UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

FACULTAD DE
INGENIERÍA
UDELAR

Redes Neuronales Generativas Adversarias para Superresolución de Modelos Digitales de Elevación

Juan Pablo Lorenzo
Facundo Locatelli
Franco Filipponi

Proyecto de grado presentado a la Facultad de Ingeniería de la Universidad de la República en cumplimiento parcial de los requerimientos para la obtención del título de Ingenieria en Computación.

Tutor

Sergio Nesmachnow

Tribunal

Eduardo Fernández
Pablo Rebufello
Rodolfo Méndez

Montevideo, Uruguay
30 de mayo de 2025

Instituto de Computación - Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay, mayo de 2025

Resumen

La superresolución propone mejorar la calidad de imágenes digitales al aumentar su resolución espacial y tiene aplicaciones en múltiples disciplinas como la medicina y la geografía (Greenspan, 2008; Wang et al., 2024). Este proyecto de grado aborda el problema de superresolución de Modelos Digitales de Elevación (DEM por sus siglas en inglés, digital elevation models), representados como imágenes de un solo canal, para incrementar su cantidad de puntos por unidad de superficie. El problema es especialmente relevante ya que la calidad de los DEMs suele estar limitada por la metodología utilizada para generarlos y por los altos costos asociados, que aumentan notablemente al incrementar la resolución (Ruiz et al., 2023).

El enfoque adoptado utiliza redes neuronales generativas adversarias (GANs por sus siglas en inglés, generative adversarial networks) como método alternativo que tiene una alta eficiencia computacional y que alcanza una buena calidad de resultados (Goodfellow et al., 2014). Las GANs aplican aprendizaje profundo sobre grandes volúmenes de datos para aprender características complejas y reconstruir detalles finos que los métodos tradicionales (como los métodos basados en interpolación) no pueden captar. Se evalúan diferentes arquitecturas de GANs (SRGAN (Ledig et al., 2017), ESRGAN (Wang et al., 2019), DSRGAN (Demiray et al., 2021) y una variante de SRGAN (Zhang y Yu, 2022)) y se adaptan para mejorar la resolución espacial de los DEMs.

Los resultados demostraron la aplicabilidad y eficacia de las GANs para la superresolución de DEMs, con mejoras significativas en métricas cuantitativas y cualitativas sobre métodos tradicionales utilizados para el problema. El modelo ESRGAN redujo en 66.4 % el error absoluto medio (MAE por sus siglas en inglés, mean absolute error) comparado con el método tradicional de interpolación bilineal.

Palabras clave: Redes neuronales generativas adversarias, Modelos digitales de elevación, Superresolución

Índice general

Capítulo 1

Introducción

La superresolución, entendida como el proceso de mejorar la calidad de imágenes digitales mediante el aumento de su resolución, es una técnica que ha adquirido gran importancia en múltiples disciplinas, incluyendo la medicina y la geografía (Greenspan, 2008; Wang et al., 2024). Existen diversos métodos de superresolución en imágenes que se clasifican en métodos tradicionales y métodos basados en aprendizaje. Los métodos tradicionales incluyen técnicas como la interpolación bilineal y la interpolación bicúbica, que estiman nuevos píxeles a partir de valores vecinos. En contraste, los métodos basados en aprendizaje utilizan grandes volúmenes de datos para aprender relaciones complejas entre las imágenes de baja y alta resolución. Entre los métodos basados en aprendizaje se destacan aquellos que emplean redes neuronales.

En el campo de la cartografía, la superresolución presenta un potencial considerable cuando se aplica sobre Modelos Digitales de Elevación (DEM por sus siglas en inglés, digital elevation models). Los DEMs son representaciones de la topografía terrestre en forma de imágenes de un solo canal que describen la altitud de los puntos en una superficie. En los DEMs se pueden aplicar los mismos métodos de superresolución que se utilizan en imágenes, realizando las modificaciones pertinentes.

El enfoque adoptado en este proyecto de grado utiliza redes neuronales generativas adversarias (GAN por sus siglas en inglés, generative adversarial networks) como método alternativo para abordar el problema de la superresolución de DEMs. Este tipo de red tiene una alta eficiencia computacional y alcanza una buena calidad de resultados (Goodfellow et al., 2014). Las GANs aplican aprendizaje profundo sobre grandes volúmenes de datos para aprender características complejas y reconstruir detalles finos que los métodos tradicionales no pueden captar.

La resolución y la calidad de los DEMs están limitadas por la metodología utilizada para generarlos y por los altos costos asociados, que aumentan notablemente al incrementar la resolución (Ruiz et al., 2023). Estas limitaciones motivan la exploración de métodos alternativos para generar DEMs de alta resolución. Como método alternativo, los algoritmos de interpolación tradicionales permiten obtener resultados aceptables con gran rapidez. Sin embargo, la calidad de estos resultados es inferior en comparación con la alcanzada por los métodos basados en GANs.

Se encuentran disponibles por el IDEUY (Infraestructura de Datos Espaciales de Uruguay) DEMs de todo el territorio uruguayo con una resolución de 2.5 metros. También, existen DEMs de 6 regiones particulares del territorio uruguayo en donde la resolución es de 0.5 metros. En este proyecto de grado se busca desarrollar un modelo basado en

GAN que permita incrementar la resolución de los DEMs del territorio uruguayo de 2.5 a 0.5 metros y que supere a los resultados alcanzados por los métodos tradicionales de superresolución.

Se realizó un relevamiento de trabajos relacionados al problema de superresolución en DEMs con GANs y se definieron las arquitecturas con las que resolver este problema. Las arquitecturas seleccionadas para evaluar fueron SRGAN (Ledig et al., 2017), ESRGAN (Wang et al., 2019), DSRGAN (Demiray et al., 2021) y una variante de la arquitectura SRGAN ya entrenada en otro conjunto de datos (Zhang y Yu, 2022). Cada arquitectura fue adaptada al problema de superresolución en DEMs y los resultados obtenidos al evaluar cada una se compararon con los resultados obtenidos al evaluar los métodos de interpolación bilineal y bicúbica.

Para los modelos de cada arquitectura se realizó una búsqueda de hiperparámetros con el objetivo de encontrar las configuraciones óptimas para cada modelo. Seguido de la búsqueda de hiperparámetros se realizaron entrenamientos en profundidad con los mejores modelos de cada arquitectura, de manera de explotar al máximo las capacidades de cada modelo. Dentro de cada arquitectura, se evaluaron cuatro variantes que surgieron de modificar los componentes de la función de pérdida y los datos utilizados al calcularla. También se evaluaron dos técnicas de aumento de datos.

Una de las variantes evaluadas en las arquitecturas fue la denominada ESRGAN-B personalizada. Esta variante presentó resultados significativamente superiores y fue la base del mejor modelo obtenido en este proyecto de grado. La ESRGAN-B personalizada, en comparación con la interpolación bilineal, presentó una reducción en el error absoluto medio (MAE por sus siglas en inglés, mean absolute error) de 66.4 %, en la raíz del error cuadrático medio (RMSE por sus siglas en inglés, root mean square error) de 48.0 % y en el error lineal de percentil 90 (LE90 por sus siglas en inglés, linear error 90) de 73.1 % en el conjunto de datos de evaluación.

Los resultados obtenidos en este proyecto de grado evidencian la eficacia de las GANs para la superresolución de DEMs. En particular, la arquitectura ESRGAN-B personalizada se destacó como la solución con mejores resultados sobre el conjunto de evaluación. La superioridad de ESRGAN-B sobre la interpolación bilineal validó el uso de GANs como una alternativa viable para mejorar la resolución espacial de los DEMs sin incurrir en los altos costos asociados a métodos convencionales de adquisición de datos de alta resolución. Además, las métricas utilizadas para la evaluación corroboraron la capacidad de los modelos basados en GANs para reconstruir detalles finos y mejorar sustancialmente la calidad de los DEMs.

Además del capítulo introductorio, el informe se organiza en cinco capítulos adicionales. En el capítulo 2 se desarrolla el marco teórico, abarcando los conceptos fundamentales de superresolución, DEMs y redes neuronales, con énfasis en las GANs. En el capítulo 3 se presentan los trabajos relacionados con el tema principal de este proyecto de grado. Además, en el capítulo 3 se incluyen trabajos relacionados con la superresolución de DEMs y trabajos sobre la aplicación de redes neuronales al campo de la superresolución. En el capítulo 4 se detalla la metodología adoptada incluyendo la selección de arquitecturas, el procesamiento de datos y los detalles del proceso de entrenamiento y evaluación de los modelos. En el capítulo 5 se exponen los experimentos realizados y los resultados obtenidos, con un análisis comparativo entre las distintas arquitecturas evaluadas. Finalmente, en el capítulo 6 se presentan las conclusiones del proyecto de grado y se plantea el trabajo futuro.

Capítulo 2

Marco teórico

Esta sección presenta los conceptos teóricos fundamentales para el desarrollo del proyecto de grado. Se abordan temas como la superresolución y sus distintos tipos, los algoritmos de interpolación y las redes neuronales, incluyendo sus principios básicos como el proceso de entrenamiento, las funciones de pérdida y otros aspectos clave. Además se presenta la definición de DEM y su uso en el marco de esta investigación. En relación con las redes neuronales, se describen sus principales tipos, con énfasis en aquellas más relevantes para este trabajo: las redes neuronales convolucionales (CNN por sus siglas en inglés, convolutional neural networks) y GANs.

2.1. Superresolución

La disponibilidad de imágenes digitales en alta resolución es crucial para una amplia variedad de aplicaciones. Más allá de la percepción humana, las imágenes de mayor resolución ofrecen un nivel superior de detalle y, por lo tanto, almacenan una mayor cantidad de información que puede ser aprovechada por diversas áreas del conocimiento, como la medicina y la geografía (Greenspan, 2008; Wang et al., 2024).

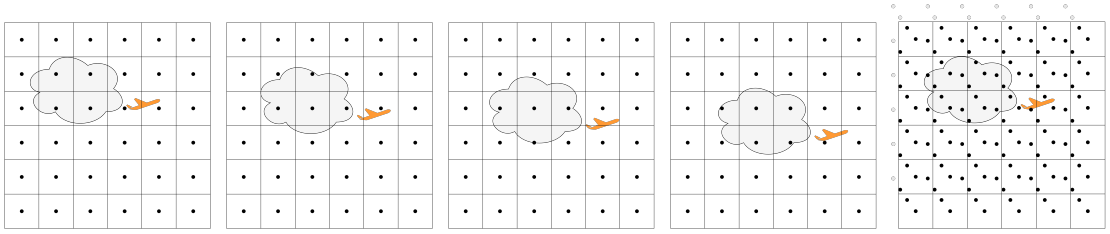
El problema de superresolución para DEMs puede abordarse de manera análoga al problema de superresolución con imágenes, debido a que los DEMs se pueden interpretar como grillas de celdas equivalentes a imágenes de un solo canal. Por lo tanto, el problema de la superresolución para los DEMs se reduce a estimar valores intermedios en la grilla de celdas original y así aumentar la cantidad de puntos de elevación de los DEMs por unidad de superficie.

Para conseguir el objetivo de aumentar la resolución de los DEMs existen diferentes algoritmos que pueden adaptarse del problema análogo de superresolución en imágenes. Los algoritmos se dividen en dos grandes familias según los conjuntos de datos disponibles: los algoritmos para superresolución multi-imagen (MISR por sus siglas en inglés, multi-image super-resolution) y los algoritmos de superresolución mono-imagen (SISR por sus siglas en inglés, single-image super-resolution). A continuación se describen brevemente ambas familias de técnicas de superresolución.

2.1.1. Superresolución multi-imagen

Para la superresolución multi-imagen se plantea obtener una imagen de alta resolución mediante la combinación y alineación de imágenes de menos resolución con pequeños desplazamientos.

En la Figura 2.1 se muestra un esquema de cómo funciona el algoritmo de alineamiento sub-pixel (Wronski et al., 2019). Una versión optimizada de este algoritmo se utiliza en dispositivos móviles para aprovechar el muestreo no uniforme causado por el movimiento de la cámara y reconstruir altas frecuencias disminuyendo el ruido (Lafenetre et al., 2023).



(a) Imagen base (b) Imagen uno (c) Imagen dos (d) Imagen tres (e) Alineación final

Figura 2.1: Alineamiento sub-pixel para superresolución multi-imagen.

