balanceada de los animales y razas presentes. Inicialmente cada conjunto poseía el 50 % del número total de las imágenes. Tras realizar un tercer conjunto de datos, la distribución de imágenes ha quedado de la siguiente manera:

- Conjunto de entrenamiento con 2944 imágenes (40 % de las imágenes totales).
- Conjunto de test con 3669 imágenes (50 % de las imágenes totales).
- Conjunto de validación con 736 imágenes (10 % de las imágenes totales).

3.1.1. Preprocesamiento

No todas las imágenes del conjunto de datos con el que vamos a trabajar tienen el mismo número de canales. Por lo tanto, el primer paso ha sido convertir todas al formato RGB, con 3 canales. Las imágenes también tienen una amplia variedad de tamaños, por lo que el siguiente paso ha sido aplicar un reescalado bilineal a todas las imágenes. Para ello hemos usado un tamaño que sea fácilmente manejable por las redes neuronales, en lo que a requisitos de cómputo se refiere. Las imágenes a alta resolución dispondrán de un tamaño de 256×256 píxeles. Partiendo de estas imágenes, se ha realizado un segundo reescalado bilineal para generar las imágenes a baja resolución. Hasta obtener un tamaño de 128×128 píxeles para una reducción de factor 2, y un tamaño de 64×64 píxeles para una reducción de factor 4.

A diferencia de las imágenes RGB, las imágenes anotadas constan de un único canal con píxeles cuyos valores poseen 3 posibles valores, en función de la categoría a la que

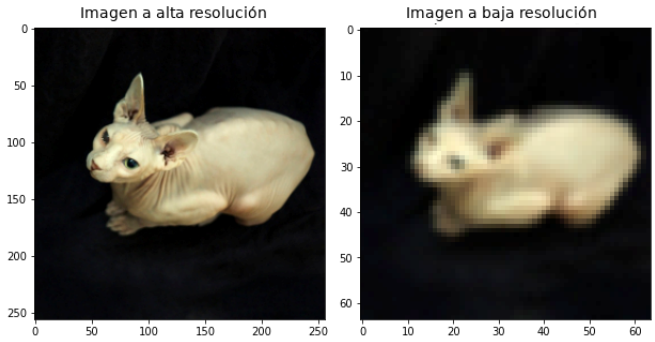

Figura 3.1: Resultado del proceso de obtención de imágenes a baja resolución. De izquierda a derecha: Imagen a alta resolución, de 256×256 píxeles, y la imagen a baja resolución, de 64×64 píxeles, resultado del proceso de obtención de imágenes a baja resolución.

pertenezca. Para mantener la coherencia de los valores de las imágenes anotadas, se ha hecho uso del método de reescalado de vecinos próximos. Una vez más, al tamaño de 256×256 píxeles.

Con el objetivo de replicar el desenfoque y calidad que obtendríamos para las imágenes a baja resolución, hemos hecho uso de una técnica habitual en el campo [Fang et al., 2019]. Para empezar, hemos añadido ruido con un filtro gaussiano (del paquete de skimage), con una desviación estándar de 3,0. Y posteriormente se ha inyectado un cierto ruido sal y pimienta. Este proceso se aplica antes de realizar el segundo reescalado al que se someten las imágenes a baja resolución. En la Figura 3.1 podemos ver el resultado de aplicar todo este proceso a una imagen.

Siguiendo la práctica habitual, hemos normalizado las imágenes, pasando del rango $[0, 255]$ al rango $[0, 1]$. Y para trabajar de forma más cómoda hemos reducido en 1 el valor original de cada clase, dejando los valores en un rango de $[0, 2]$. El 0 representa el animal, ya sea perro o gato, el 1 representa el fondo, y el 2 representa el contorno del animal.

Finalmente se ha hecho uso de técnicas de aumento de datos, en la etapa de entrenamiento. Al trabajar también con imágenes anotadas, no se puede aplicar cualquier técnica de aumento de datos, se deben de aplicar técnicas que no afecten a los valores de los píxeles. Ejemplos de operaciones válidas serian: rotaciones de múltiplos de 90 grados, dar la vuelta de izquierda a derecha o dar la vuelta de arriba a abajo. En nuestro caso hemos optado por dar la vuelta de izquierda a derecha a las imágenes. Para mantener la relación entre la imagen de entrada y la imagen objetivo, de manera aleatoria, con una probabilidad del 50 %, se les darán la vuelta de izquierda a derecha a ambas imágenes, tanto la de entrada, como la objetivo.

3.2. Punto de referencia

Antes de empezar con el objetivo principal, se ha partido por la base de generar un punto de referencia con el cual poder comparar cada modelo. Para ello se ha hecho uso de la red especializada para cada tarea y una amplia exploración de hiperparámetros.

3.2.1. Arquitecturas base para segmentación semántica

Para la tarea de segmentación semántica hemos probado tanto la red U-Net como ResUNet, con una profundidad de 4 niveles. Puesto que la tarea consiste en generar una máscara con la clasificación de cada píxel, el enfoque utilizado ha sido: dada una imagen de tamaño $256 \times 256 \times 3$ píxeles ($altura \times anchura \times canal$) de entrada, en la salida devolver una señal de las mismas dimensiones, donde cada canal de la salida corresponde a un mapa de probabilidades para una clase. Cada valor del este mapa c indica cómo de probable es que pertenezca a la clase c-ésima.

Sobre la salida se ha aplicado la función de pérdida de entropía cruzada categórica dispersa (ver sección 2.1.2). Con la imagen de salida, se puede generar una imagen de dimensiones $256 \times 256 \times 1$, donde el valor de cada píxel corresponde al canal con mayor probabilidad para ese píxel. De esta manera se obtiene la segmentación semántica deseada, sobre la que le aplicarán las métricas para evaluar su semejanza respecto a la imagen anotada original.

Para la construcción de la red ResUNet se ha hecho uso de bloques residuales como los presentados en la Figura 2.10. En la Figura 3.3 podemos ver la arquitectura de la red ResUNet, donde el bloque residual mencionado sería nuestro bloque de procesamiento. En el caso de la red U-Net, la Figura 2.10 representaría la arquitectura de la red de forma bastante fiel. Las únicas diferencias son: la convolución inicial desaparece, la conexión residual larga y la suma elemento a elemento final también desaparecen, y por último, el bloque de procesamiento cambia. El bloque pasaría a ser el presente en la Figura 3.2.

En ambas redes existe una convolución final que da la salida en el formato deseado. Esta última convolución cuenta con la función de activación lineal y un filtro de tamaño 1×1, para poder hacer la clasificación por píxel.

En lo que a canales se refiere, a medida que se reduce por 2 el tamaño de la imagen, se duplica el número de canales. Y de forma simétrica, en la segunda mitad, a medida que

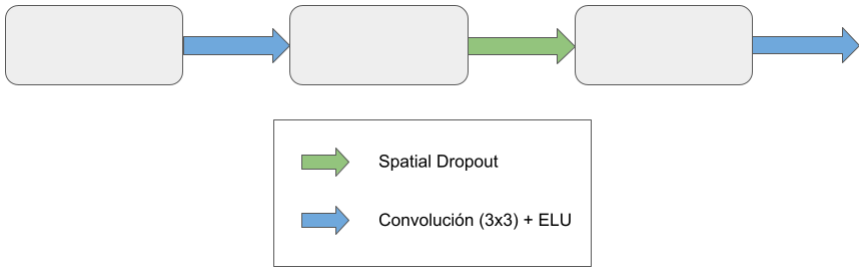

Figura 3.2: Bloque U-Net. Mientras que los datos en las distintas etapas se reflejan con bloques, las flechas señalan distintas operaciones. La leyenda se encuentra debajo, en un recuadro.

se duplica el tamaño de la imagen, se reduce por 2 el número de canales; dejando como resultado una entrada y salida de las mismas dimensiones.

En cuanto a las funciones de activación utilizadas a lo largo de la red, se ha probado tanto la función de activación ReLU como *Exponential Linear Unit* (ELU). Con un rendimiento y resultado similar, se ha optado por usar la función de activación ELU por obtener resultados ligeramente mejores. La función de activación ELU vendría definida como:

$$ELU(x) = \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases} \tag{3.1}$$

En caso de que x sea positiva, la respuesta será esta misma entrada (x). Y en caso de ser negativa, la curva tenderá suavemente a $-\alpha$, siendo α una constante positiva. En caso de $\alpha = 0$ nos encontraríamos frente a un caso particular, donde la respuesta de la función ELU será idéntica a la función ReLU. A lo largo del proyecto trabajaremos con el valor por defecto, es decir, $\alpha = 1$.

3.2.2. Arquitectura base para súper-resolución

Para la tarea de súper-resolución se ha hecho uso de la arquitectura RCAN propuesta por [Zhang et al., 2018a]. Dada una imagen de entrada a baja resolución (ver sección 3.1.1) de dimensiones $64 \times 64 \times 3$ (para un aumento de resolución de factor 4, o $128 \times 128 \times 3$ para un aumento de resolución de factor 2), se devuelve una imagen de dimensiones $256 \times 256 \times 3$.

El módulo de aumento de escala ubicado al final de la red, está compuesto por la técnica de convolución de subpíxeles, con un factor de 2, y una posterior función de activación ReLU. Esta estructura se repetirá tantas veces como sea necesario para alcanzar el factor

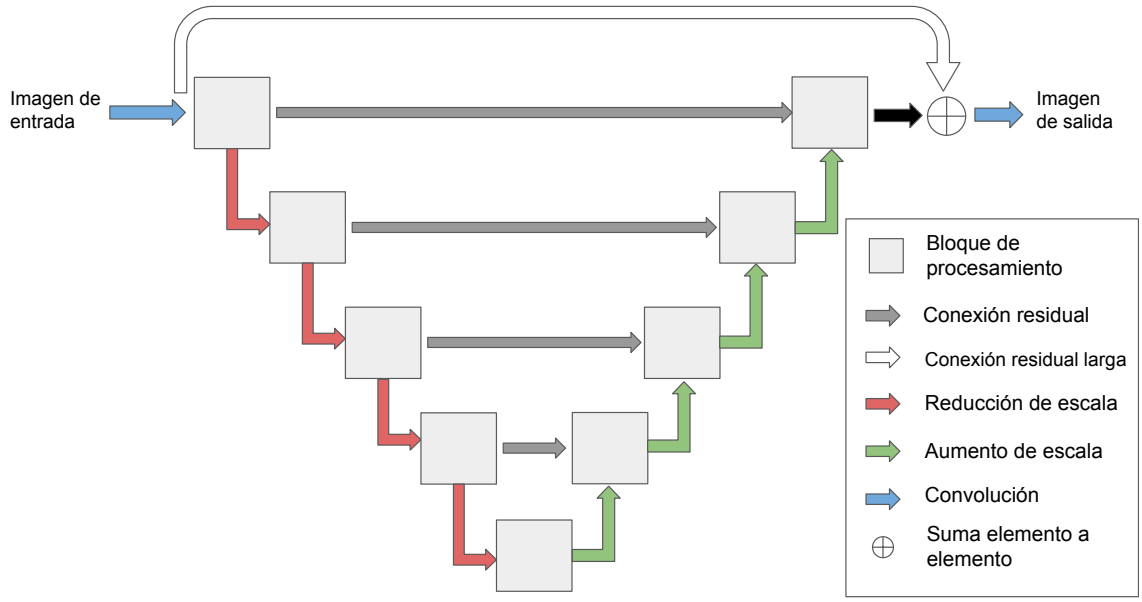

Figura 3.3: Arquitectura ResUNet. En el centro se encuentra la base de la arquitectura ResUNet. A la derecha, dentro de un recuadro se encuentra la leyenda.

de aumento final deseado. Por último se reconstruirá la imagen final mediante una última convolución, con un filtro de tamaño 3×3, que dará como resultado una imagen de $256 \times 256 \times 3$.