2.1.2. Superresolución mono-imagen

Los algoritmos para resolución mono-imagen mejoran la calidad de una imagen de baja resolución utilizando únicamente la información contenida en la imagen. Estos algoritmos se agrupan en dos grandes categorías: los tradicionales (por ejemplo, los algoritmos basados en interpolación) y los basados en aprendizaje.

Algoritmos basados en interpolación

Los algoritmos de superresolución basados en interpolación son métodos clásicos que generan una imagen de mayor resolución a partir de otra imagen de menor resolución mediante funciones matemáticas diseñadas para estimar los valores de los píxeles intermedios. Estos métodos se destacan por su simplicidad y rapidez de ejecución, aunque suelen ser menos precisos en comparación con los enfoques modernos basados en aprendizaje profundo. Entre los métodos de interpolación más comunes se encuentran la interpolación de vecino más cercano, la interpolación bilineal y la interpolación bicúbica (Han, 2013).

La interpolación de vecino más cercano es el método de interpolación más simple. Consiste en asignar a cada píxel de la imagen de mayor resolución el valor del píxel más cercano en la imagen de menor resolución. Aunque es rápido, la interpolación de vecino más cercano tiende a generar imágenes con bordes escalonados y un nivel de detalle limitado.

En el algoritmo de interpolación bilineal el valor de cada píxel nuevo se calcula como un promedio ponderado de los cuatro píxeles más cercanos de la imagen original. Como resultado, se obtiene una transición más suave entre píxeles, lo que mejora la calidad visual de la imagen en comparación con el método del vecino más cercano, que genera bordes bruscos.

La interpolación bicúbica utiliza los 16 píxeles más cercanos para calcular el valor de cada píxel de la imagen de alta resolución. Al considerar un mayor número de píxeles se logran transiciones aún más suaves y se preservan mejor los detalles finos de la imagen. La interpolación bicúbica es uno de los métodos de interpolación más utilizados para tareas de superresolución debido a su equilibrio entre calidad y velocidad de procesamiento.

La Figura 2.2 muestra los resultados obtenidos al aplicar los métodos de interpolación de vecino más cercano, bilineal y bicúbica a un DEM. El DEM original (HR) se redujo mediante un escalado con un factor de 0.2 para obtener un DEM en baja resolución. Este DEM en baja resolución se utilizó como entrada para los tres algoritmos de interpolación que devolvieron un DEM en alta resolución con un aumento en un factor de 5.

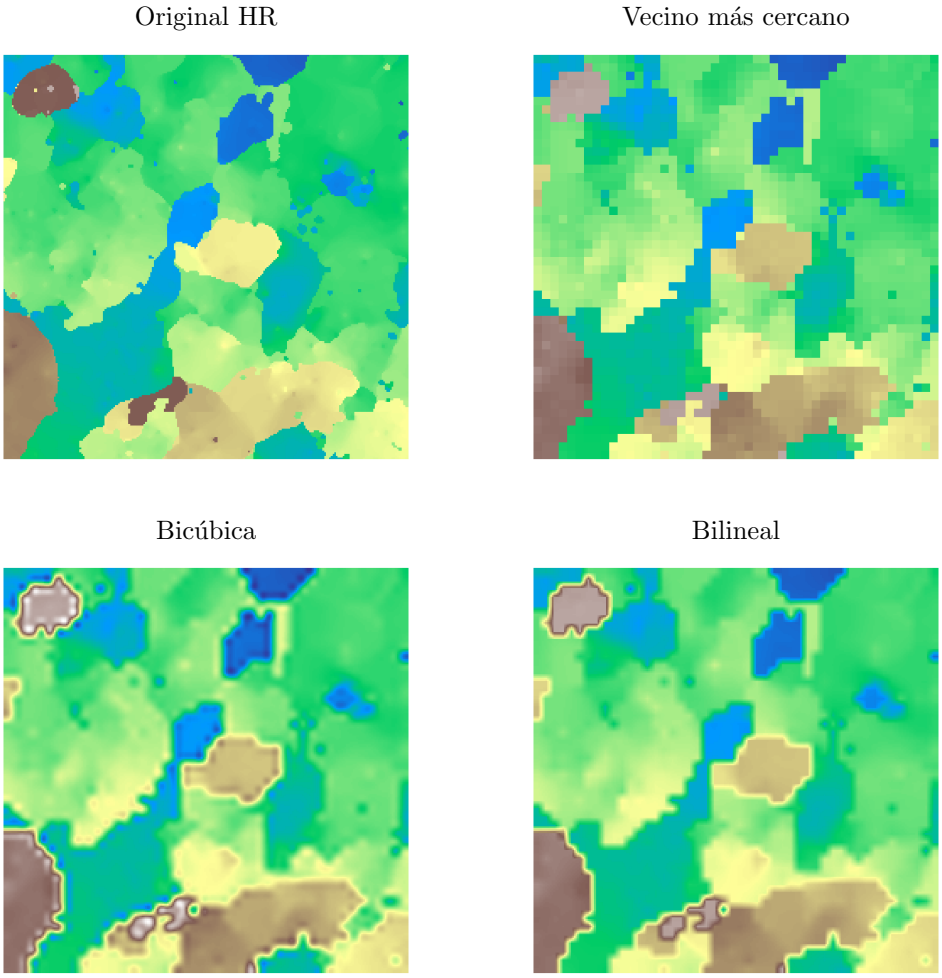


Figura 2.2: Comparación de súper resolución con métodos de interpolación.

Aunque los métodos de interpolación permiten mejorar la resolución de imágenes de manera rápida y sencilla, presentan limitaciones importantes en su capacidad para reconstruir detalles finos y características de alta frecuencia. Estas limitaciones se deben a que los algoritmos basados en interpolación no generan nueva información en la imagen, sino que estiman los valores de los píxeles intermedios a partir de la información existente. Por esta razón, los enfoques basados en interpolación suelen ser menos eficaces que

los métodos basados en aprendizaje profundo en tareas que requieren la recuperación de detalles de textura complejos o una mejora significativa en la calidad de la imagen superresuelta (Zhang y Yu, 2022).

Algoritmos basados en aprendizaje

Los algoritmos basados en aprendizaje para la superresolución de imágenes emplean técnicas de inteligencia artificial, en particular redes neuronales profundas, para generar imágenes de alta resolución a partir de imágenes de baja resolución. A diferencia de los métodos de interpolación tradicionales, que dependen de funciones matemáticas predefinidas, los enfoques basados en aprendizaje profundo son capaces de aprender patrones a partir de grandes volúmenes de datos. En la Sección 2.3 se detalla con mayor profundidad el funcionamiento de las redes neuronales.

Las redes neuronales suelen entrenarse utilizando conjuntos de datos compuestos por pares de imágenes de baja y alta resolución. Durante el proceso de entrenamiento, las redes neuronales aprenden a relacionar las características presentes en las imágenes de baja resolución con sus correspondientes versiones en alta resolución. Las relaciones aprendidas permiten a las redes neuronales lograr una reconstrucción de detalles significativamente superior a la ofrecida por los métodos tradicionales.

Si bien existe una amplia variedad de modelos de redes neuronales, existen tres que han demostrado tener mejores resultados en el procesamiento de imágenes en general y en la tarea concreta de superresolución:

- CNNs: Las CNNs son altamente efectivas en tareas de superresolución debido a su capacidad para capturar características espaciales en las imágenes. Este tipo de redes supera significativamente a los métodos de interpolación, permitiendo recuperar un mayor nivel de detalle y generar imágenes con un aspecto menos borroso (Dong et al., 2014; Yamanaka et al., 2017). En la sección 2.3.3 se explica con mayor detalle el funcionamiento de las redes neuronales convolucionales.

- GANs: Las GANs son un tipo de arquitectura que combina dos redes neuronales: un generador y un discriminador. En el contexto de la superresolución, el generador crea imágenes de alta resolución a partir de sus contrapartes de baja resolución, mientras que el discriminador intenta distinguir entre imágenes reales y generadas. En la sección 2.3.3 se explica con mayor detalle el funcionamiento de las GANs.

- Transformadores (transformers): los modelos basados en transformers han demostrado un gran potencial en la superresolución de imágenes. Estos modelos utilizan mecanismos de atención que les permiten enfocarse en diferentes regiones de la imagen para lograr una reconstrucción más precisa.

Los métodos basados en aprendizaje profundo han demostrado superar significativamente a los métodos de interpolación en términos de calidad visual y precisión en la reconstrucción de detalles para las imágenes generadas. Sin embargo, estos métodos también presentan desafíos, como la necesidad de grandes volúmenes de datos de entrenamiento y tiempos de procesamiento prolongados. A pesar de estos desafíos, la capacidad de los algoritmos basados en aprendizaje para capturar patrones complejos y su flexibilidad para adaptarse a distintos tipos de imágenes los han convertido en una herramienta esencial en el campo de la superresolución (Zhang y Yu, 2022).

2.2. Modelos de elevación digital

Según lo propuesto por Guth et al. (2021) se definió para este proyecto de grado qué es un DEM y cómo debe manipularse. Los DEMs son modelos de representación de datos utilizados en una amplia variedad de áreas de estudio. Un DEM se representa como una grilla de celdas (Burrough et al., 2015) en donde cada celda contiene información específica que representa una abstracción de la superficie terrestre. En la Figura 2.3 se muestra gráficamente la relación entre la representación e interpretación de un DEM como grilla de celdas.

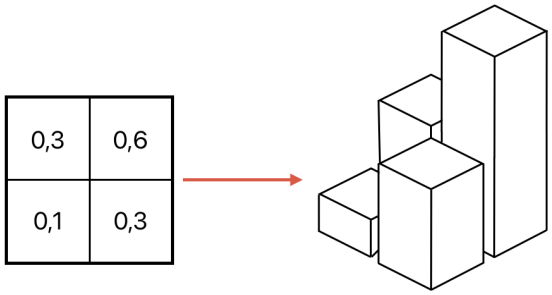


Figura 2.3: DEM representado como una matriz e interpretado como valores de elevación de terreno.

Matemáticamente, los DEMs pueden representarse como una matriz que contiene datos de elevación. A partir de la información de elevación de cada DEM, se pueden obtener otras características del terreno como la pendiente dada por el cambio de elevación entre celdas, o como la orientación, que representa la dirección de las pendientes en el terreno (Burrough et al., 2015).

Para la correcta interpretación y comparación de DEMs que representan la misma parcela de tierra, es fundamental que coincidan el sistema de coordenadas, los datos horizontales y verticales, así como las representaciones en celdas entre los modelos que se están comparando. En este sentido, los metadatos asociados a un DEM desempeñan un papel crucial ya que incluyen el código EPSG, los bordes del DEM en un sistema de referencia como WGS84 y parámetros como el ancho y la altura que permiten definir el rectángulo geográfico que el DEM representa (Guth et al., 2021).

2.2.1. Tipos de DEMs

Existen diferentes tipos de DEMs, entre los cuales se destacan los Modelos de Superficie Digital (DSM por sus siglas en inglés, digital surface models) y los Modelos de Terreno Digital (DTM por sus siglas en inglés, digital terrain models). Los DSM capturan la elevación de la superficie, incluyendo vegetación y estructuras artificiales y los DTM representan exclusivamente la elevación del terreno. La Figura 2.4 muestra las diferencias entre ambos.

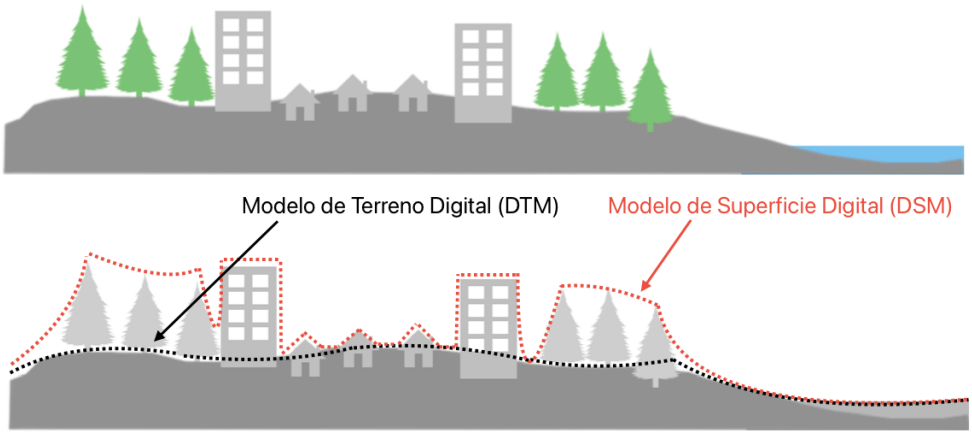


Figura 2.4: Representación de las diferencias entre DSM y DTM.