Contando con la imagen a alta resolución predicha y original, se ha hecho uso de la función de pérdida de MSE (Ecuación 2.4). Sobre estas dos mismas imágenes se han aplicado también las métricas SSIM y PSNR mencionadas en la Sección 2.2.1.

El número inicial de filtros se define en la primera convolución de la arquitectura. Este número limitará la capacidad de la red. Por lo tanto, el número de filtros iniciales es importante tanto para el resultado obtenido como para el tiempo de ejecución. Esto también se aplica a las redes U-Net y ResUNet.

Los detalles de implementación no especificados son los propuestos en el artículo original de las RCAN [Zhang et al., 2018a].

3.3. Adaptación por tarea de las arquitecturas base

Partiendo de las redes implementadas para obtener el punto de referencia se han aplicado modificaciones en función de la tarea a realizar.

3.3.1. Adaptación de RCAN para segmentación

Para la tarea de segmentación con la arquitectura de RCAN, se ha extraído el módulo final para aumentar la escala. Pero se mantiene la convolución final para realizar la reconstrucción final con 3 canales. Sin embargo, ahora la convolución se realizará con un filtro de tamaño 1×1. De esta manera, tanto la entrada de la red como la salida serían imágenes de dimensiones $256 \times 256 \times 3$. Pero a diferencia de en súper-resolución, se hará uso de la función de pérdida de entropía cruzada categórica dispersa, con los mapas de segmentación predichos y el mapa de segmentación verdadero.

Hay que tener en cuenta que la arquitectura RCAN para súper-resolución hace uso del método *post-Upsampling*. Al quitar el módulo de aumento de escala final y abastecer la red desde un comienzo con imágenes de gran tamaño, el coste computacional aumenta de forma considerable.

3.3.2. Adaptación de redes tipo U-Net para súper-resolución

Para la tarea de súper-resolución con las redes ResUNet y U-Net, hemos tenido que romper la simetría de la arquitectura añadiendo una capa adicional para obtener como resultado una imagen de una resolución mayor a la dada.

Se han explorado 3 técnicas distintas para esta nueva capa de aumento de escala:

- **Convoluciones transpuestas**

  Tal y como se venía haciendo en la etapa deconvolucional, o de decodificación de la red.

- **Vecinos próximos**

  A diferencia del resto de métodos, este no altera el número de canales de la imagen. Puesto que también nos interesa obtener un cierto número de canales, se ha añadido una capa convolucional, tras el proceso de aumento de escala.

- **Convolución de subpíxeles**

  Al igual que se ha hecho en la arquitectura RCAN empleada para súper-resolución.

De todas las técnicas mencionadas aquella que mejor resultados ha obtenido ha sido la técnica de convolución de subpíxeles. Por lo que, se ha seguido usando para posteriores ejecuciones.

Una vez decidida la técnica, se debe de decidir la ubicación de esta: antes (*pre-upsampling*) o después (*post-upsampling*) de la red principal. De las pruebas realizadas el método que mejor resultado ha dado ha sido el de *pre-upsampling*, es decir, aumentar la escala al principio. Para diferenciarlas, de aquí en adelante ambas redes se nombrarán con el prefijo del método, es decir, pre-U-Net y pre-ResUNet.

Para obtener una imagen de salida con los 3 canales deseados, se ha modificado la última capa convolucional, poniéndole un filtro de tamaño 3×3. La función de activación utilizada para esta última capa, es la función de activación lineal.

Las arquitecturas tendrían una gran similitud a las arquitecturas vistas en segmentación, en la Figura 3.3, las diferencias son, por un lado, la primera convolución que se sustituye por el módulo de aumento de escala escogido, y por otro lado la última convolución que pasaría de tener un filtro de tamaño 1×1 a 3×3.

3.3.3. Arquitectura base multitarea

Para las redes multitarea se han tomado como punto de partida las arquitecturas de súper-resolución, tanto pre-ResUNet como RCAN. Con la diferencia de que la última convolución aplicada para la reconstrucción final de la imagen ha sido sustituida por 2 convoluciones. Ambas se aplican sobre la misma entrada, pero cada una realizará una reconstrucción distinta. La primera convolución con un filtro de tamaño 3×3 será la encargada de generar la imagen. La segunda convolución con un filtro de tamaño 1×1 será la encargada de generar los mapas de segmentación. Este cambio se puede ver ilustrado en la Figura 3.4.

De esta manera, dada una imagen a baja calidad de dimensiones $64 \times 64 \times 3$ para un aumento de resolución de factor 4 (o $128 \times 128 \times 3$ para un aumento de resolución de factor 2), se generarán dos salidas, ambas con dimensiones $256 \times 256 \times 3$. Una salida será la imagen súper-resuelta, y la otra, un mapa de segmentación súper-resuelto.

La función de pérdida consta de una combinación ponderada (mediante un factor α) de

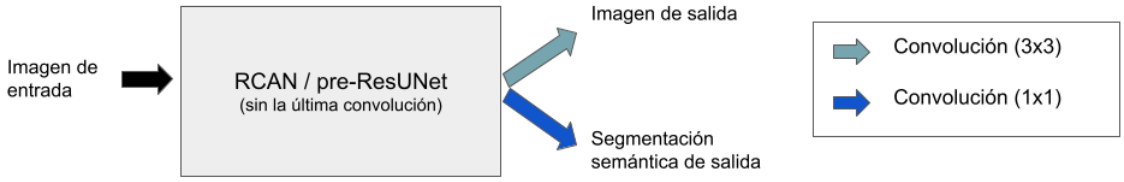

Figura 3.4: Arquitectura multitarea. El cuerpo de las redes RCAN o pre-ResUNet se mantienen, por lo que se han representado con una caja. La única diferencia es la ausencia de la convolución final en ambas arquitecturas. A continuación, se ilustra mediante flechas las nuevas convoluciones finales, dando cada una de ellas una salida distinta. A la derecha, dentro de un recuadro se encuentra la leyenda.

las dos funciones de pérdida utilizadas para cada tarea por separado, es decir, entropía cruzada categórica dispersa y MSE:

$$L(x_1, x_2, y_1, y_2) = \alpha \cdot MSE(x_1, y_1) + (1 - \alpha) \cdot SCCE(x_2, y_2) \tag{3.2}$$

Siendo α una constante dentro del rango $[0, 1]$; x_1 la imagen súper-resuelta; x_2 los mapas de segmentación generados; y_1 la imagen a alta resolución ideal e y_2 el mapa de segmentación ideal. Tomando como referencia los resultados experimentales obtenidos en ambas tareas por separado, se ha optado por equilibrar ambas funciones de pérdida con $\alpha = 0{,}996$.

3.4. Exploración de arquitecturas

A continuación se explicarán cuáles han sido las alteraciones, modificaciones o combinaciones más destacables que se han explorado. Para empezar, se ha explorado una nueva arquitectura híbrida entre ResUNet y RCAN. Y en un segundo experimento, se ha probado a alterar la arquitectura RCAN.

3.4.1. Arquitectura híbrida ResCAUNet

Tras analizar las arquitecturas ResUNet y RCAN, hemos visto que hay varias similitudes, como que ambas están compuestas de bloques residuales. Pero a diferencia de RCAN, ResUNet reduce en varias ocasiones el tamaño de las imágenes, lo que hace la red más simple computacionalmente. Por otro lado, RCAN hace uso de atención por canales, lo

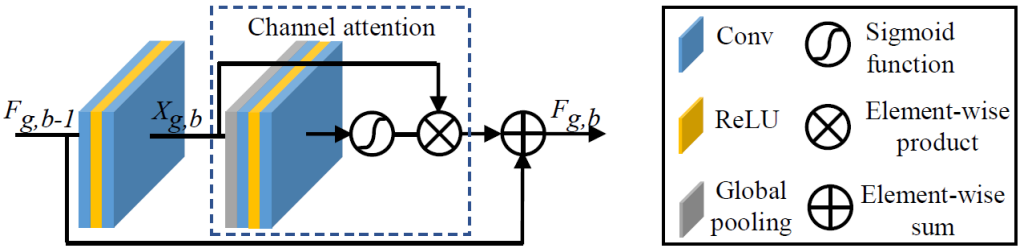

Figura 3.5: Bloque residual con atención por canales (RCAB), de RCAN. Se puede apreciar la estructura interna de los bloques RCAB, compuesta de diversos bloques. Los bloques están divididos por colores en función del tipo de operación realizada. El significado de cada elemento se indica en la zona derecha de la imagen. Dentro del bloque RCAB se puede ver marcado por líneas discontinuas, el bloque encargado del cálculo de la atención por canales. Fuente: [Zhang et al., 2018a].

que ayuda a priorizar entre los distintos contenidos de la imagen. Por este motivo, hemos probado a combinar ambas características en una única red a la que le hemos otorgado el nombre *Residual Channel Attention U-Net* (ResCAUNet).

La red consiste en una ResUNet a la que se le ha añadido una atención por canales dentro de cada bloque residual. La atención se calcula al final del bloque y se suma a la señal de salida, tal y como se realiza en los módulos *Residual Channel Attention Block* (RCAB) de la arquitectura RCAN. Podemos ver el módulo RCAB representado en la Figura 3.5.

Esta arquitectura híbrida es considerablemente más rápida que la arquitectura RCAN. Pero el cálculo de la atención añade un coste adicional, por lo que no es tan rápida como la arquitectura ResUNet. Esta red se ha probado tanto para segmentación como para súper-resolución, por separado y en multitarea, tal y como se ha hecho con el resto de arquitecturas.

3.4.2. RCAN sin atención

Tal y como se ha mencionado previamente, analizando las redes RCAN y ResUNet, por un lado se ha identificado la atención por canales de RCAN y por otro la reducción del tamaño de las imágenes de la ResUNet. En el experimento anterior se ha probado a reducir el tamaño y aplicar la atención. En este experimento, en cambio, ni se reducirá el tamaño de las imágenes, ni se aplicará la atención por canales. Para ello se utilizará la arquitectura RCAN, a la que hemos extraído la atención por canales. De esta manera, en ningún momento varía el número de canales de la imagen. El tamaño de la imagen única-

mente varía en el módulo de aumento de escala ubicado al final de la red, para la tarea de súper-resolución o multitarea.

El prescindir de la atención hace considerablemente más rápida a la red. Y al igual que con el resto de arquitecturas, esta red se ha probado tanto para segmentación como para súper-resolución, por separado y en multitarea.

4. CAPÍTULO

Experimentación

En este capítulo se mostrarán y analizarán los resultados obtenidos para cada tarea. De forma complementaria, se ha analizado superficialmente la capacidad de generalización adquirida por algunas de las redes entrenadas. Antes, se especificarán ciertos detalles acerca de las ejecuciones realizadas, y la exploración de hiperparámetros llevada a cabo.

Con el fin de generar experimentos reproducibles, se ha fijado una semilla para todos aquellos elementos pseudoaleatorios. Todas las redes han sido implementadas en el lenguaje de programación Python v3 [Van Rossum and Drake, 2009] con el principal uso de las siguientes librerías: Tensorflow v2.4.1 [Abadi et al., 2015], Keras [Chollet et al., 2015] y Numpy [Harris et al., 2020]. El código utilizado se encuentra accesible en `https://github.com/Aituni/SR-segm_DL`.

El entorno de desarrollo principal ha sido Google Colab con el uso de cuadernos Jupyter. Para acelerar el tiempo de ejecución se ha hecho uso de las tarjetas gráficas proporcionadas por la plataforma. Se tiene que tener en cuenta que la plataforma no siempre suministra a sus usuarios con la misma tarjeta gráfica. De forma excepcional, se han entrenado redes con el equipo personal y con el equipo proporcionado por el Donostia International Physics Center (DIPC).

4.1. Exploración de hiperparámetros

Dado el alto tiempo de ejecución necesario para entrenar la red RCAN, la búsqueda de hiperparámetros se ha realizado sobre las redes U-Net y ResUNet, para todas las tareas. Tras la búsqueda, los hiperparámetros utilizados para las redes basadas en RCAN han sido una combinación de los hiperparámetros del artículo [Zhang et al., 2018a] y los obtenidos en la búsqueda.

El proceso de la búsqueda ha consistido en aplicar un cambio cada vez, fijando el resto de parámetros. En caso de que el cambio mejore los resultados, se mantiene el parámetro y en caso contrario se retoma el anterior. Tras el proceso de búsqueda, se ha hecho uso de los mejores hiperparámetros obtenidos, incluso en el resto de redes. En alguna ocasión, se ha explorado un poco más si se daba el caso de identificar algún patrón que nos indicase la necesidad de algún cambio. Por ejemplo, cuando los resultados de validación no parecían haber convergido, lo que nos indica que el modelo podría seguir aprendiendo, aumentamos las épocas.

Para la tarea de súper-resolución, cuanto mayor sea el factor de aumento, mayor es la complejidad de la tarea. Por ello, se ha explorado más extensamente el factor de aumento 4. El margen de mejora es mayor, y con ello, la evaluación cualitativa es más simple. La búsqueda de hiperparámetros se encuentra reflejada en las tablas del apéndice B.

Una vez terminado todo el proceso de búsqueda, con los mejores hiperparámetros se han realizado 5 ejecuciones de cada una de las redes que forman el punto de referencia. Estas ejecuciones se han realizado en una GPU (Nvidia GTX 2080ti), y no cuentan con una semilla explícitamente definida para los elementos pseudoaleatorios. En el caso de la red de súper-resolución, no se han realizado las 5 ejecuciones para el aumento de factor 2, por falta de tiempo.

4.2. Resultados para la tarea de segmentación semántica

Podemos ver en la Tabla 4.1 los valores obtenidos para cada red en la tarea de segmentación semántica. Estos valores se han obtenido con las mejores asignaciones de hiperparámetros encontradas. Podemos apreciar que ResUNet obtiene ligeramente mejores resultados que U-Net. Por otro lado, si comparamos ResUNet con su derivado ResCAUNet, la atención no parece jugar un papel muy importante. ResCAUNet únicamente logra me-

Arquitectura	**mIoU**	σ
U-Net*	0,7284	$2{,}5369e-06$
ResUNet*$^{+}$	0,7330	$3{,}3646e-05$
RCAN*	**0,7658**	$2{,}6445e-06$
RCAN (sin atención)	0,7573	–
ResCAUnet	0,7354	–

* Media obtenida de 5 ejecuciones.
$^{+}$ Punto de referencia para esta tarea.

Tabla 4.1: Resumen de los resultados obtenidos en segmentación.

jorar de forma muy poco significativa el resultado de ResUNet. En cambio, en RCAN la atención supone una diferencia mayor que la presente en ResUNet.

Hay que destacar el resultado de RCAN, ya que ha conseguido superar de forma notoria el punto de referencia impuesto por la red ResUNet. Este suceso se ve mejor representado en la Figura 4.3. Podemos ver como no solo supera tanto a U-Net como a ResUNet, sino que además, lo hace de forma muy consistente. Los resultados obtenidos para la ResUNet, en cambio, fluctúan más. Por otra parte, pese a obtener ligeramente peores resultados con U-Net, los resultados son más estables que en ResUNet.

Podemos ver la media, desviación y tiempo medio de entrenamiento de cada red en la Tabla 4.2. Tal y como se indica en la tabla, los mejores resultados los obtiene RCAN, pero el tiempo de entrenamiento es bastante superior también.

Hay que tener en cuenta que una vez entrenada la red, la inferencia es mucho más rápida. En el caso de RCAN el tiempo de inferencia es de cerca de 1 segundo por imagen. Con unidades de tiempo mucho más pequeñas, la diferencia de tiempos en inferencia no son tan notorias. Podemos apreciar un par de imágenes procesadas por RCAN en las Figuras 4.1 y 4.2.

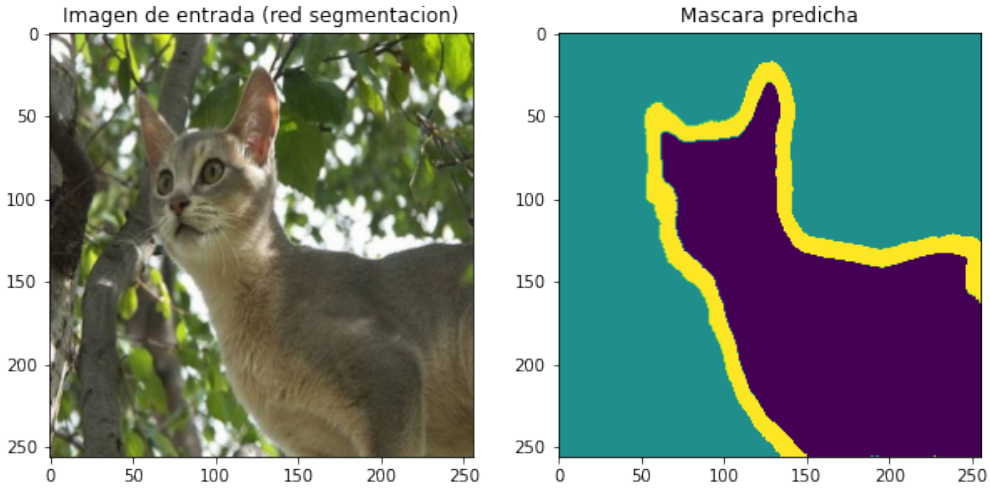

Figura 4.1: Primer ejemplo de los resultados para segmentación semántica. De izquierda a derecha: imagen dada como entrada de la red RCAN, y la máscara predicha por la red.

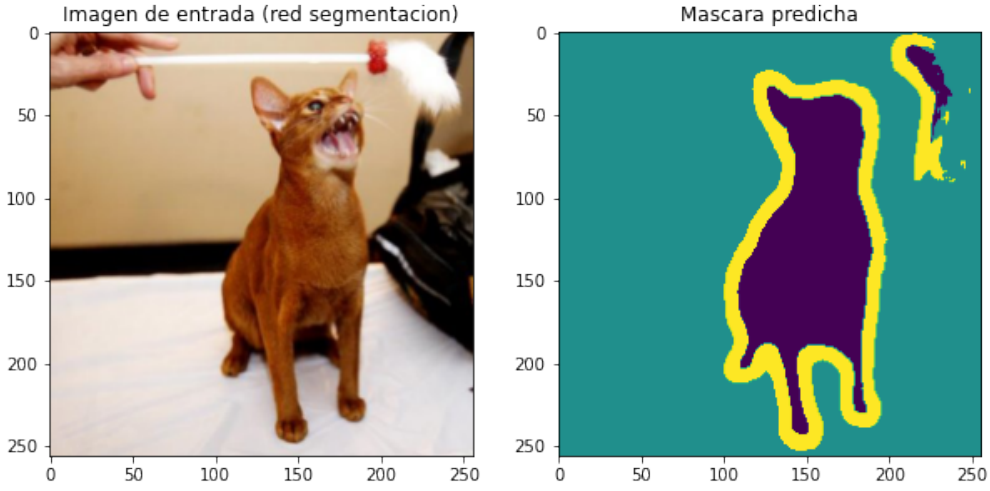

Figura 4.2: Segundo ejemplo de los resultados para segmentación semántica. De izquierda a derecha: imagen dada como entrada de la red RCAN, y la máscara predicha por la red.

Arquitectura	**mIoU**		**Tiempo medio**
	μ	σ	
pre-U-Net	0.7284	$2{,}5369e-06$	21 *min*
pre-ResUNet	0.7330	$3{,}3646e-05$	25 *min*
RCAN	**0.7658**	$2{,}6445e-06$	10 *h* 38 *min*

Tabla 4.2: Resumen de los resultados obtenidos y el tiempo medio de entrenamiento de 5 ejecuciones para segmentación semántica.

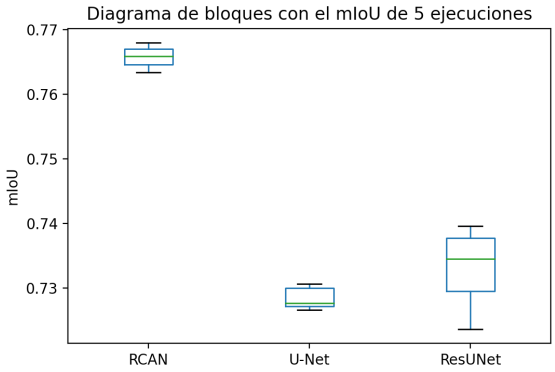

Figura 4.3: Diagrama de bloques con los resultados de 5 ejecuciones en segmentación semántica. En el diagrama de bloques se encuentran los resultados de las redes: RCAN, U-Net y ResUNet, respectivamente.