2.2.2. Almacenamiento y formato de archivo para DEMs

Generalmente, los DEMs se almacenan en formato ráster (Burrough et al., 2015). De manera simple, un dato se considera almacenado en forma ráster cuando está organizado en una matriz donde cada celda contiene un valor que representa información específica, generalmente alguna magnitud física, como la altura en el caso particular de un DEM. Este formato de almacenamiento no solo es intuitivo y sencillo, sino que también facilita la realización de ciertas operaciones, como las de recorte.

Existen diversos formatos de archivo capaces de almacenar un DEM en forma de ráster, incluyendo formatos de imagen estándar como JPG, PNG o TIFF, entre otros. En particular, el formato GeoTIFF es el más utilizado. El formato GeoTIFF fue presentado en 1995 con el objetivo de establecer un estándar para la comunicación de datos geográficos ráster sin depender de ninguna compañía propietaria. Desde entonces se ha convertido en el estándar ampliamente aceptado en el ámbito de los Sistemas de Información Geográfica (GIS por sus siglas en inglés, geographical information system) (Ritter y Ruth, 1995).

Un GIS es una herramienta informática diseñada para capturar, almacenar, analizar, gestionar y presentar datos espaciales o geográficos. Esta herramienta es ampliamente utilizada en diversas disciplinas para trabajar con información relacionada con la ubicación geográfica y permite una comprensión visual y analítica de los datos.

El formato GeoTIFF permite incorporar información de georreferencia directamente en los archivos con formato TIFF, lo que facilita su interpretación mediante software especializado como QGIS, que se describe en la sección siguiente.

2.2.3. QGIS

QGIS (QGIS Development Team, 2025) es un software de código abierto que ofrece una amplia variedad de funcionalidades para el procesamiento y análisis de datos geoespaciales. Entre sus características principales se encuentran la capacidad de manejar distintos formatos de datos geográficos, la realización de análisis espaciales avanzados y la creación de mapas de alta calidad para presentar resultados de manera clara y comprensible. En la Figura 2.5 se muestra una captura de pantalla de la herramienta QGIS.

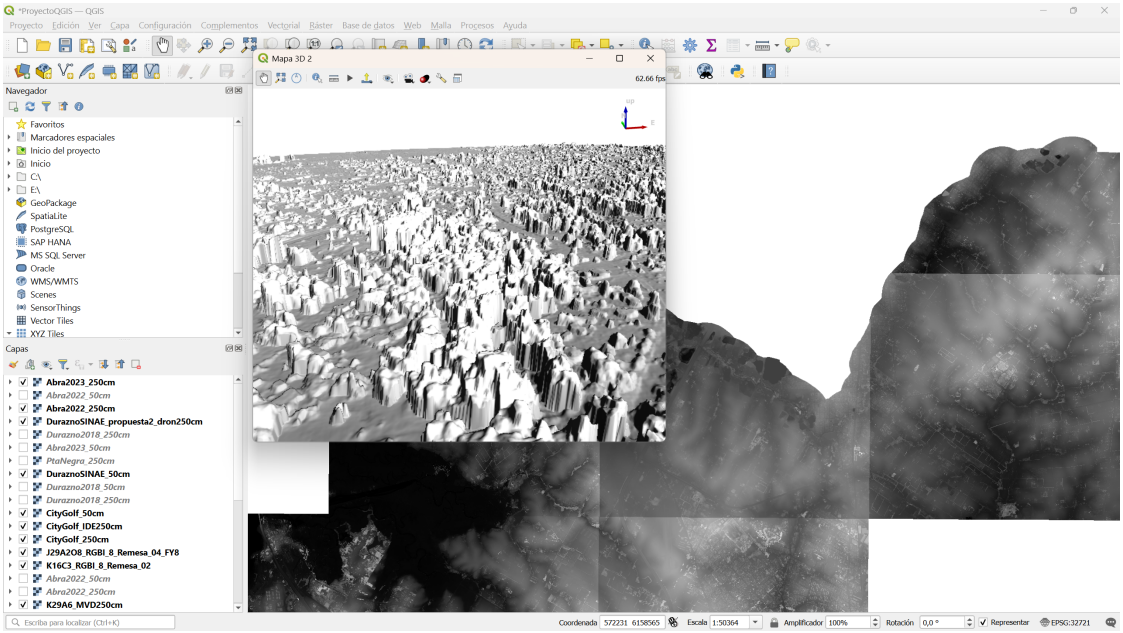


Figura 2.5: Captura de pantalla de la herramienta QGIS para Windows.

Como se muestra en la Figura 2.5, QGIS permite visualizar los DEMs en formato plano o mediante un visor 3D que representa de forma más intuitiva la información de alturas. Además, la Figura 2.5 muestra la variedad de opciones y herramientas que este software ofrece a sus usuarios.

2.3. Redes neuronales

Las redes neuronales son modelos computacionales diseñados para reconocer patrones y aprender a partir de datos mediante un proceso de entrenamiento. Este proceso les permite generalizar a partir de ejemplos específicos, lo que convierte a las redes neuronales en modelos adecuados para resolver tareas complejas y diversas, como clasificación de imágenes, reconocimiento de voz, traducción automática y generación de texto.

2.3.1. La neurona

Las redes neuronales se basan en el concepto de neurona artificial. En su formulación más primitiva, la lógica de la neurona artificial se inspira en el funcionamiento de las neuronas del cerebro humano. En 1943, McCulloch y Pitts (1943) sentaron las bases matemáticas y lógicas para el desarrollo posterior de las redes neuronales. Luego Rosenblatt (1958) creó lo que se considera como el primer modelo computacional capaz de aprender, el perceptrón.

La estructura del perceptrón está definida por la Ecuación 2.1 y se representa en la Figura 2.6.

$$y = \sigma(\sum_{i=1}^{n} w_i x_i + w_0) \tag{2.1}$$

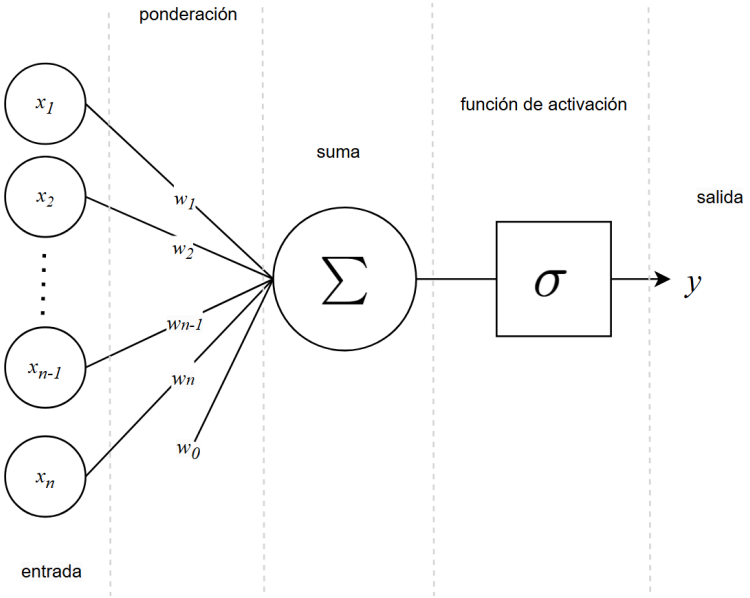


Figura 2.6: Representación del perceptrón

El perceptrón posee cinco componentes principales:

- Entradas (x_i): el perceptrón recibe múltiples entradas, cada una representada por un valor numérico.
- Pesos (w_i): cada entrada x_i está asociada con un peso w_i que determina la importancia de esa entrada en el proceso de decisión del perceptrón. Estos pesos son los que serán ajustados durante el entrenamiento. Al peso w_0 se lo llama sesgo.
- Suma ponderada ($\sum w_i x_i$): el perceptrón calcula una suma ponderada de las entradas multiplicadas por sus respectivos pesos. Matemáticamente, esta suma se escribe como:

$$z = \sum_{i=1}^{n} w_i x_i$$

- Función de Activación (σ): la salida de la suma ponderada de la etapa anterior se evalúa mediante una función de activación $\sigma(z)$, que introduce la no linealidad y permite al perceptrón aprender relaciones más complejas entre las entradas y las salidas. En el caso del perceptrón clásico, se utiliza una función de activación escalón (Rosenblatt, 1958), que se define como:

$$\sigma(z) = \begin{cases} 1 & \text{si } z \geq 0 \\ 0 & \text{si } z < 0 \end{cases}$$

  Esta función transforma el resultado de la suma ponderada en una salida binaria, dependiendo del signo de z.

- Salida (y): la salida y del perceptrón es el resultado de aplicarle la función de activación a la suma ponderada.

En el caso de dos entradas binarias, si se asignan los pesos $w_1 = 1$ y $w_2 = 1$ y se define la función σ como:

$$\sigma(z) = \begin{cases} 1 & \text{si } z \geq 3/2 \\ 0 & \text{si } z < 3/2 \end{cases}$$

el perceptrón resultante implementa la operación lógica AND.

2.3.2. La red

Una única unidad de perceptrón no tiene utilidad real para resolver problemas complejos. Se demostró que la operación lógica XOR no puede implementarse con un solo perceptrón (Goodfellow et al., 2016). Al igual que sucede en el cerebro humano, la verdadera potencia de este modelo surge cuando múltiples perceptrones se combinan formando una red.

Las redes de perceptrones, o redes neuronales de ahora en adelante, están compuestas por miles, millones o incluso miles de millones de perceptrones, lo que permite modelar comportamientos más inteligentes y aproximar funciones complejas.

Las redes suelen organizarse en capas que se conectan entre sí. El tipo de capa más común es la totalmente conectada, en la que todas las neuronas de una capa se conectan con todas las neuronas de la capa siguiente, como muestra la Figura 2.7.

En una red neuronal, se distinguen dos capas fundamentales: la capa de entrada, determinada por el tamaño de la entrada, y la capa de salida, cuya configuración depende de la tarea que realiza la red. Las capas intermedias se denominan capas ocultas.

2.3.3. Tipos de redes neuronales

Las redes neuronales artificiales han evolucionado en diversas arquitecturas diseñadas para abordar tareas específicas de aprendizaje automático. A continuación, se describen los principales tipos de redes neuronales y se destacan sus características, aplicaciones y diferencias fundamentales.

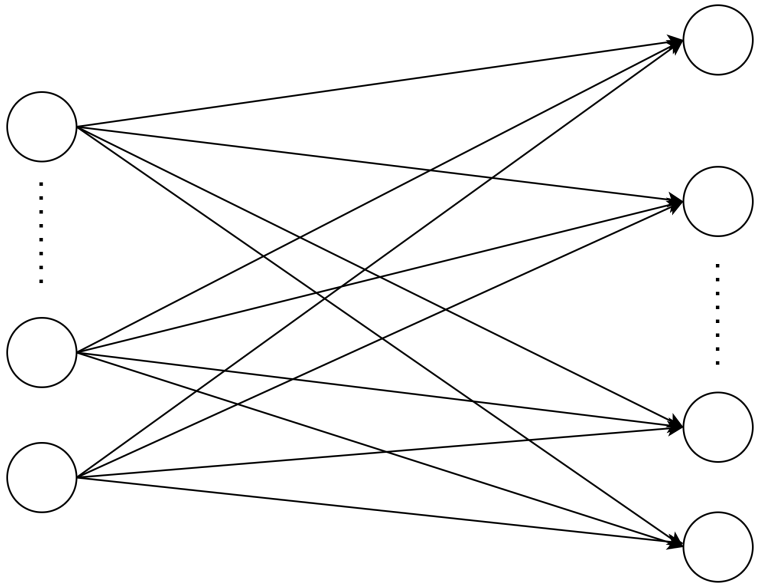


Figura 2.7: Representación de una capa totalmente conectada

Redes neuronales prealimentadas

En las redes neuronales de tipo prealimentada (feedforward), la información circula en un único sentido desde la entrada hasta la salida, sin existir conexiones cíclicas entre las neuronas. Estas redes son más sencillas de entrenar en contraposición a los otros tipos mencionados en esta sección, y se emplean principalmente para tareas de clasificación y regresión. Las redes feedforward son la base sobre la que se desarrollan otras redes más complejas, como las redes convolucionales.