4.3. Resultados para la tarea de súper-resolución

Podemos ver en la Tabla 4.3 los valores obtenidos para cada red en la tarea de súper-resolución con un factor de aumento de 4. Estos valores se han obtenido con las mejores asignaciones de hiperparámetros encontradas. Podemos apreciar que en cualquier caso, las redes de aprendizaje profundo obtienen un resultado significativamente superior a los obtenidos con métodos tradicionales de interpolación.

Por otro lado, tanto pre-U-Net como pre-ResUNet han sido capaces de igualar el punto de referencia establecido por RCAN. Llegando incluso a superar a RCAN de forma poco significativa. Mientras que la atención parece haberle ayudado a pre-ResCAUNet frente a pre-ResUNet. En RCAN parece no seguir la misma tendencia. RCAN sin atención ha sido la red que mejores resultados ha obtenido. En cualquier caso, la variación de los resultados entre distintas arquitecturas ha sido muy sutil. Esto también se aplica a los resultados obtenidos en el aumento de factor 2, reflejados en la Tabla 4.4, con la única diferencia de que esta vez RCAN ha sido la red que mejores resultados ha obtenido.

En la Figura 4.6 y la Tabla 4.5 podemos ver más detalles acerca de los resultados obtenidos para las 5 repeticiones. Se puede apreciar que mientras pre-ResUNet y pre-U-Net obtienen

Arquitectura	SSIM	σ	PSNR	σ
pre-U-Net*	0,8563	$6{,}1147e-07$	28.0593	0,0014
pre-ResUNet*	0,8572	$3{,}5425e-06$	28.0949	0,0115
RCAN*$^{+}$	0,8546	$1{,}3379e-05$	27.8628	0,0544
RCAN (sin atención)	**0,8609**	–	**28,2426**	–
pre-ResCAUNet	0,8584	–	28,1717	–

Algoritmo de interpolación	SSIM	PSNR
Vecinos próximos	0,7450	23,7003
Bilineal	0,7558	23,8519
Bicúbico	**0,7655**	**24,2047**

* Media obtenida de 5 ejecuciones.
$^{+}$ Punto de referencia para esta tarea.

Tabla 4.3: Resumen de los resultados obtenidos en súper-resolución con un aumento de factor 4.

Arquitectura	SSIM	PSNR
pre-U-Net	0,8821	29,1541
pre-ResUNet	0,8852	29,3466
RCAN^{+}	**0,8864**	**29,4274**

$^{+}$ Punto de referencia para esta tarea.

Tabla 4.4: Resumen de los resultados obtenidos en súper-resolución con un aumento de factor 2.

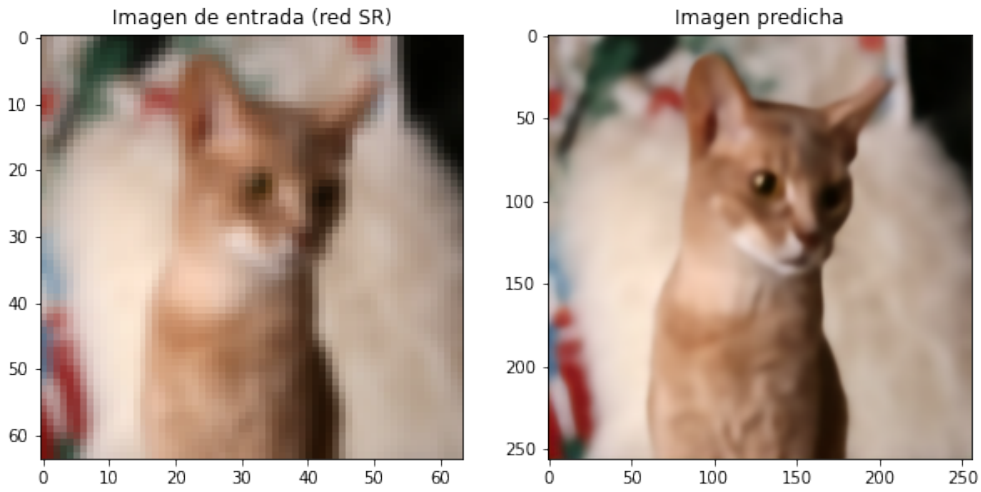

Figura 4.4: Primer ejemplo de los resultados para súper-resolución. De izquierda a derecha: imagen dada como entrada de la red pre-ResUNet, y la imagen súper-resuelta por la red.

Arquitectura	SSIM		PSNR		Tiempo medio
	μ	σ	μ	σ	
pre-U-Net	0,8563	$6{,}1147e-07$	28.0593	0,0014	56 *min*
pre-ResUNet	**0,8572**	$3{,}5425e-06$	**28.0949**	0,0115	1 *h* 10 *min*
RCAN	0,8546	$1{,}3379e-05$	27.8628	0,0544	4 *h* 38 *min*

Tabla 4.5: Resumen de los resultados obtenidos y el tiempo medio de 5 ejecuciones para súper-resolución con un aumento de factor 4.

valores muy estables, RCAN varía considerablemente más. Además de superar a RCAN, tanto pre-ResUNet como pre-U-Net son redes más pequeñas y bastante más veloces que RCAN. Respecto a los valores atípicos presentes en las redes pre-U-Net y pre-ResUNet, ambos corresponden a ejecuciones que se han detenido antes que el resto, debido a la paciencia impuesta. Podemos ver un par de imágenes procesadas por pre-ResUNet en las Figuras 4.4 y 4.5.

4.4. Resultados de la arquitectura multitarea

En la Tabla 4.6, se muestran los resultados de la realización de ambas tareas de forma simultánea. Mientras los resultados para la tarea de súper-resolución se mantienen más

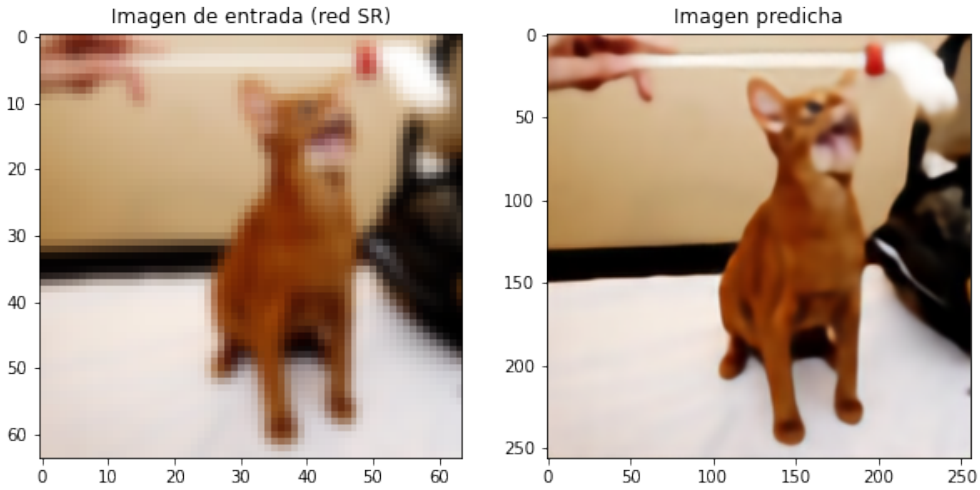

Figura 4.5: Segundo ejemplo de los resultados para súper-resolución. De izquierda a derecha: imagen dada como entrada de la red pre-ResUNet, y la imagen súper-resuelta por la red.

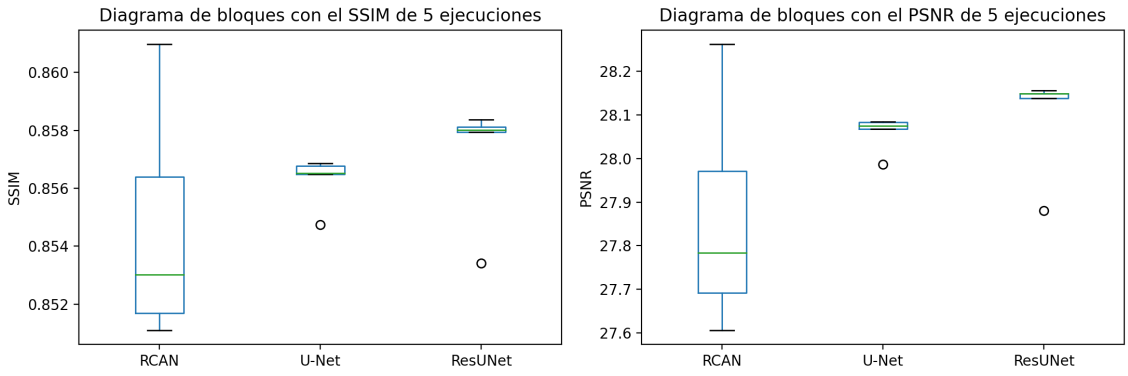

Figura 4.6: Diagrama de bloques con los resultados de 5 ejecuciones en súper-resolución En el diagrama de bloques se encuentran los resultados (SSIM y PSNR) para un aumento de factor 4, de las redes: RCAN, pre-U-Net, pre-ResUNet, respectivamente.

Arquitectura	mIoU	SSIM	PSNR
Punto de referencia*	0,7330	0,8546	27,8628
pre-ResUNet	**0,6940**	**0,8501**	**27,8044**
RCAN	0,6354	0,8443	27,3676
RCAN (sin atención)	0,6052	0,8349	26,9250
pre-ResCAUnet	0,6646	0,8477	27,6478

* Redes entrenadas para cada tarea por separado

Tabla 4.6: Resumen de los resultados obtenidos en multitarea, con un aumento de factor 4.

estables a lo largo de las distintas arquitecturas. los resultados para la tarea de segmentación semántica fluctúan más. A pesar de no obtener mejores resultados, las redes sí que han sido capaces de obtener valores que se aproximan al punto de referencia de cada tarea por separado. Hay que tener en cuenta que las redes están haciendo ambas tareas en una única arquitectura, lo que supone un aumento despreciable en el coste computacional respecto a las utilizadas para las tareas por separado.

Mientras en pre-ResCAUNet la atención perjudica a los resultados respecto a pre-ResUNet, en RCAN beneficia a ambas tareas. Pero aun con RCAN haciendo uso de la atención, se han obtenido peores resultados que con pre-ResCAUNet. Finalmente, es pre-ResUNet la red que mejores resultados ha obtenido en multitarea. Esta es además la red de menor tamaño entre las 4 presentes en la tabla, y con ello, la más rápida de todas. Podemos apreciar una imagen procesada por pre-ResUNet en la Figura 4.7.

4.5. Pruebas preliminares sobre capacidad de generalización de las redes entrenadas

Con el fin de comprobar la capacidad de generalizar de RCAN para segmentación semántica y de pre-ResUNet para súper-resolución (con factor de escala 4), se han realizado pruebas con imágenes fuera del conjunto inicial de datos.

Se puede apreciar que la red de súper-resolución, desempeña bien su papel indiferentemente del contenido de la imagen. Por otro lado, parece que la red de segmentación generaliza bien para animales que cuenten con pelaje. Como es el caso del conejo de la

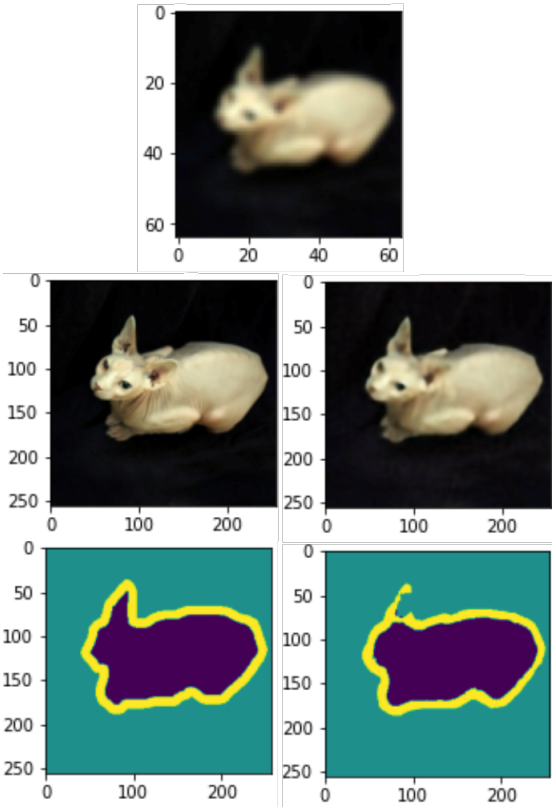

Figura 4.7: Ejemplo de resultados en multitarea. Empezando por arriba, por filas, de izquierda a derecha, nos encontramos con: imagen a baja resolución y entrada de la red; imagen a alta resolución esperada; imagen súper-resuelta obtenida; máscara de la imagen esperada; segmentación semántica súper-resuelta obtenida.

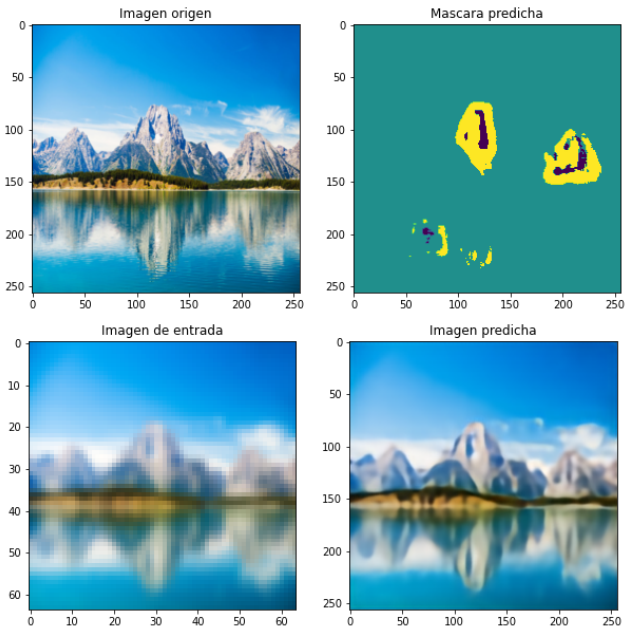

Figura 4.8: Primera prueba, imagen fuera del conjunto de datos. De izquierda a derecha, partiendo de la primera fila: entrada de la red de segmentación (imagen a alta resolución) y máscara de segmentación predicha. En la segunda fila: entrada de la red de súper-resolución (imagen a baja resolución) e imagen a alta resolución predicha. Fuente: astelus.com

Figura 4.10 o la ardilla de la Figura 4.11, a pesar de ser ambos animales que la red nunca había visto antes. También observamos como le cuesta bastante más clasificar animales sin pelaje, como puedan ser los anfibios. A modo de ejemplo podemos ver el camaleón de la Figura 4.12.

En el caso del tucán de la Figura 4.13, parece que toda la garganta y zona lateral de la cabeza repleta con un plumaje blanco, con una textura similar a la del pelaje, sí la clasifica. Pero esta zona es la única segmentada por la red como animal, como si el plumaje negro y el pico no formasen parte del animal.

Por otro lado parece que la red ha aprendido a segmentar elementos sobre un fondo de algún color, como pueda ser el ejemplo de la bicicleta (Figura 4.9). Es una segmentación mejorable, pero se puede apreciar claramente el intento. Por último, en el paisaje (Figura 4.8) podemos apreciar ciertos falsos positivos.

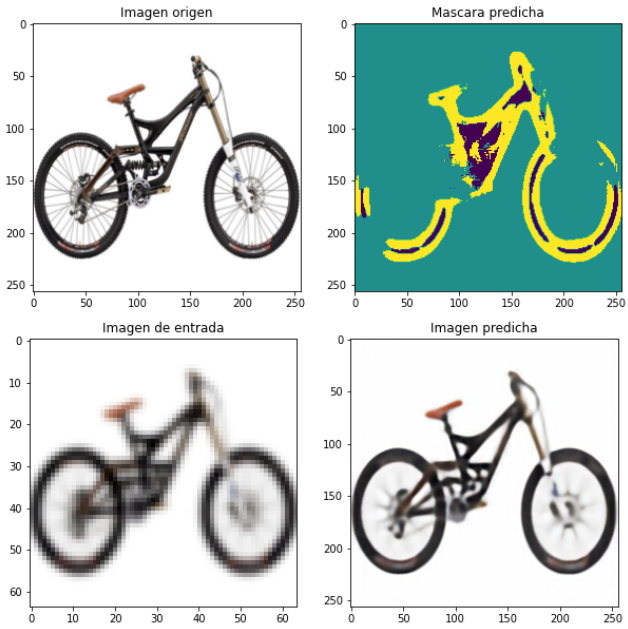

Figura 4.9: Segunda prueba, imagen fuera del conjunto de datos. De izquierda a derecha, partiendo de la primera fila: entrada de la red de segmentación (imagen a alta resolución) y máscara de segmentación predicha. En la segunda fila: entrada de la red de súper-resolución (imagen a baja resolución) e imagen a alta resolución predicha. Fuente: 1.bp.blogspot.com

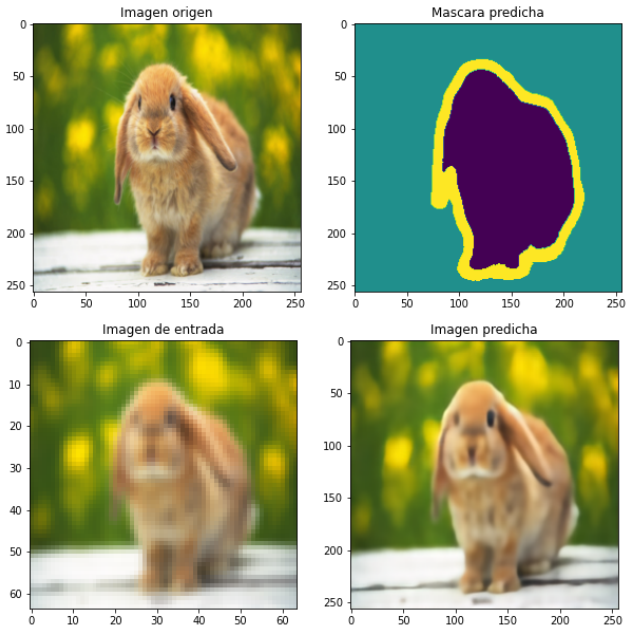

Figura 4.10: Tercera prueba, imagen fuera del conjunto de datos. De izquierda a derecha, partiendo de la primera fila: entrada de la red de segmentación (imagen a alta resolución) y máscara de segmentación predicha. En la segunda fila: entrada de la red de súper-resolución (imagen a baja resolución) e imagen a alta resolución predicha. Fuente: 3.bp.blogspot.com

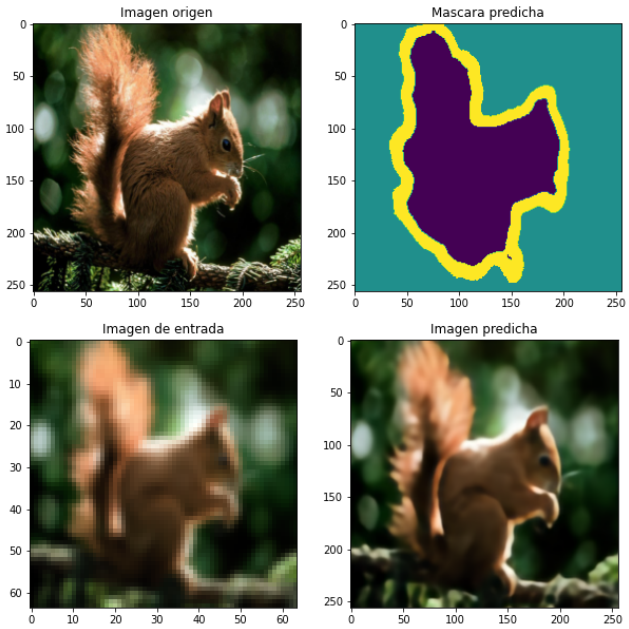

Figura 4.11: Cuarta prueba, imagen fuera del conjunto de datos. De izquierda a derecha, partiendo de la primera fila: entrada de la red de segmentación (imagen a alta resolución) y máscara de segmentación predicha. En la segunda fila: entrada de la red de súper-resolución (imagen a baja resolución) e imagen a alta resolución predicha. Fuente: 4.bp.blogspot.com

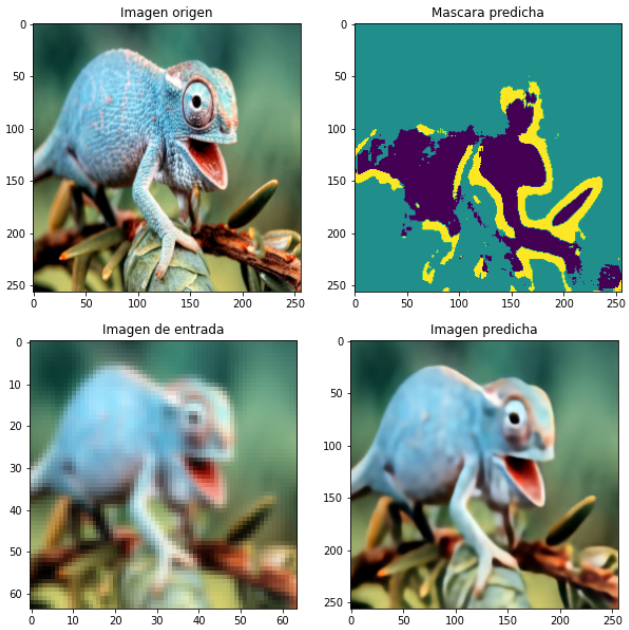

Figura 4.12: Quinta prueba, imagen fuera del conjunto de datos. De izquierda a derecha, partiendo de la primera fila: entrada de la red de segmentación (imagen a alta resolución) y máscara de segmentación predicha. En la segunda fila: entrada de la red de súper-resolución (imagen a baja resolución) e imagen a alta resolución predicha. Fuente: tailandfur.com