Existe un tipo especial de red neuronal feedforward que se denomina red totalmente conectada en la que todas las neuronas de una capa están conectadas con todas las neuronas de la siguiente capa.

Redes neuronales recurrentes

Una red neuronal recurrente (RNN por sus siglas en inglés, recurrent neural network) es un tipo de red diseñada para procesar secuencias de datos. A diferencia de las redes neuronales feedforward, las RNN poseen conexiones cíclicas que permiten conservar y transmitir información a lo largo del tiempo.

Las RNNs son adecuadas para tareas que involucran datos secuenciales, como el procesamiento del lenguaje natural, el reconocimiento de voz y la predicción de series temporales. Sin embargo, las RNNs tradicionales sufren de problemas como el desvanecimiento y explosión del gradiente. Estos problemas limitan la capacidad de las RNNs de manejar dependencias a largo plazo. Para mitigar estos problemas se han desarrollado variantes más avanzadas de redes neuronales, como las redes de memoria a corto y largo plazo (LSTM por sus siglas en inglés, long short-term memory) (Sherstinsky, 2020) y las unidades recurrentes cerradas (GRU por sus siglas en inglés, gated recurrent units) (Cho et al., 2014).

Redes neuronales convolucionales

Las CNNs están diseñadas específicamente para procesar datos con estructura de grilla, como las imágenes, y se diferencian por emplear al menos una capa convolucional. Se le llama capa convolucional a una capa de neuronas que aplica la operación de convolución en lugar de la multiplicación matricial tradicional (O'Shea y Nash, 2015).

La convolución es una operación matemática que combina dos funciones (f y g) para generar una tercera (c). La función resultado (c) describe cómo la forma de una función (f) se modifica por la otra función (g). En el contexto del procesamiento de imágenes y datos, la convolución se emplea para extraer características locales (como bordes, texturas y patrones) mediante la aplicación de un filtro. Este filtro se desplaza sobre la entrada realizando la operación de convolución.

En el procesamiento de imágenes, la operación de convolución se realiza aplicando un filtro de tamaño reducido (generalmente una matriz cuadrada de dimensiones 3×3 o 5×5) sobre una matriz que representa la imagen. Cada posición del filtro tiene un valor numérico específico que se multiplica por los valores de los píxeles de la imagen que cubre. La suma de estos productos determina el valor del nuevo píxel en la imagen resultante de la convolución.

Para una imagen I y un filtro K, la convolución discreta se define en la Ecuación 2.2:

$$(I * K)(i,j) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} I(i-m, j-n)\, K(m,n), \tag{2.2}$$

En la Ecuación 2.2, (i,j) representa la posición del píxel de la imagen de salida, y a y b definen el rango del filtro en las dimensiones horizontal y vertical, respectivamente. En general, se utilizan filtros cuadrados donde a y b tienen el mismo valor.

En la Figura 2.8 se representa con una ilustración el resultado de la aplicación de un filtro de tamaño 3×3 a una matriz 6×6 y el correspondiente resultado.

En las redes neuronales convolucionales, los valores del filtro de la convolución corresponden a los pesos que se aprenden a partir de los datos. Es común aplicar múltiples filtros a la capa de entrada, cada uno de estos con valores independientes que serán aprendidos por la red durante el entrenamiento. El resultado de la convolución entre la entrada a la red y un filtro se denomina mapa de características (feature map). El tamaño del mapa de características en cada capa de la red está determinado por la cantidad de filtros utilizados en la capa, ya que cada filtro genera un mapa de características diferente al ser aplicado (O'Shea y Nash, 2015).

Las CNNs son altamente eficaces en tareas de visión por computadora, como el reconocimiento de objetos, la clasificación de imágenes y la segmentación semántica. Su capacidad para aprender características espaciales y reducir la dimensionalidad del problema sin perder información relevante es lo que las hace populares en aplicaciones de procesamiento de imágenes. También, las CNNs son utilizadas como bloques de construcción para otras arquitecturas más complejas, por ejemplo, SRGAN (O'Shea y Nash, 2015).

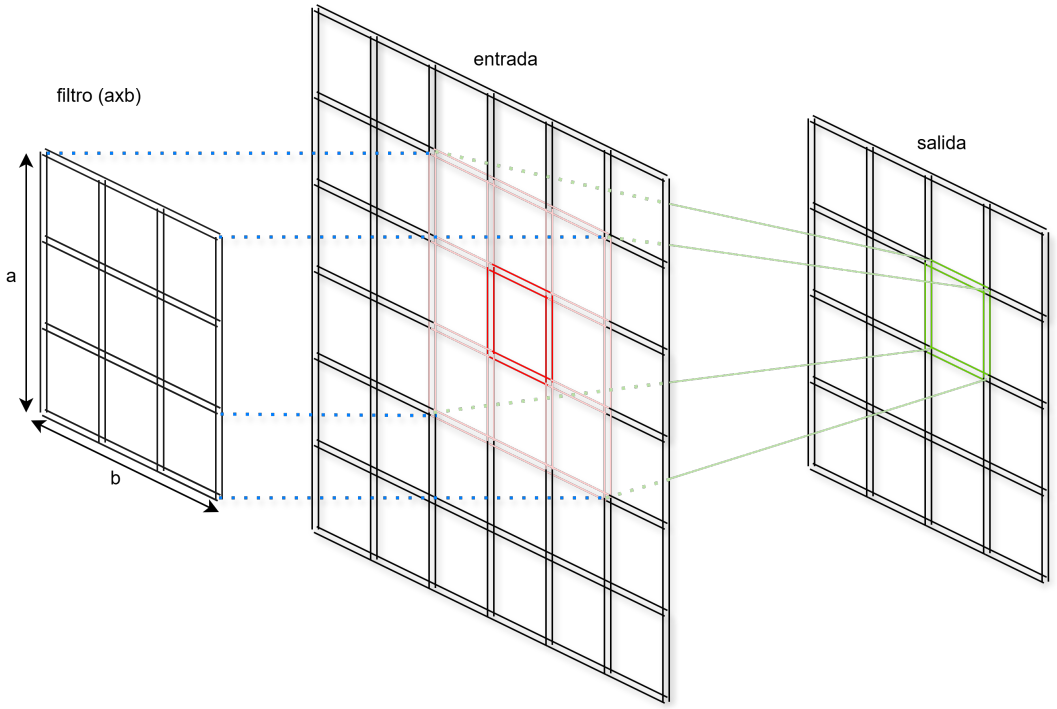


Figura 2.8: Ilustración demostrativa de resultado de aplicación de un filtro a una matriz

Red convolucional VGG

La red VGG es una arquitectura de red neuronal formada por capas convolucionales. El nombre VGG proviene de las siglas en inglés del equipo que la propuso: «Visual Geometry Group». Esta red presentada por Simonyan y Zisserman (2015) se caracteriza por el uso de filtros de convolución pequeños de tamaño 3×3 y una gran profundidad, que puede variar desde 11 hasta 19 capas de pesos. La arquitectura detallada de la red VGG se muestra en la Figura 2.9.

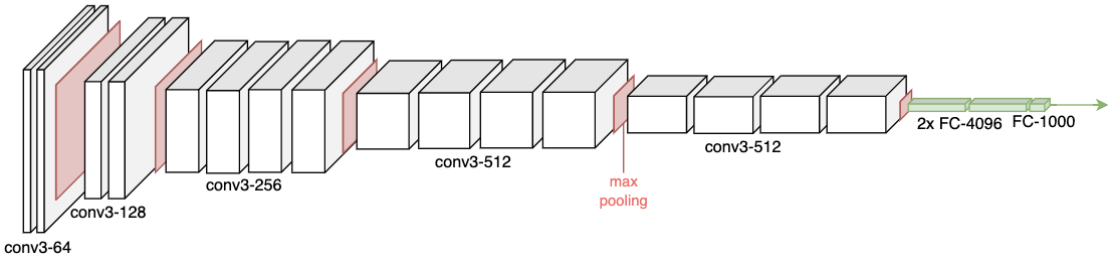


Figura 2.9: Representación gráfica de la red VGG-19.

La Figura 2.9 muestra una representación de la red VGG-19, que se compone de 19 capas de pesos. Cada capa convolucional está seguida por la función de activación de unidad lineal rectificada (ReLU por sus siglas en inglés, rectified linear unit) y utiliza un filtro de tamaño 3×3. La arquitectura se organiza en cinco grupos de convoluciones donde cada uno finaliza con una capa de max pooling (presentado en la Sección 2.3.4)

representada con el color rojo en la Figura 2.9. Estos grupos se diferencian por la cantidad de filtros que tiene cada convolución: comienza con 64 filtros y continúa con 128, 256 y 512 en los dos grupos finales. El aumento de los filtros por grupo es representado en la Figura 2.9 con el incremento del ancho de las capas convolucionales. También se diferencian en la cantidad de capas de convolución que tiene cada grupo, donde los primeros dos grupos tienen dos capas cada uno y los siguientes tres grupos tienen cuatro capas cada uno. Para completar la arquitectura, la red incluye dos capas completamente conectadas (presentado en la Sección 2.3.3) con 4096 neuronas cada una, seguidas de una última capa completamente conectada con 1000 neuronas y la función de activación softmax al final.

La red VGG fue entrenada específicamente para la clasificación de imágenes a gran escala y demostró un desempeño de estado del arte en conjuntos de datos como ImageNet (Deng et al., 2009). Debido al desempeño alcanzado por esta red se evidenció la importancia de la profundidad en las redes convolucionales para resolver tareas de clasificación.

Redes generativas adversarias

Las GANs, presentadas por Goodfellow et al. (2014), son un modelo generativo basado en la competencia entre dos modelos: un modelo G generador y un modelo D discriminador, ambos compuestos por redes neuronales. El modelo G crea muestras a partir de ruido y el modelo D intenta determinar si dichas muestras fueron generadas por G o forman parte de los datos reales. El entrenamiento se lleva a cabo de manera que G aprenda a generar muestras que sean indistinguibles de las originales y D aprenda a maximizar la probabilidad de clasificar correctamente los datos reales y los generados por G. Un ejemplo visual de una red GAN se muestra en la Figura 2.10. Para entrenar de forma eficiente y no sobreajustar a los datos de entrenamiento se propone realizar el entrenamiento en un bucle de k pasos de optimización de D y un paso de optimización de G. En ambos modelos se utiliza retropropagación (backpropagation) para actualizar los gradientes.

Entre los experimentos realizados por Goodfellow et al. (2014) los modelos generador y discriminador de la GAN fueron entrenados con distintos conjuntos de datos y lograron generar muestras competitivas en comparación a otros métodos. La arquitectura y entrenamiento propuestos fueron solo el comienzo ya que, a partir de su aporte, se crearon otras arquitecturas manteniendo las GANs como base. Entre estas arquitecturas se destacan SRGAN, DSRGAN, y ESRGAN, explicadas en detalle en la Sección 4.2.

Las aplicaciones de las redes GANs van más allá de la generación de imágenes. Se pueden utilizar las arquitecturas de las GANs para generar datos artificiales y así complementar conjuntos de datos de escaso material, como realizaron Frid-Adar et al. (2018) para aumentar el desempeño de una arquitectura de redes convolucionales aplicada al ámbito de la medicina.

2.3.4. Elementos de redes neuronales

En esta sección se describen componentes relevantes en las redes neuronales para el procesamiento de datos, incluyendo técnicas de normalización, reducción de dimensionalidad, funciones de activación y métodos de aumento de resolución.