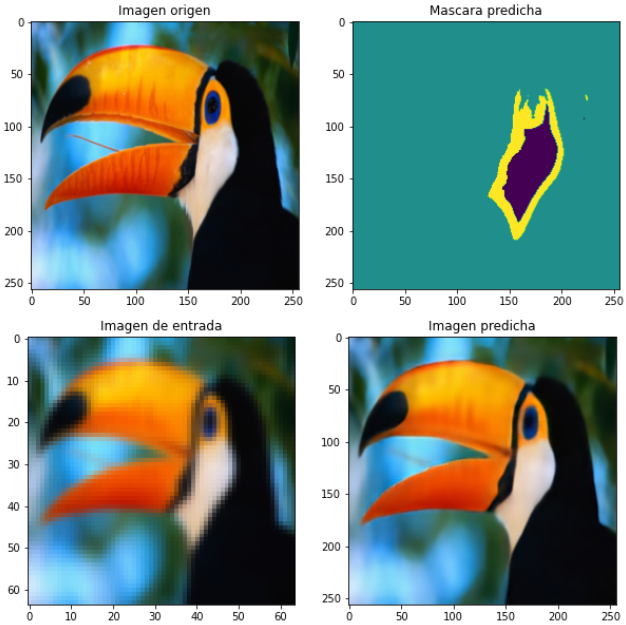

Figura 4.13: Sexta prueba, imagen fuera del conjunto de datos. De izquierda a derecha, partiendo de la primera fila: entrada de la red de segmentación (imagen a alta resolución) y máscara de segmentación predicha. En la segunda fila: entrada de la red de súper-resolución (imagen a baja resolución) e imagen a alta resolución predicha. Fuente: pixelstalk.net

5. CAPÍTULO

Conclusiones

A lo largo de este proyecto hemos visto en qué consisten las tareas de súper-resolución y segmentación semántica, y hemos explorado las distintas estrategias de aprendizaje profundo utilizadas para la resolución de ambas tareas. Además, hemos analizado el estado del arte para ambas tareas y finalmente lo hemos implementado, trabajando bajo la hipótesis de se pueden resolver ambas tareas satisfactoriamente con una misma red. Para ello, hemos realizado una amplia exploración de hiperparámetros, y desarrollado diversas pruebas, tanto por separado como en multitarea. De forma adicional, hemos probado dos nuevas arquitecturas en las que hemos tratado de combinar las cualidades de cada red por separado.

Tras analizar los resultados, podemos confirmar que las redes obtienen un buen desempeño en ambas tareas, sin importar su especialidad original, pudiendo incluso superar a las arquitecturas habitualmente utilizadas para esa tarea (por ejemplo, RCAN para segmentación). Y esto no se limita únicamente a las métricas obtenidas, también pueden lograr igualar los resultados haciendo uso de un tiempo mucho menor, y con una red de menor tamaño, como es el caso de pre-U-Net o pre-ResUNet en la tarea de súper-resolución. También hemos visto que una misma red puede obtener un rendimiento bastante competente a la hora de realizar ambas tareas de forma simultánea.

Por otra parte, hemos visto que, en función de la tarea y la arquitectura, la atención por canales puede ayudar o no. También hemos observado que la atención por canales de RCAN no es la única cualidad que la diferencia del resto. RCAN cuenta con numerosos bloques residuales más que ResUNet, y la distribución del resto de conexiones residuales

también es distinta. Así pues, tras quitar la atención por canales de RCAN, los resultados obtenidos con una red u otra pueden variar.

Finalmente, se ha explorado de forma superficial y cualitativa la capacidad de generalización de algunas redes. En concreto, RCAN para segmentación semántica y pre-ResUNet para súper-resolución.

5.1. Trabajo Futuro

A lo largo del desarrollo del proyecto hemos sido testigos de tendencias que pueden dar pie a nuevas exploraciones en esta área.

Hemos podido ver que en la arquitectura RCAN el rendimiento se veía drásticamente ligado al número de canales seleccionados, siendo esta una tendencia ascendente. Sin embargo, el tiempo de ejecución también aumenta acorde al número de canales. Teniendo en cuenta que es una red de gran tamaño, a pesar de usar pocos canales, contamos con tiempos de entrenamiento altos, lo que nos ha limitado la exploración de hiperparámetros.

Por otra parte, hemos podido observar que la influencia de la atención no ha sido muy destacada. Pero al mantener la tasa de reducción del artículo original de RCAN [Zhang et al., 2018a] y reducir el número de canales, hemos estado trabajando con un cuello de botella muy estrecho en el cálculo de la atención. De este modo, surge así la duda de qué tanto influye el tamaño de dicho cuello de botella en la calidad de la atención, y cuán estrecho debería de ser, para obtener mejores resultados.

También sería interesante explorar la influencia de la atención por canales en arquitecturas que no hagan uso de convoluciones únicamente. Como es el caso de algunas de las redes más recientes, que combinan las arquitecturas convolucionales con *transformers* [Esser et al., 2020].

Anexos

A. ANEXO

Gestión del proyecto

A.1. Descripción y objetivos del proyecto

El objetivo para el proyecto es analizar y comparar distintas arquitecturas de redes neuronales para tareas de procesamiento de imágenes. Para ello, será necesario obtener una buena base teórica.

A.2. Planificación del proyecto

A.2.1. Organización del trabajo

Tal y como se presenta en la Estructura de Descomposición del Trabajo (EDT) de la imagen A.1, los paquetes de trabajo son las siguientes:

- **Rama de gestión**
  - **Planificación**

    En este paquete se encuentra la realización de la planificación del proyecto. Se establecerán las tareas y objetivos del proyecto, entregables y fechas límite.
  - **Seguimiento y control**

El paquete de trabajo **Seguimiento y control** agrupará las tareas necesarias para garantizar el adecuado desarrollo del proyecto y, en particular, el cumplimiento de plazos. Para ello, también se fijará la metodología de comunicación y seguimiento del proyecto.

- **Base Teórica**

  Esta sección abarca el conocimiento teórico necesario. Tanto acerca del funcionamiento de las redes, como de las tareas a realizar con estas.

  - **Estado del arte**

    Dentro de la teoría necesaria, se incluye también: buscar, conocer y entender, las arquitecturas que se encuentran dentro del estado del arte para las tareas escogidas.

- **Diseño e implementación**

  - **Especificaciones de diseño**

    En esta subtarea se concretarán aquellos aspectos necesarios para la implementación de las redes.

  - **Implementación**

    El objetivo de esta subtarea será la de implementar las redes.

- **Experimentación y evaluación**

  - **Experimentación**

    Se realizarán los experimentos necesarios, así como la exploración de hiperparámetros.

  - **Evaluación y resultados**

    Se evaluarán los experimentos realizados, y se cuantificará su calidad.

    - **Extraer conclusiones**
      A partir de los resultados obtenidos, se analizarán y extraerán diversas conclusiones.

- **Entregables**

  Esta tarea corresponde en la realización de los dos documentos entregables del Trabajo de Fin de Grado. Trabajos en los que se condensará el conocimiento, experiencia, y resultados obtenidos a lo largo del proyecto.

Figura A.1: Diagrama de la estructura de descomposición del trabajo.

10/03/2021	Fecha límite para la exploración de nuevos experimentos
14/04/2021	Terminar experimentación
10/06/2021	Terminar memoria para la revisión previa por parte de los tutores
20/06/2021	Entrega de la memoria
28/06/2021 - 09/07/2021	Defensa del trabajo

Tabla A.1: Fechas límite del proyecto.

- **Memoria**

  Dentro de este bloque se encuentra la tarea de redacción de la memoria final

- **Presentación**

  El objetivo de esta subtarea será la de realizar y preparar la presentación y defensa del proyecto.

Fechas límite

Además de las fechas impuestas por la universidad para los entregables, se han impuesto fechas límite intermedias a lo largo del proyecto. Todas las fechas están reflejadas en la tabla A.1

Estimación de horas

En la tabla A.2 se recogen tanto las horas estimadas en un principio para cada tarea, como las horas finalmente requeridas.

Tarea	Horas previstas	Horas empleadas
Planificación	5	5
Seguimiento y control	5	5
Adquisición de la base teórica	43	48
Exploración del estado del arte	40	43
Especificaciones de diseño	5	4
Implementación	90	85
Experimentación	40	56
Evaluación	5	5
Extraer conclusiones	2	2
Memoria	60	72
Presentación	5	5
Total	300	330

Tabla A.2: Horas previstas y empleadas para cada tarea.

A.2.2. Comunicación

En esta sección se explicará cómo ha sido la metodología de comunicación utilizada.

Interesados

Se han identificado los siguientes interesados para el proceso de comunicación:

- **Aitor González**. Al ser el autor, es el principal interesado del proyecto. Por un lado el alumno del grado de ingeniería informática depende de este proyecto para obtener los créditos restantes para la obtención del título. Y por otro, concede al interesado un primer contacto con el mundo de la investigación, en un tema que le resulta de gran interés.
- **Ignacio Arganda y Gorka Azkune**. Ambos doctorados e investigando en áreas relacionadas con el tratamiento de imágenes mediante técnicas de aprendizaje profundo. Serán los directores del proyecto, encargados de supervisar el proyecto.

Tras evaluar la disponibilidad de todos los interesados se ha fijado una hora a la semana en la que se realizará una reunión de seguimiento y control. Esta reunión se ha realizado siempre en una misma sala de Google Hangouts. Todos los temas que puedan surgir a lo largo de la semana se han redactado en un documento de seguimiento compartido

entre todos los interesados. En este mismo documento se han fijado los comentarios o resoluciones que hayan podido surgir durante la reunión de esa semana. Para dicha documentación de seguimiento se ha hecho uso de una determinada plantilla común, a lo largo de las semanas.

Para cualquier otra comunicación que pueda ser necesaria a lo largo de la semana, se ha hecho uso de mensajería mediante correo electrónico. Un ejemplo de este tipo de comunicaciones puede ser el avisar en caso de no poder asistir a la reunión semanal. De esta forma, se han establecido dos canales de comunicación: correo y videollamada.