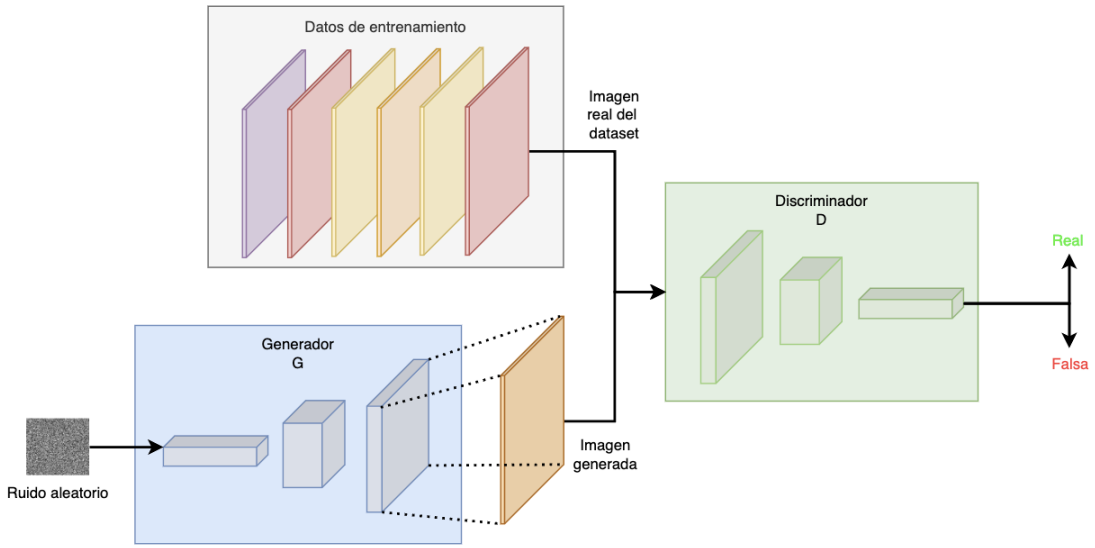


Figura 2.10: Representación de forma básica para funcionamiento y distribución de una red GAN.

Batch normalization

La técnica de normalización por lotes (batch normalization) fue presentada por Ioffe y Szegedy (2015). Esta técnica permite alcanzar buenos resultados en un menor tiempo de entrenamiento en comparación con a aquellas redes que no emplean batch normalization. Además, batch normalization permite usar tasas de aprendizaje (learning rates) mayores y también hace menos crítico el proceso de inicialización de los parámetros de la red.

Las mejoras en el proceso de aprendizaje se obtienen porque batch normalization mitiga el problema conocido como desplazamiento de la covarianza interna. La covarianza interna refiere al cambio de las distribuciones de las activaciones de la red provocado por las modificaciones de los parámetros durante el entrenamiento. Al reducir la covarianza interna se mejora la eficiencia del entrenamiento de la red.

Batch normalization es una técnica de normalización aplicada a cada lote (batch) durante el entrenamiento, en el que el descenso por gradiente considera la normalización independiente de cada característica (feature) de la red, ajustándolas para que tengan una media de 0 y una varianza de 1. Este ajuste de media y varianza se realiza sobre una estimación de la distribución del conjunto total de entrenamiento calculada a partir de los datos dentro de cada mini lote (mini-batch). Además, las distribuciones resultantes de cada activación pueden desplazarse y escalarse mediante dos parámetros aprendidos durante el entrenamiento. Batch normalization es aplicable a las entradas de cualquier capa dentro de la red neuronal.

En el trabajo original que introdujo batch normalization (Ioffe y Szegedy, 2015), se logró igualar el desempeño del mejor modelo de clasificación de imágenes en ImageNet solo con el 7 % de los pasos de entrenamiento requeridos respecto al modelo original.

Upsampling block

Los bloques de aumento de resolución (upsampling blocks), se diseñaron para incorporarse a la arquitectura de una red neuronal e incrementar la resolución de la entrada a la red. En las redes neuronales para superresolución se suele colocar una serie de capas convolucionales justo después de la entrada. Estas capas convolucionales provocan que el tamaño de la entrada disminuya o se mantenga igual, pero nunca aumente. Un mecanismo que permite lograr el aumento de resolución en arquitecturas de redes neuronales es el uso de los upsampling blocks usados en SRGAN, DSRGAN y ESRGAN.

Los upsampling blocks están compuestos por una capa convolucional que recibe $in\_channels$ canales provenientes de la red y genera $in\_channels \times upsample^2$ canales de salida, donde $upsample$ representa el factor por el cual se aumenta la resolución. El aumento de resolución luego de la convolución se obtiene al añadir una capa de PixelShuffle (Shi et al., 2016). Esta capa toma los $in\_channels \times upsample^2$ canales resultantes de la convolución y genera una única salida con un tamaño ampliado $upsample$ veces respecto al original. La Figura 2.11 ilustra gráficamente el funcionamiento del PixelShuffle.

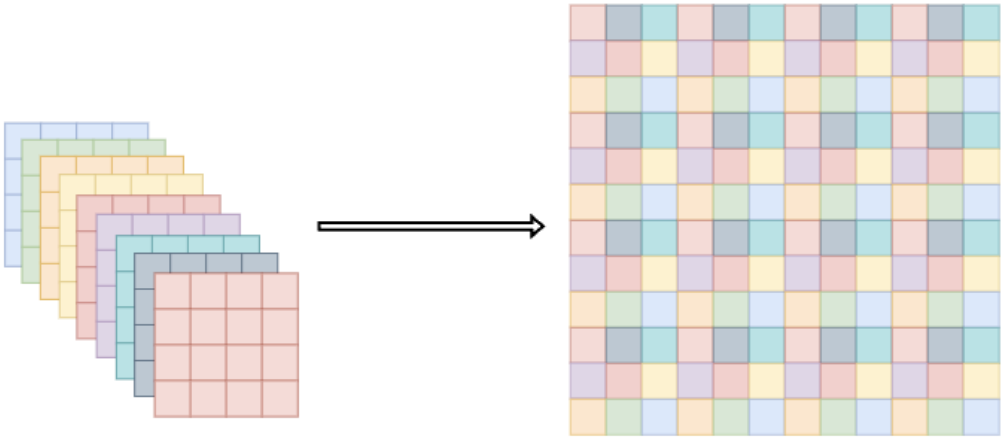


Figura 2.11: Representación gráfica de la capa de PixelShuffle contenida dentro de los upsample blocks.

En la Figura 2.11 se presenta una entrada con 9 canales, cada uno con dimensiones de 4×4. La capa PixelShuffle produce una salida de un único canal con dimensiones de 12×12. La resolución de la entrada se incrementa en un factor de 3 respecto a la original.

El upsampling block incluye la función de activación PReLU (Parametric ReLU) que se describe posteriormente en esta sección. La arquitectura completa del bloque se representa en la Figura 2.12, que muestra un ejemplo específico aplicado a la arquitectura DSRGAN.

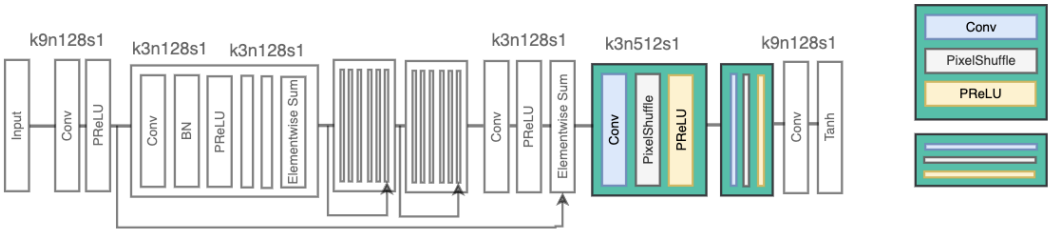


Figura 2.12: Upsampling block y su aplicación en la arquitectura DSRGAN.

Pooling

En las redes convolucionales, una operación utilizada para reducir las dimensiones de los datos es el pooling. El pooling emplea un filtro lógico que, basándose en un tamaño y un paso o zancada (stride) aplica una operación predefinida sobre los valores de la entrada afectados por el filtro. Esta operación disminuye la resolución espacial de las representaciones intermedias y reduce la complejidad computacional del modelo sin perder las características más relevantes.

El max pooling, por ejemplo, selecciona el valor máximo dentro de cada región definida por el filtro y se mueve el filtro según el stride especificado. Este proceso se ilustra en la Figura 2.13.

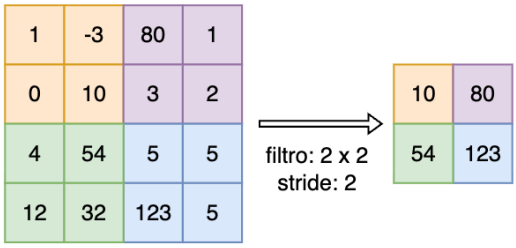
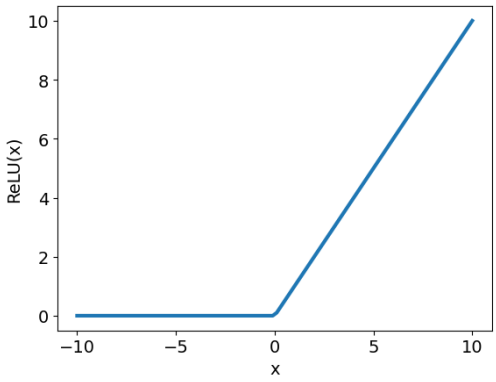


Figura 2.13: Representación gráfica del max pooling.

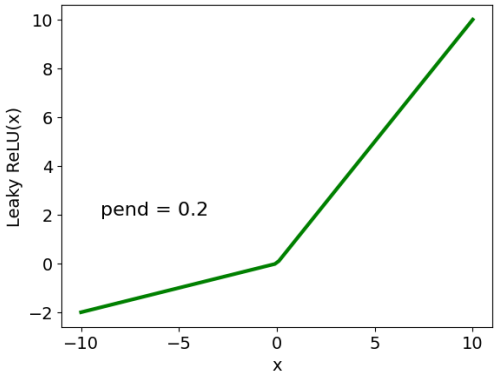
En la Figura 2.13 se muestra cómo utilizando un filtro de tamaño 2×2 y un stride de 2 se logra reducir una matriz de tamaño 4×4 a una matriz de tamaño 2×2.

Funciones de activación

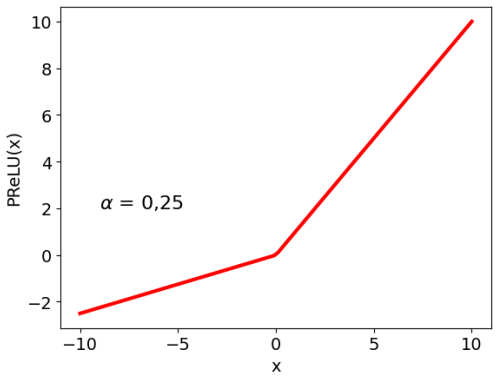
Como se introdujo en la Sección 2.3.1, la función de activación es una función matemática que se aplica a la salida de una neurona con el propósito de introducir no linealidad en el aprendizaje de la red. En la Sección 2.3.1 se menciona la función escalón y en esta sección se analizan otras cinco funciones de activación específicas: ReLU, LReLU, PReLU, tangente hiperbólica y Swish. En la Figura 2.14 se muestran los gráficos de las distintas funciones de activación presentadas.



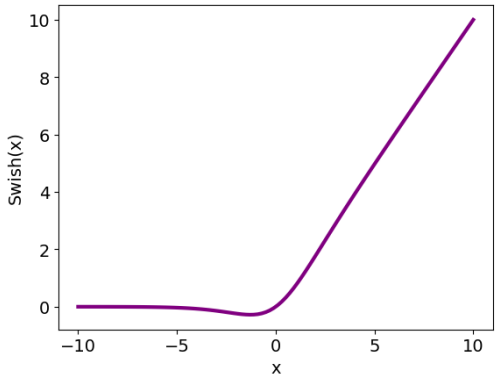
(a) Función de activación ReLU



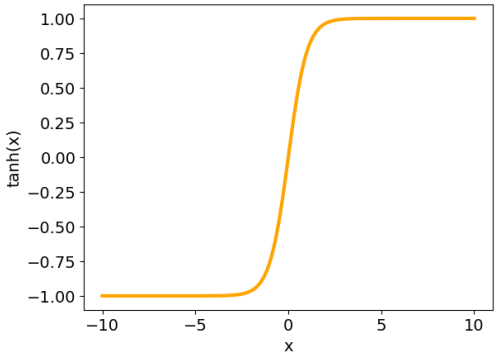
(b) Función de activación Leaky ReLU



(c) Función de activación PReLU



(d) Función de activación Swish



(e) Función de activación tangente hiperbólica

Figura 2.14: Representación de las funciones de activación con un gráfico correspondiente a cada una de ellas.

La función de activación ReLU se define matemáticamente como $ReLU(x) = max(0, x)$, donde x representa el valor de salida de la neurona. Una de las características que hacen útil a la función ReLU es que los gradientes calculados para entradas positivas son siempre 1 y los gradientes calculados para entradas negativas son siempre 0. Con estos valores del gradiente se busca mitigar problemas como la desaparición y explosión del gradiente en el entrenamiento de las redes, como se detalla en la Sección 2.3.5.

Basándose en la función de activación ReLU, surge una variante denominada Leaky ReLU. Esta modificación de la función ReLU introduce una pequeña pendiente para las entradas negativas, que en la ReLU original se mantenían constantes en 0. De este modo, la Leaky ReLU permite la propagación de un gradiente pequeño no nulo en las entradas negativas, a diferencia de la ReLU, donde el gradiente en esta región permanece igual a 0. La pendiente para las entradas negativas se define previamente y se mantiene constante durante todo el proceso de entrenamiento. La ecuación de Leaky ReLU se define como $LReLU(x) = max(0, x) + neg\_pend * min(0, x)$ siendo neg_pend la pendiente introducida.

La función de activación ReLU paramétrica (PReLU por sus siglas, parametric ReLU) es una variante de Leaky ReLU. En la PReLU, la pendiente a se convierte en un parámetro que es aprendido durante el entrenamiento de la red neuronal, en lugar de permanecer fijo como en el caso de la función Leaky ReLU. La ecuación de PReLU se define como $PReLU(x) = max(0, x) + a * min(0, x)$.

La tangente hiperbólica es una función de activación que, a diferencia de Leaky ReLU y Parametric ReLU, no está basada en la función ReLU. La función Tangente Hiperbólica tiene como codominio el rango $[-1, 1]$, lo que resulta útil en ciertas circunstancias. Por ejemplo, si se trabaja con DEMs normalizados y se define a los puntos sin datos con el valor -1, la función de tangente hiperbólica resulta especialmente adecuada si se utiliza como función de activación en la capa final. Matemáticamente, la tangente hiperbólica se define como $tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$.

La función de activación Swish se diferencia de las funciones basadas en ReLU al ser una función suave y no monótona. Swish se define matemáticamente como $Swish(x) = x \cdot \sigma(x)$ donde $\sigma(x)$ representa la función sigmoide $\sigma(x) = 1/(1 + e^{-x})$. A diferencia de ReLU, que anula por completo los valores negativos, Swish permite valores negativos atenuados.

2.3.5. Entrenamiento

El entrenamiento es el proceso mediante el cual una red neuronal, de cualquier tipo, aprende a realizar una tarea específica a partir de un conjunto de datos. El aprendizaje implica ajustar los pesos y sesgos internos de la red para minimizar la discrepancia entre sus predicciones y los valores deseados. El proceso de entrenamiento se lleva a cabo de forma iterativa y está guiado por un conjunto de componentes clave que son descritos a continuación.

Conjunto de entrenamiento

Se denomina conjunto de entrenamiento al subconjunto de datos utilizado para ajustar los parámetros de una red neuronal. El conjunto de entrenamiento debe ser representativo de los datos reales con los que se espera que la red infiera, permitiéndole así aprender patrones y características relevantes. La baja calidad y baja diversidad de los datos puede provocar problemas de sobreajuste y subajuste. Estos problemas afectan la capacidad de la red neuronal para generalizar en datos no vistos previamente.

Función de Pérdida

La función de pérdida cuantifica la diferencia entre las predicciones de la red y los valores esperados (etiquetas o resultados correctos). Esta función dirige el proceso de aprendizaje ya que, durante el entrenamiento, el objetivo de la red es minimizar la función de pérdida ajustando los pesos. Dependiendo del tipo de tarea, se eligen funciones de pérdida específicas, como la entropía cruzada para problemas de clasificación y el error cuadrático medio para problemas de regresión. En el caso de las GANs, se suele combinar la función de pérdida adversarial con otras funciones de pérdida, como la pérdida perceptual en el caso en que se esté trabajando con imágenes, de manera de equilibrar la calidad visual y la precisión del modelo.

Backpropagation

Backpropagation es un algoritmo presentado por Rumelhart et al. (1986) basado en el cálculo de derivadas parciales de la función de pérdida mencionada en la Sección 2.3.5. El algoritmo de backpropagation es el que permite a las redes aprender, ya que es utilizado para ajustar los parámetros.

Una vez que la red ha dado una predicción, se calcula la diferencia entre la predicción y el resultado esperado y se propaga desde la capa de salida a través de la red para actualizar los pesos en función de su contribución al error total. El proceso se repite en cada iteración del aprendizaje, utilizando optimizadores como el descenso de gradiente estocástico (SGD por sus siglas en inglés, stochastic gradient descent) o algoritmos derivados, como Adam, que mejoran la convergencia (Goodfellow et al., 2016). El algoritmo backpropagation permite que la red aprenda de los errores de forma eficiente, ajustando sus pesos en direcciones que minimizan la función de pérdida. En la Figura 2.15 se muestra un diagrama que ilustra el proceso de entrenamiento con backpropagation.

Optimizadores

La actualización de los pesos de una red neuronal con el gradiente de la función de pérdida se realiza a través de algoritmos conocidos como optimizadores. Estos algoritmos ajustan los pesos del modelo con el objetivo de minimizar la función de pérdida de manera eficiente.

El optimizador Adam (cuyo nombre proviene de adaptive moment estimation en inglés) fue presentado por Kingma y Ba (2014) como un método para la optimización estocástica con bajos requerimientos de memoria y que solo necesita gradientes de primer orden. La característica estocástica de este método proviene de la selección aleatoria de ejemplos de datos para el cálculo del gradiente y del ruido presente en la función objetivo a optimizar. El optimizador Adam calcula los parámetros de actualización de los pesos

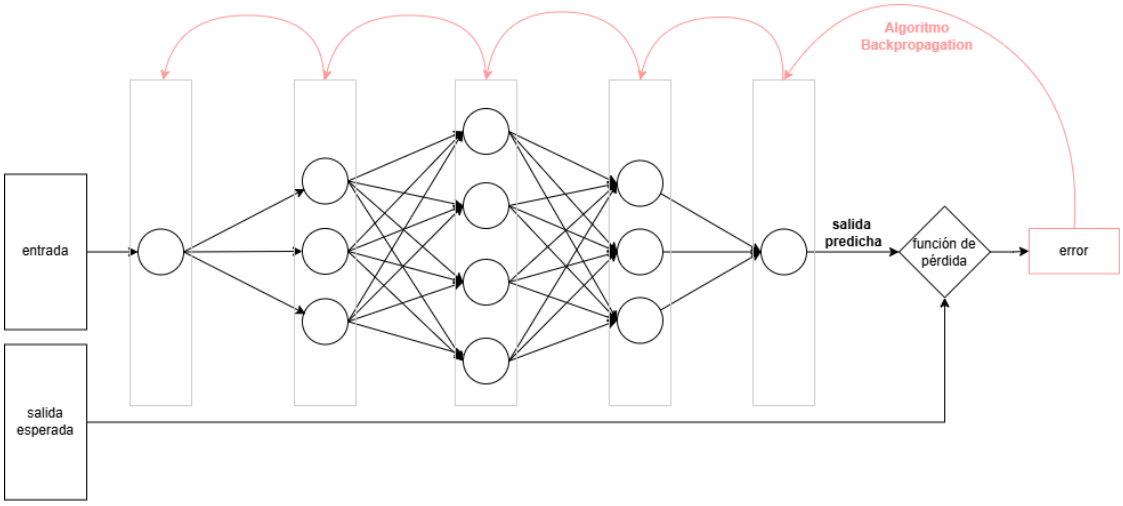


Figura 2.15: Diagrama de entrenamiento de una red neuronal con el algoritmo backpropagation

iterativamente a partir del primer y segundo momento del gradiente como muestra la Ecuación 2.3. El primer momento corresponde a la media del gradiente mostrado en la Ecuación 2.4, con su corrección para evitar el sesgo de inicialización como muestra la Ecuación 2.5. El segundo momento corresponde a la varianza sin centrar del gradiente de la Ecuación 2.6 que, al igual que el primer momento, tiene su propia corrección de sesgo de inicialización en la Ecuación 2.7. Para realizar las actualizaciones de los pesos, el optimizador Adam utiliza un hiperparámetro denominado learning rate (α en la Ecuación 2.3) que regula la magnitud de los cambios en los pesos del modelo durante el descenso por gradiente. Este hiperparámetro debe ajustarse manualmente y determina el tamaño del paso en cada iteración del algoritmo de optimización.

$$\theta_t = \theta_{t-1} - \alpha \times \hat{m}_t(\sqrt{\hat{v}_t} + \epsilon) \tag{2.3}$$

$$m_t = \beta_1 \times m_{t-1} + (1 - \beta_1) \times \nabla f_t(\theta_{t-1}) \tag{2.4}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.5}$$

$$v_t = \beta_2 \times v_{t-1} + (1 - \beta_2) \times \nabla f_t(\theta_{t-1})^2 \tag{2.6}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.7}$$

Adam combina las fortalezas de dos métodos de optimización anteriores a su publicación: AdaGrad y RMSProp. El optimizador Adam ha demostrado una mejora significativa en comparación con otros optimizadores, particularmente en el entrenamiento de redes neuronales convolucionales.

Entrenamiento por lotes

El entrenamiento por lotes o batches, también conocido como entrenamiento en mini-batches, es una estrategia en la que el conjunto de datos de entrenamiento se divide en pequeños subconjuntos o batches que se procesan de forma independiente durante cada iteración. Esta estrategia permite calcular y actualizar los gradientes de manera más eficiente en comparación con el entrenamiento sobre el conjunto completo de datos en cada paso. Al utilizar mini-batches, el modelo actualiza sus parámetros después de procesar cada lote y no luego de procesar cada elemento, lo que acelera el tiempo de entrenamiento, especialmente en conjuntos de datos de gran tamaño. Además, el uso de batches permite una paralelización más efectiva en unidades de procesamiento gráfico (GPU por su sigla en inglés, graphics processing unit), mejorando así la eficiencia computacional.

Problemas con el entrenamiento

Si bien el algoritmo backpropagation explicado en la Sección 2.3.5 es muy utilizado para el entrenamiento de los modelos, puede generar problemas que impidan que la red aprenda correctamente. Los problemas más comunes en el entrenamiento de redes neuronales son el sobreajuste (overfitting), la desaparición del gradiente (vanishing gradient) y la explosión del gradiente (exploding gradient), los que afectan especialmente a redes profundas.

El sobreajuste en redes neuronales es un problema que se presenta cuando un modelo aprende en exceso los patrones y el ruido específicos del conjunto de entrenamiento, lo que resulta en que el modelo final no tenga la capacidad de generalizar a la hora de ser utilizado con datos nuevos (Goodfellow et al., 2016). El problema de sobreajuste ocurre cuando la complejidad del modelo es demasiado alta para la cantidad y diversidad de los datos de entrenamiento utilizados, lo que provoca que el modelo se ajuste a características irrelevantes en lugar de capturar tendencias generales. Como consecuencia, se observa que al evaluar los modelos se obtiene un bajo error en el conjunto de entrenamiento, pero un error significativamente mayor en los conjuntos de validación y evaluación. Para mitigar el problema de sobreajuste, se emplean diversas técnicas, como la regularización L1 y L2, el dropout, el aumento de datos (data augmentation) y el uso de estrategias de validación cruzada para evaluar el desempeño del modelo de manera más robusta (Goodfellow et al., 2016).

La desaparición o desvanecimiento del gradiente es un problema que ocurre cuando los valores de los gradientes se vuelven extremadamente pequeños a medida que se retropropagan desde las capas de salida hacia las capas iniciales de la red. El problema es más frecuente en redes neuronales profundas o en arquitecturas como las RNNs (Bengio et al., 1994). Cuando los gradientes son muy pequeños, las actualizaciones de los pesos durante el entrenamiento se vuelven casi nulas, lo que impide que las capas más cercanas a la entrada aprendan de manera efectiva. Como resultado, la red se estanca y no mejora su desempeño. Uno de los mecanismos de mitigación para el desvanecimiento del gradiente

es forzar a las matrices jacobianas (matrices que contienen los gradientes) a conservar la norma en las direcciones relevantes del error y de esta forma evitar el colapso a 0 de los valores de los gradientes durante el entrenamiento (Pascanu et al., 2013).