Sistema de información

Todos los contenidos del proyecto se han almacenado de forma digital, distribuidas en distintas plataformas:

- **Ordenador personal**. Se ha almacenado una copia de seguridad de todos los documentos en el ordenador personal de Aitor González.
- **Google Drive**. El código, resultados y documentos de seguimiento se han almacenado en la plataforma de Google Drive. Desde esta misma plataforma, nos facilita el acceso a la herramienta de Google Colab con la que se ha trabajado. Los documentos se han almacenado en una carpeta a la que todos los interesados tienen acceso.
- **Servidor DIPC**. Únicamente se han almacenado los documentos necesarios para el entrenamiento de ciertas redes en el servidor. La comunicación con el servidor se ha realizado principalmente mediante SSH.
- **Overleaf**. La redacción de la memoria se ha realizado en la plataforma de Overleaf. La plataforma también permite el acceso a todos los interesados para las revisiones.

A.2.3. Gestión de riesgos

En esta sección se explicarán los diversos riesgos del proyecto. Para ello se expondrán: los riesgos, su impacto, medidas de mitigación y planes de contingencia.

Ejecución interrumpida

- **Descripción:** Ejecución de entrenamiento detenida. Esto puede suceder por varios motivos. Algunos son más manejables, como errores cometidos en el código. Pero

puede darse el caso de que la ejecución en Colab se vea interrumpida por factores como puedan ser: la detección de inactividad; sobrepasar el límite estático de 12 h como máximo; sobrepasar el límite dinámico por el uso de componentes de aceleración por hardware (en nuestro caso las tarjetas gráficas); o simplemente la pérdida de la conexión con la plataforma, ya sea por recargar, un fallo del servidor o por perder la conexión a internet de forma momentánea.

- **Probabilidad:** Muy alta.
- **Medidas de mitigación:** Todas las ejecuciones realizadas en la plataforma de Google Colab se deben de realizar bajo la constante supervisión. Aquellas ejecuciones que estén cerca o sobrepasen el límite estático de tiempo proporcionado por Google Colab se ejecutarán en el equipo personal o servidor del DIPC. En caso de ser una ejecución muy larga ($> 16\,h$), directamente se ejecutará en el servidor del DIPC que no requiere supervisión constante, y no limita el continuar trabajando. Para aquellas ejecuciones realizadas en el equipo personal o en los servidores del DIPC, será necesaria una supervisión intermitente para detectar lo antes posible la interrupción de la ejecución.

  Antes de realizar una ejecución larga, se reducirán los hiperparámetros (época, número de canales, etc.) con el fin de hacer una ejecución completa previa, en el menor tiempo posible. Y comprobar así, el correcto funcionamiento de todos los elementos.
- **Plan de contingencia:** En caso de que la ejecución se detenga, se retomará tan pronto como sea posible. En caso de no ser una cantidad de horas asumibles para un entrenamiento supervisado, se ejecutará en el servidor del DIPC. Y en caso de no ser una cantidad de horas asumibles bajo ninguna circunstancia, se readaptará el proyecto para lidiar con ello con una posible mención.

Pérdida de datos

- **Descripción:** Debido a algún tipo de problema técnico o humano se pierden los datos almacenados en los servicios principales (Google Drive, Overleaf o el servidor del DIPC).
- **Probabilidad:** Baja.
- **Medidas de mitigación:** De forma periódica, se realizarán copias de seguridad de todos los servicios, y se almacenarán en el equipo personal del autor.

- **Plan de contingencia:** Internamente todos los servicios utilizados cuentan con sus propios planes de contingencia internos. Todos cuentan con diversas estrategias, entre las que se encuentran las copias de seguridad periódicas. En caso de perder los datos, se evaluarán las pérdidas y se tratará de restablecer el trabajo perdido a partir del último punto almacenado. En el supuesto de suponer una pérdida severa, que suponga muchas horas, y no sean asumibles, se readaptará el proyecto para poder continuar.

B. ANEXO

Búsqueda de hiperparámetros

Los tiempos de ejecución pueden variar en función de los hiperparámetros escogidos. El tiempo mostrado en las tablas corresponde a la mejor asignación.

B.1. Glosario

En esta sección se esclarecerán términos presentes en las tablas de la próxima sección.

- ***Spatial dropout*:** Técnica de regularización basada en anular neuronas en la etapa de entrenamiento, con una cierta probabilidad. El valor señalado indica la probabilidad usada. En caso de mostrar una lista, indica las probabilidades por niveles de profundidad.
- **Función de pérdida "mix":** Es una función de pérdida compuesta por otras dos funciones de pérdida, y ponderada por un valor α, al igual que la función de pérdida utilizada en multitarea. Más concretamente, la función de pérdida "mix" trabajará con las funciones de pérdida SSIM y MAE. Para el valor de α se hará uso del valor 0,84.
- ***Scheduler*:** Nos ayudaremos de esta herramienta para adaptar la tasa de aprendizaje utilizada durante el proceso de entrenamiento. Para ello, hemos probado diversas estrategias: OneCycle [Smith, 2018], Reduce LR on plateau (Reduce), o no usar ninguno (*None*).

Infraestructura de cómputo	GPU (Colab)
Mejor mIOU en Test	0,7305
Tiempo de ejecución	<1 h
Tarea	Segmentación

Hiperparámetro	**Espacio de búsqueda**	**Mejor asignación**
Número de épocas	[20, 30, 40, 50]	20
Batch size	[1, 2, 8, 16, 32, 64]	1
Spatial dropout	0.1, 0.1, 0.2, 0.2, 0.3	0.1, 0.1, 0.2, 0.2, 0.3
Función de activación	[ELU, ReLU]	ELU
Módulo de reducción de escala	[Max, Average] Pooling	Max Pooling
Épocas de paciencia	[3,10,15,30]	3
Tasa de aprendizaje	[1e-5, 1e-4, 5e-4, 1e-3]	5e-4
Scheduler	[Reduce, None, oneCycle]	oneCycle
Función de pérdida	SCCE	SCCE
Optimizador	Adam	Adam
Aumento de datos	[Sí, No]	Sí
Arquitectura	[U-Net, ResUNet]	ResUNet
Número de filtros iniciales	[16, 32]	32

Tabla B.1: Espacio de búsqueda y asignaciones de Res/U-Net para segmentación.

- **Épocas de paciencia:** Es el número de épocas que se le permite al modelo proseguir su entrenamiento, sin mejorar el resultado obtenido en evaluación. En el momento que se supere el número de épocas fijado, se detendrá la ejecución, retomando el modelo que haya obtenido un mejor resultado.
- ***Batch size*:** Es el tamaño de lote con en el que se abastece la red.
- ***TConv*:** Haciendo referencia al nombre en inglés, Transposed Convolution, se refiere a la convolución transpuesta.

B.2. Tablas

Infraestructura de cómputo	GPU (Nvidia GTX 1070)
Mejor mIoU en Test	0,7647
Tiempo de ejecución	16 h
Tarea	Segmentación

Hiperparámetro	**Asignación**
Número de épocas	20
Épocas de paciencia	3
Tasa de aprendizaje	5e-4
Scheduler	oneCycle
Función de pérdida	SCCE
Optimizador	Adam
Arquitectura	RCAN
Aumento de datos	Sí
Número de filtros iniciales	32

Tabla B.2: Asignaciones de RCAN para segmentación.

Infraestructura de cómputo	GPU (Colab)
Mejor mIoU en Test	0,7573
Tiempo de ejecución	11 h
Tarea	Segmentación

Hiperparámetro	**Asignación**
Número de épocas	20
Épocas de paciencia	3
Tasa de aprendizaje	5e-4
Scheduler	oneCycle
Función de pérdida	SCCE
Optimizador	Adam
Arquitectura	RCAN (sin atención)
Aumento de datos	Sí
Número de filtros iniciales	32

Tabla B.3: Asignaciones de RCAN sin atención para segmentación.

Infraestructura de cómputo	GPU (Colab)
Mejor mIOU en Test	0,7354
Tiempo de ejecución	1 h
Tarea	Segmentación

Hiperparámetro	**Espacio de búsqueda**	**Mejor asignación**
Número de épocas	20	20
Batch size	1	1
Spatial dropout	0.1, 0.1, 0.2, 0.2, 0.3	0.1, 0.1, 0.2, 0.2, 0.3
Función de activación	[ELU, ReLU]	ReLU
Módulo de reducción de escala	Max Pooling	Max Pooling
Épocas de paciencia	3	3
Tasa de aprendizaje	5e-4	5e-4
Scheduler	oneCycle	oneCycle
Función de pérdida	SCCE	SCCE
Optimizador	Adam	Adam
Aumento de datos	Sí	Sí
Arquitectura	ResCAUNet	ResCAUNet
Número de filtros iniciales	32	32

Tabla B.4: Espacio de búsqueda y asignaciones de ResCAUNet para segmentación.

Infraestructura de cómputo	GPU (Colab)
Mejor SSIM en Test	0,8616
Mejor PSNR en Test	28,2546
Tiempo de ejecución	7 h
Factor de aumento	x4
Tarea	súper-resolución

Hiperparámetro	**Asignación**
Número de épocas	20
Épocas de paciencia	3
Tasa de aprendizaje	1e-3
Scheduler	Reduce
Función de pérdida	MSE
Optimizador	RMSprop
Arquitectura	RCAN
Aumento de datos	Sí
Número de filtros iniciales	32

Tabla B.5: Asignaciones de RCAN para súper-resolución con un factor de aumento de 4.

Infraestructura de cómputo	GPU (Colab)
Mejor SSIM en Test	0,8584
Mejor PSNR en Test	28,1842
Tiempo de ejecución	1 h
Factor de aumento	x4
Tarea	súper-resolución

Hiperparámetro	**Espacio de búsqueda**	**Mejor asignación**
Número de épocas	50	50
Batch size	[64,32,1]	1
Spatial dropout	0,2	0,2
Función de activación	[ReLU, ELU]	ELU
Módulo de reducción de escala	Max Pooling	Max Pooling
Módulo de aumento de escala	[TConv, Subpixel, NN]	Subpixel
Épocas de paciencia	5	5
Tasa de aprendizaje	1e-3	1e-3
Scheduler	Reduce	Reduce
Función de pérdida	[MSE, MAE, mix]	MSE
Optimizador	[RMSprop, Adam]	RMSprop
Aumento de datos	[Sí, No]	Sí
Arquitectura	[(pre, post)ResUNet, preUNet]	preResUNet
Número de filtros iniciales	[16,32]	16

Tabla B.6: Espacio de búsqueda y asignaciones de Res/U-Net para súper-resolución con un factor de aumento de 4.

Infraestructura de cómputo	GPU (Colab)
Mejor mIoU en Test	0,7573
Tiempo de ejecución	11 h
Tarea	Segmentación

Hiperparámetro	**Asignación**
Número de épocas	20
Épocas de paciencia	3
Tasa de aprendizaje	5e-4
Scheduler	oneCycle
Función de pérdida	SCCE
Optimizador	Adam
Arquitectura	RCAN (sin atención)
Aumento de datos	Sí
Número de filtros iniciales	32

Tabla B.7: Asignaciones de RCAN sin atención para súper-resolución con un factor de aumento de 4.