La explosión del gradiente ocurre cuando los valores de los gradientes aumentan de manera exagerada a medida que se retropropagan a través de la red. Esto sucede cuando los pesos de la red son inicializados con valores muy grandes o cuando la arquitectura es muy profunda, lo que amplifica las multiplicaciones sucesivas de los gradientes. Cuando ocurre la explosión del gradiente, los pesos se actualizan con valores extremadamente grandes, lo que genera inestabilidad en el proceso de entrenamiento. Como resultado, la función de pérdida puede oscilar de manera abrupta o tomar valores infinitos, lo que impide la convergencia del modelo (Pascanu et al., 2013). Una de las soluciones más simples para el problema de la explosión del gradiente es el recorte del gradiente si su norma supera un cierto umbral predefinido. Utilizando este recorte se evita que el gradiente crezca sin control y el entrenamiento de la red diverja (Pascanu et al., 2013).

2.4. Metricas

Las métricas son un elemento fundamental en la evaluación y análisis de sistemas en diversas disciplinas. El principal objetivo al utilizar métricas es medir el desempeño de un proceso, modelo o sistema en estudio. La selección de métricas adecuadas es un aspecto crítico, pues influye en la validez y fiabilidad de los resultados obtenidos. Por lo tanto, se debe tener cuidado de seleccionar únicamente métricas relevantes para el fenómeno estudiado. Esta sección presenta las principales métricas empleadas en este proyecto de grado para la evaluación de los modelos.

2.4.1. PSNR

Entre las métricas cuantitativas relevantes para el proyecto de grado se encuentra la Proporción Máxima de Señal a Ruido (PSNR por sus siglas en inglés, peak signal-to-noise ratio) que mide la diferencia entre una imagen generada y una real teniendo en cuenta el ruido en la imagen generada (Moreira et al., 2023).

El PSNR para un DEM (Jiao et al., 2020) se calcula con la Ecuación 2.8, donde el término ΔS_i representa la diferencia entre el valor máximo y mínimo que se encuentra en el DEM particular al cual se le está calculando la métrica.

$$\text{PSNR} = 10 * log_{10} \frac{\Delta S_i^2}{\text{MSE}} \tag{2.8}$$

El error cuadrático medio (MSE por sus siglas en inglés, mean square Error) se calcula con la Ecuación 2.9, donde n representa la cantidad de celdas del DEM y e_i corresponde a la elevación en la celda i.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (e_i - \hat{e}_i)^2 \tag{2.9}$$

Al analizar los resultados generados se busca que el PSNR sea lo mayor posible.

2.4.2. MAE y RMSE

Para evaluar los DEMs generados se deben utilizar métricas que tengan en consideración las características del terreno (Zhang y Yu, 2022). Las métricas más comunes para comparar los DEMs generados con los originales son MAE y RMSE. Estas métricas proporcionan una medida global de cuán similar es el DEM generado con respecto al original (Zhang y Yu, 2022).

Tanto MAE como RMSE se calculan celda a celda mediante la comparación con los valores del DEM original. Los atributos evaluados son la altura o elevación, la pendiente y la orientación. Para calcular la métrica MAE se utiliza la Ecuación 2.10, donde para cada celda i del DEM generado se calcula la diferencia con el valor de la celda en el DEM original considerando una característica v del terreno (elevación, pendiente u orientación).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|v_i - \hat{v}_i| \tag{2.10}$$

Al igual que en la ecuación anterior, el RMSE se calcula celda a celda para la elevación, la pendiente u orientación, como se muestra en la Ecuación 2.11.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_i - \hat{v}_i)^2} \tag{2.11}$$

Los valores de MAE y RMSE representan la diferencia entre el DEM generado y el original, por lo que se tiene como objetivo la minimización de estas métricas para mejorar el desempeño de la red.

La métrica MAE representa una medida intuitiva de la precisión del DEM generado en comparación con el original. Unos pocos valores muy desalineados entre el DEM generado y el DEM original pueden incrementar considerablemente el RMSE, mientras que el impacto de estos valores en la métrica MAE es menos pronunciado.

2.4.3. Max error

El error máximo absoluto (max error) representa el máximo error obtenido en la comparación celda a celda entre los DEMs originales y los DEMs generados. La métrica max error se calcula con la Ecuación 2.12, en donde SR es el DEM súper resuelto y HR es el DEM original en alta resolución. El valor de max error se obtiene iterando sobre todas las celdas de ambos DEMs (SR y HR) tomando el valor absoluto de la diferencia por celda y conservando el máximo valor encontrado. Un bajo error máximo indica un buen resultado.

$$\text{max error} = \text{máx}(|SR_i - HR_i| \ \ \forall i) \tag{2.12}$$

Al considerar las métricas RMSE, MAE y max error, se abarca el uso de las normas matemáticas L_2, L_1 e L_∞, respectivamente. Esto es especialmente útil para evaluar los distintos DEMs generados por las redes.

2.4.4. LE90

Una métrica relevante para reportar es el error absoluto calculado mediante el percentil 90 del error lineal o LE90. Esta métrica representa el error máximo bajo el cual se encuentran el 90 % de los datos.

El LE90 se determina en dos pasos. Primero, se calcula para cada celda el error lineal entre los DEMs superresueltos generados y los DEMs originales en alta resolución. En la Ecuación 2.13, SR representa el DEM superresuelto y HR el DEM original en alta resolución, obteniendo así el tensor LE que almacena en cada celda la diferencia entre los DEMs comparados.

$$\text{LE} = |SR - HR| \tag{2.13}$$

En segundo lugar, se identifica el valor bajo el cual se encuentran al menos el 90 % de los errores, como muestra la Ecuación 2.14.

$$\text{LE90} = \mathit{percentil}(90,\ \text{LE}) \tag{2.14}$$

La métrica LE90 debe ser reportada junto con las métricas MAE y RMSE, dado que puede proporcionar una mejor interpretación de las diferencias observadas entre estas métricas, especialmente en presencia de valores atípicos.

2.4.5. Precision y recall

La precisión (precision) es una métrica que en el contexto de un problema de clasificación se define como el número de verdaderos positivos (instancias que se clasifican de manera correcta como positivas) dividido entre la suma de los verdaderos positivos y los falsos positivos (instancias que se clasifican como positivas pero que en realidad son negativas) como muestra la Ecuación 2.15. Esta métrica representa qué proporción de los casos clasificados como positivos son correctamente clasificados.

$$\text{precision} = \frac{T_p}{T_p + F_p} \tag{2.15}$$

En el caso de la tasa de verdaderos positivos (recall), considerando el mismo contexto de clasificación del párrafo anterior, esta métrica se define como el número de verdaderos positivos dividido entre la suma de los verdaderos positivos y el número de falsos negativos (aquellas instancias que se clasifican como negativas pero que eran positivas) como muestra la Ecuación 2.16. Esta métrica representa la proporción de los casos clasificados como positivos con respecto al total de ejemplos positivos que se tienen para clasificar.

$$\text{recall} = \frac{T_p}{T_p + F_n} \tag{2.16}$$

2.4.6. Métricas geográficas

Existen métricas geográficas que evalúan los DEMs desde su semántica y no como modelo matemático. Un ejemplo es el método propuesto por Zhang y Yu (2022) que evalúa puntos pico y puntos de valle a partir del DEM original y verifica su preservación en los DEMs generados. Se utiliza como métrica el desplazamiento de estos puntos para evaluar qué tan bien el método genera los DEMs. Esta métrica suele complementarse con la evaluación de redes de drenaje (Zhang y Yu, 2022).

Capítulo 3

Trabajos relacionados

En este capítulo se presentan los resúmenes de los trabajos publicados que resultaron relevantes durante la investigación preliminar del proyecto de grado. Al investigar y recopilar los trabajos presentados se obtuvo una visión general del estado del arte para la tarea de superresolución con redes neuronales. La investigación se centró particularmente en arquitecturas de redes neuronales basadas en GANs. También se analizaron y resumieron trabajos que demostraron la efectividad de las redes GAN en la superresolución de DEMs.

3.1. Superresolución en DEMs

En esta sección se presenta un trabajo sobre el análisis del estado del arte en el uso de redes neuronales profundas para la superresolución de DEMs en el que se destacaron las ventajas, desafíos actuales y desafíos futuros del uso de estas redes. Además, en esta sección se analiza un estudio comparativo que evaluó diversos enfoques de superresolución, incluidos métodos basados en GANs, para determinar su eficacia en la superresolución de DEMs.

3.1.1. Estado del arte para la superresolución de DEMs

Ruiz-Lendínez et al. (2023) presentaron una revisión del estado del arte sobre el uso de métodos de aprendizaje profundo aplicados a DEMs. El estudio analizó diversas arquitecturas y enfoques utilizados en tareas como la superresolución y la clasificación de terreno.

Se presentaron distintas técnicas basadas en CNNs y GANs, detallando sus ventajas y limitaciones en el procesamiento de DEMs. Además, el artículo discutió los desafíos en la aplicación del aprendizaje profundo a DEMs, como la disponibilidad de datos de alta calidad, la necesidad de modelos eficientes en términos computacionales y la generalización de los modelos en diferentes tipos de terreno.

Finalmente, los autores destacaron la importancia de futuras investigaciones en tres áreas: las fuentes de datos, los modelos de aprendizaje profundo y las aplicaciones a problemas reales. Los autores plantearon además la importancia de desarrollar a futuro modelos de caja blanca que sean más transparentes con el procesamiento de los datos.

3.1.2. Comparación de métodos para superresolución de DEMs

Zhang y Yu (2022) realizaron una comparación de métodos de superresolución enfocada en DEMs. El trabajo se centró en investigar si los métodos basados en GAN, que demostraron tener éxito en mejorar la resolución de imágenes, son también aplicables a los DEMs. También se compararon los resultados de los métodos basados en GAN con la interpolación bicúbica. Concretamente, los autores compararon los resultados de 4 métodos:

- Interpolación Bicúbica: se interpola el punto intermedio basándose en los datos de los puntos vecinos.
- SRGAN: arquitectura de superresolución basada en GAN con eficacia probada para imágenes.
- ESRGAN: arquitectura basada en SRGAN que demostró resultados superiores a SRGAN.
- CEDGAN: arquitectura basada en Condicional GAN diseñado especialmente para interpolación espacial.

Para el entrenamiento de los métodos basados en aprendizaje se normalizaron los valores de altura de los DEMs y se dividieron en recortes (crops) obteniendo finalmente 31,360 DEMs de 64×64 celdas. De estos 31,360 crops se utilizó un 80 % (25,088) para entrenar los modelos y 20 % (6,272) para validación. Los modelos se entrenaron para conseguir un aumento de resolución ×4 en los DEMs.

Para la evaluación se utilizó como métrica principal el RMSE para los valores de elevación. Además, para realizar un análisis más detallado se agrupó a los DEMs del conjunto de validación basándose en una escala de complejidad del terreno definida por los autores y se presentaron los resultados de RMSE para cada método en cada grupo. Finalmente en la etapa de validación los autores evaluaron la preservación de las características del terreno, particularmente la pendiente, orientación y líneas estructurales (ríos y crestas) para cada modelo.

Las evaluaciones realizadas demostraron que SRGAN fue el método que se comportó de manera más robusta, dando resultados levemente superiores a la interpolación bicúbica para la mayoría de métricas. Por otra parte ESRGAN arrojó peores resultados que SRGAN e interpolación bicúbica debido a exagerar o distorsionar las texturas. Finalmente CEDGAN fue el método con peores resultados dado que introdujo demasiado ruido en los DEMs generados.

3.2. Aplicación de GANs a la superresolución de imágenes

En esta sección se analizan trabajos sobre las arquitecturas SRGAN y ESRGAN que utilizaron GANs para la tarea de superresolución de imágenes. Se detalla cada arquitectura, con énfasis en sus elementos distintivos, funciones de pérdida y estrategias de entrenamiento.

3.2.1. Arquitectura SRGAN

Ledig et al. (2017) desarrollaron el modelo SRGAN basado en GANs para generar imágenes de alta resolución a partir de imágenes de baja resolución. Según los resultados presentados, SRGAN superó a los métodos existentes en términos de calidad visual.