Infraestructura de cómputo	GPU (Colab)
Mejor SSIM en Test	0,8584
Mejor PSNR en Test	28,1717
Tiempo de ejecución	3 h
Factor de aumento	x4
Tarea	súper-resolución

Hiperparámetro	**Asignación**
Número de épocas	50
Batch size	1
Spatial dropout	0,2
Función de activación	ELU
Módulo de reducción de escala	Max Pooling
Módulo de aumento de escala	Subpixel
Épocas de paciencia	5
Tasa de aprendizaje	1e-3
Scheduler	Reduce
Función de pérdida	MSE
Optimizador	RMSprop
Aumento de datos	Sí
Arquitectura	pre-ResCAUNet
Número de filtros iniciales	32

Tabla B.8: Asignaciones de ResCAUNet para súper-resolución con un factor de aumento de 4.

Infraestructura de cómputo	GPU (Colab)
Mejor SSIM en Test	0,8864
Mejor PSNR en Test	29,4274
Tiempo de ejecución	8 h
Factor de aumento	x2
Tarea	súper-resolución

Hiperparámetro	**Asignación**
Número de épocas	20
Épocas de paciencia	3
Tasa de aprendizaje	1e-3
Scheduler	Reduce
Función de pérdida	MSE
Optimizador	RMSprop
Arquitectura	RCAN
Aumento de datos	Sí
Número de filtros iniciales	32

Tabla B.9: Asignaciones de RCAN para súper-resolución con un factor de aumento de 2.

Infraestructura de cómputo	GPU (Colab)
Mejor SSIM en Test	0,8852
Mejor PSNR en Test	29,3466
Tiempo de ejecución	1 h
Factor de aumento	x2
Tarea	súper-resolución

Hiperparámetro	**Espacio de búsqueda**	**Mejor asignación**
Número de épocas	50	50
Batch size	[64,1]	1
Spatial dropout	0,2	0,2
Función de activación	ELU	ELU
Módulo de reducción de escala	Max Pooling	Max Pooling
Módulo de aumento de escala	Subpixel	Subpixel
Épocas de paciencia	5	5
Tasa de aprendizaje	1e-3	1e-3
Scheduler	Reduce	Reduce
Función de pérdida	[MSE, MAE, mix]	MSE
Optimizador	RMSprop	RMSprop
Aumento de datos	[Sí, No]	Sí
Arquitectura	preResUNet	preResUNet
Número de filtros iniciales	[16,32]	32

Tabla B.10: Espacio de búsqueda y asignaciones de ResUNet para súper-resolución con un factor de aumento de 2.

Infraestructura de cómputo	GPU (Colab)
Mejor mIOU en Test	0,6940
Mejor SSIM en Test	0,8501
Mejor PSNR en Test	27,8044
Factor de aumento	x4
Tiempo de ejecución	2 h
Tarea	Segmentación

Hiperparámetro	**Espacio de búsqueda**	**Mejor asignación**
Número de épocas	[20, 50, 75, 100]	75
Batch size	1	1
Spatial dropout	0.2	0.2
Función de activación	ELU	ELU
Módulo de reducción de escala	Max Pooling	Max Pooling
Épocas de paciencia	[3, 5, 10, 15, 20]	10
Tasa de aprendizaje	1e-3	1e-3
Scheduler	Reduce	Reduce
Función de pérdida	SCCE + MSE	SCCE + MSE
Optimizador	RMSprop	RMSprop
Aumento de datos	[Sí, No]	Sí
Arquitectura	ResUNet	ResUNet
Número de filtros iniciales	[16, 32]	32

Tabla B.11: Espacio de búsqueda y asignaciones de Res/U-Net para multitarea.

Infraestructura de cómputo	GPU (Colab)
Mejor mIOU en Test	0,6258
Mejor SSIM en Test	0,8410
Mejor PSNR en Test	27,2377
Factor de aumento	x4
Tiempo de ejecución	8 h
Tarea	Segmentación

Hiperparámetro	**Espacio de búsqueda**
Número de épocas	20
Batch size	1
Épocas de paciencia	5
Tasa de aprendizaje	1e-3
Scheduler	Reduce
Función de pérdida	SCCE + MSE
Optimizador	RMSprop
Aumento de datos	Sí
Arquitectura	RCAN
Número de filtros iniciales	32

Tabla B.12: Asignaciones de RCAN para multitarea.

Infraestructura de cómputo	GPU (Colab)
Mejor mIoU en Test	0,6052
Mejor SSIM en Test	0,8349
Mejor PSNR en Test	26,9250
Tiempo de ejecución	3 h
Factor de aumento	x4
Tarea	SR y segmentación

Hiperparámetro	**Asignación**
Número de épocas	20
Batch size	1
Épocas de paciencia	3
Tasa de aprendizaje	1e-3
Scheduler	Reduce
Función de pérdida	SCCE + MSE
Optimizador	RMSprop
Arquitectura	RCAN (sin atención)
Aumento de datos	Sí
Número de filtros iniciales	32

Tabla B.13: Asignaciones de RCAN sin atención para multitarea.

Infraestructura de cómputo	GPU (Colab)
Mejor mIoU en Test	0,6646
Mejor SSIM en Test	0,8477
Mejor PSNR en Test	27,6478
Tiempo de ejecución	1 h
Factor de aumento	x4
Tarea	SR y segmentación

Hiperparámetro	**Espacio de búsqueda**	**Mejor asignación**
Número de épocas	[50,100]	100
Batch size	1	1
Spatial dropout	0,2	0,2
Función de activación	ELU	ELU
Módulo de reducción de escala	Max Pooling	Max Pooling
Módulo de aumento de escala	Subpixel	Subpixel
Épocas de paciencia	10	10
Tasa de aprendizaje	1e-3	1e-3
Scheduler	Reduce	Reduce
Función de pérdida	SCCE + MSE	SCCE + MSE
Optimizador	RMSprop	RMSprop
Aumento de datos	Sí	Sí
Arquitectura	pre-ResCAUNet	pre-ResCAUNet
Número de filtros iniciales	32	32

Tabla B.14: Espacio de búsqueda y asignaciones de ResCAUNet para multitarea.

Bibliografía

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

[Alonso et al., 2020] Alonso, I., Riazuelo, L., and Murillo, A. C. (2020). Mininet: An efficient semantic segmentation convnet for real-time robotic applications. *IEEE Transactions on Robotics*, 36(4):1340–1347.

[Arnab et al., 2018] Arnab, A., Zheng, S., Jayasumana, S., Romera-Paredes, B., Larsson, M., Kirillov, A., Savchynskyy, B., Rother, C., Kahl, F., and Torr, P. (2018). Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction. *IEEE Signal Processing Magazine*, 35:37–52.

[Chen et al., 2016] Chen, H., Qi, X., Yu, L., and Heng, P.-A. (2016). Dcan: Deep contour-aware networks for accurate gland segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2487–2496.

[Chollet et al., 2015] Chollet, F. et al. (2015). Keras.

[Drozdzal et al., 2016] Drozdzal, M., Vorontsov, E., Chartrand, G., Kadoury, S., and Pal, C. (2016). The importance of skip connections in biomedical image segmentation. In Carneiro, G., Mateus, D., Peter, L., Bradley, A., Tavares, J. M. R. S., Belagiannis, V., Papa, J. P., Nascimento, J. C., Loog, M., Lu, Z., Cardoso, J. S., and Cornebise, J.,

editors, *Deep Learning and Data Labeling for Medical Applications*, pages 179–187, Cham. Springer International Publishing.

[Erhan et al., 2009] Erhan, D., Bengio, Y., Courville, A., and Vincent, P. (2009). Visualizing higher-layer features of a deep network. *Technical Report, Univeristé de Montréal*.

[Esser et al., 2020] Esser, P., Rombach, R., and Ommer, B. (2020). Taming transformers for high-resolution image synthesis. *CoRR*, abs/2012.09841.

[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*.

[Harris et al., 2020] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585:357–362.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[Isensee et al., 2020] Isensee, F., Jaeger, P. F., Full, P. M., Vollmuth, P., and Maier-Hein, K. H. (2020). nnu-net for brain tumor segmentation.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

[Milioto et al., 2018] Milioto, A., Lottes, P., and Stachniss, C. (2018). Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2229–2235.

[Parkhi et al., 2012] Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V. (2012). Oxford-iiit pet dataset. https://www.robots.ox.ac.uk/~vgg/data/pets/.

[Qin et al., 2018] Qin, Z., Yu, F., Liu, C., and Chen, X. (2018). How convolutional neural network see the world - A survey of convolutional neural network visualization methods. *CoRR*, abs/1804.11191.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

[Sagar and Soundrapandiyan, 2020] Sagar, A. and Soundrapandiyan, R. (2020). Semantic segmentation with multi scale spatial attention for self driving cars.

[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.

[Smith, 2018] Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay.

[Sultana et al., 2020] Sultana, F., Sufian, A., and Dutta, P. (2020). Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems*, 201-202:106062.

[Thoma, 2016] Thoma, M. (2016). A survey of semantic segmentation.

[Tim-Oliver et al., 2020] Tim-Oliver, B., Mangal, P., Alexander, K., and Florian, J. (2020). DenoiSeg: Joint Denoising and Segmentation. *arXiv preprint, arXiv:2005.02987v2*.

[van Ouwerkerk, 2006] van Ouwerkerk, J. (2006). Image super-resolution survey. *Image and Vision Computing*, 24(10):1039–1052.

[Van Rossum and Drake, 2009] Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

[Wang et al., 2020a] Wang, L., Li, D., Zhu, Y., Tian, L., and Shan, Y. (2020a). Dual super-resolution learning for semantic segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3773–3782.

[Wang et al., 2004] Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

[Wang et al., 2020b] Wang, Z., Chen, J., and Hoi, S. C. (2020b). Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

[Wu et al., 2019] Wu, N., Phang, J., Park, J., Shen, Y., Huang, Z., Zorin, M., Jastrzębski, S., Févry, T., Katsnelson, J., Kim, E., Wolfson, S., Parikh, U., Gaddam, S., Lin, L. L. Y., Ho, K., Weinstein, J. D., Reig, B., Gao, Y., Toth, H., Pysarenko, K., Lewin, A., Lee, J., Airola, K., Mema, E., Chung, S., Hwang, E., Samreen, N., Kim, S. G., Heacock, L., Moy, L., Cho, K., and Geras, K. J. (2019). Deep neural networks improve radiologists' performance in breast cancer screening.

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

[Zhang et al., 2018b] Zhang, Z., Liu, Q., and Wang, Y. (2018b). Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753.