Se utilizó para el entrenamiento de la red una muestra aleatoria de 350,000 imágenes en alta resolución de ImageNet (Deng et al., 2009) a las que se aplicó submuestreo (downsampling) bicúbico para obtener las imágenes de baja resolución. Los autores utilizaron una función de pérdida perceptiva (perceptual loss) que combina la pérdida de contenido (content loss) y la pérdida adversaria (adversarial loss) a diferencia de la arquitectura SRResNet, introducida en el mismo artículo, que solamente se entrenó utilizando la content loss (basada en VGG o en MSE) debido a que no es una GAN. La content loss está basada en la comparación de los feature maps de la red convolucional VGG19 (versión de la red VGG presentada en la Sección 2.3.3 que tiene 19 capas de profundidad). Por otra parte, la adversarial loss se calcula en base a la salida del discriminador. El uso combinado de la content loss y adversarial loss permitió que SRGAN no solo se enfoque en la precisión a nivel de celda, sino también en la fidelidad perceptiva de las imágenes generadas.

En el generador de la arquitectura SRGAN se utilizó una CNN profunda con 16 bloques residuales, como se muestra en la Figura 3.1.

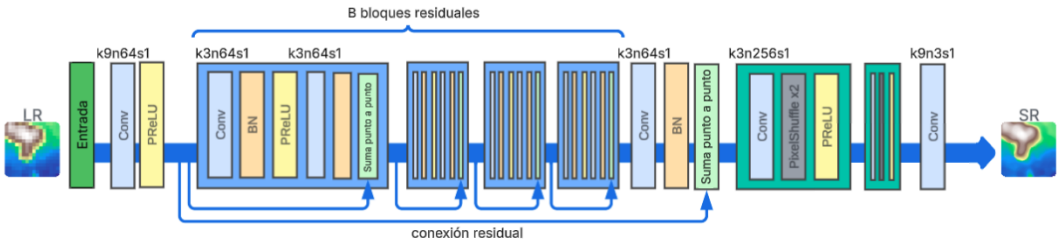


Figura 3.1: Arquitectura del generador de SRGAN

En la arquitectura del generador de SRGAN se destaca el uso de upsample blocks, mencionados en la Sección 2.3.4, para incrementar la resolución de la entrada. En SRGAN, la imagen de entrada se somete a distintas convoluciones que mantienen el tamaño de la imagen de entrada gracias al uso de stride con valor 1 y relleno (padding) con valor 1. Como consecuencia, el tamaño de la imagen permanece constante a lo largo de la red generador hasta que se aplica un mecanismo que permite aumentar su resolución. Este aumento de resolución se logra mediante el uso de upsample blocks, mencionados en la Sección 2.3.4, con un factor de reescalado de 4.

La arquitectura de la red discriminador se muestra en la Figura 3.2. El discriminador fue entrenado para distinguir entre imágenes reales de alta resolución y las imágenes generadas.

Se realizaron evaluaciones cuantitativas, comparando SRGAN con otros métodos de superresolución de última generación utilizando métricas estándar como PSNR y el Índice de Similitud Estructural (SSIM por sus siglas en inglés, structural similarity index measure). Aunque SRResNet estableció nuevos puntos de referencia en estas métricas, SRGAN sobresalió en calidad perceptiva, como lo demuestran los puntajes significativamente más altos en las evaluaciones de Puntuación de Opinión Media (MOS por sus siglas

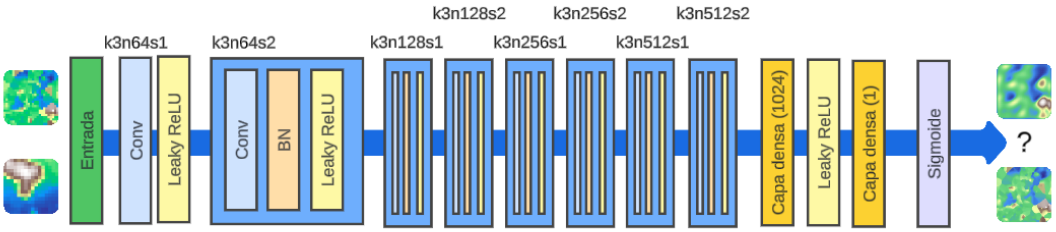


Figura 3.2: Arquitectura del discriminador de SRGAN

en inglés, mean opinion score), acercándose más a la calidad de las imágenes originales de alta resolución según evaluadores humanos.

El modelo SRGAN destacó por su capacidad para producir resultados visualmente atractivos incluso para regiones complejas y altamente texturizadas, que suelen ser un desafío para los métodos tradicionales de superresolución. Los resultados presentados por Ledig et al. (2017) demostraron que SRGAN puede aumentar efectivamente la resolución de las imágenes en un factor de $\times 4$.

En resumen, este artículo presentó un avance significativo en las técnicas de superresolución de imágenes. La integración de la adversarial loss con la content loss estableció un nuevo estándar para la generación de imágenes superresueltas de alta calidad. El éxito de SRGAN subrayó el potencial de las GANs en la mejora de tareas de procesamiento de imágenes, ofreciendo valiosos conocimientos y metodologías para futuras investigaciones y aplicaciones en visión por computadora.

3.2.2. Arquitectura ESRGAN

Wang et al. (2019) introdujeron la arquitectura ESRGAN. Esta es una evolución de la arquitectura SRGAN, sobre la que se presentaron mejoras que permiten a ESRGAN generar imágenes de mejor calidad y más realistas.

Los autores realizaron 3 mejoras importantes a SRGAN:

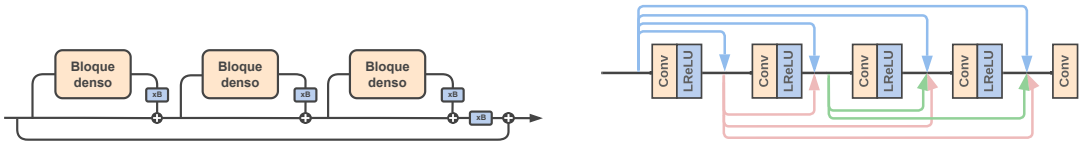
1. Arquitectura del generador: se modificó el generador introduciendo al residual-in-residual dense block (RRDB por sus siglas en inglés), que sustituye al residual block presentado en SRGAN. El RRDB incluye conexiones residuales dentro del bloque además de conexiones residuales entre distintos bloques y elimina las capas de batch normalization. Como activación dentro de los bloques, se sustituyó la función de activación Parametric ReLU por Leaky ReLU. A su vez, se realizó un escalado en la salida de cada bloque RRDB por un factor $\beta = 0{,}2$ antes de sumárselo a la rama principal. Por último, se inicializaron los pesos del generador de ESRGAN con inicialización He, pero dividiendo los mismos entre 10.

2. Discriminador mejorado: se utilizó un discriminador promedio relativista (RaD por sus siglas en inglés, relativistic average discriminator) que es una versión diferente de el discriminador relativista (RD por sus siglas en inglés, relativistic discriminator). El RD utilizado para SRGAN, en lugar de evaluar que tan real es una imagen, dadas dos imágenes distintas evalúa que tan real es una con respecto a la otra. Por otro lado, el discriminador RaD utilizado en ESRGAN, dada una imagen de un

tipo (real o generada), devuelve cuán real es la imagen comparada con el promedio de las imágenes del otro tipo en un mini-batch.

3. Pérdida perceptual mejorada: se modificó la perceptual loss, usando las características de VGG antes de la capa de activación en lugar de hacerlo después, como es el caso de SRGAN.

El RRDB se muestra en la Figura 3.3. A la izquierda se muestran las conexiones residuales entre los bloques densos del RRDB, y a la derecha las conexiones residuales dentro de un bloque.



(a) Residual-in-Residual Dense Block (RRDB) (b) Bloque denso

Figura 3.3: Izquierda: arquitectura de un RRDB, incluyendo el factor de escalado residual β. Derecha: arquitectura de un bloque denso de un RRDB, incluyendo sus capas y sus conexiones residuales.

En el generador de ESRGAN se utilizaron 23 RRDB durante el entrenamiento, aunque este número es un hiperparámetro más del modelo. A pesar del aumento de bloques y de la complejidad de estos comparados con los bloques residuales de SRGAN, se reduce la complejidad computacional con la eliminación de las capas de batch normalization.

El generador, como en toda GAN, tiene el objetivo de confundir al discriminador a la hora de distinguir las imágenes generadas de las reales. En el caso del RaD, se pretende durante el entrenamiento que el discriminador vea cada vez más realistas las imágenes generadas al compararlas con las reales. Es por esta razón que el generador de ESRGAN toma en cuenta los gradientes de las imágenes generadas así como los gradientes de las imágenes reales a diferencia de SRGAN, en donde solo las imágenes generadas influían sobre el generador. Los autores mencionaron que el uso de este nuevo discriminador ayuda a aprender bordes más nítidos y texturas más detalladas.

En conclusión, ESRGAN presentó un avance significativo en el campo de la superresolución con respecto a los modelos previos, logrando una mejor calidad perceptual y mayor realismo en las imágenes generadas. Esto quedó demostrado con la obtención del primer lugar en la competencia PIRM2018-SR Challenge (región 3), donde obtuvieron el mejor índice perceptual.

3.3. Aplicación de GANs a la superresolución de imágenes satelitales

En esta sección se analizan trabajos sobre las arquitecturas RS-ESRGAN, MCWESRGAN y SWCGAN que utilizaron GANs para la tarea de superresolución de imágenes satelitales. Estas arquitecturas introdujeron modificaciones a arquitecturas existentes como ESRGAN, con el objetivo de adaptarse a las particularidades de las imágenes satelitales.

3.3.1. Arquitectura RS-ESRGAN

Salgueiro Romero et al. (2020) propusieron una nueva arquitectura basándose en los estudios de superresolución anteriores y en la arquitectura ESRGAN. Esta nueva implementación denominada RS-ESRGAN consiste en adaptar el modelo ESRGAN al contexto de imágenes satelitales de varias bandas.

El objetivo de este trabajo fue entrenar una red en una base de imágenes de alta resolución provenientes del satélite WorldView para luego mejorar la resolución de las imágenes obtenidas con el satélite Sentinel-2 que son más abundantes y fáciles de obtener pero tienen menor resolución.

En este trabajo se destaca el preprocesamiento que se realizó sobre el conjunto de datos. En particular, se realizó la división por canales y se utilizaron puntos de referencia y de rotación para unir las imágenes de ambos satélites en el mismo conjunto de datos. Para la red RS-ESRGAN se realizó un preentrenamiento y posteriormente un ajuste fino (fine-tuning) de la red preentrenada.

Entre los resultados presentados, se muestran comparaciones de imágenes reales en baja resolución y su contraparte superresuelta generada por la RS-ESRGAN. También se reportan algunas de las métricas más comunes en la evaluación de calidad de imágenes como PSNR, SSIM y ERGAS. En todas las comparaciones con otras arquitecturas enfocadas al problema de superresolución y entrenadas de la misma manera, se concluyó que RS-ESRGAN es superior tanto en métricas como visualmente.

3.3.2. Arquitectura MCWESRGAN

Karwowska y Wierzbicki (2023) introdujeron la MCWESRGAN, una modificación de la ESRGAN enfocada en mejorar la calidad de imágenes satelitales y aéreas.

Los autores introdujeron un discriminador multicolumna, que en el caso particular de MCWESRGAN, consta de dos columnas en donde cada una es un discriminador. La predicción final del discriminador multicolumna corresponde al promedio de las salidas de cada columna, ya que esta estrategia resultó ser la más efectiva para combinar todas las salidas. Los autores mostraron que este cambio permitió reducir en un factor de 10 el tiempo de entrenamiento debido a la mejora en la evaluación de las imágenes generadas, además de mejorar los resultados de las métricas de evaluación utilizadas con respecto a ESRGAN.

Otra modificación realizada sobre la ESRGAN fue el uso de la función de pérdida Wasserstein (Wasserstein Loss). En particular, utilizaron Wasserstein Loss con penalización del gradiente. Esta modificación añade un término para penalizar al modelo si la norma de los gradientes se aleja de 1, forzando al discriminador a ser una función 1-Lipschitz. Los autores comentaron que la Wasserstein Loss permitió al discriminador diferenciar mejor las imágenes reales de las generadas, redujo significativamente el problema del desvanecimiento del gradiente y aceleró el entrenamiento de la GAN. Los autores también notaron que si se utilizaban imágenes de entrenamiento homogéneas, es decir, imágenes obtenidas en condiciones similares, la probabilidad de ocurrencia de desvanecimiento del gradiente aumentaba.

En conclusión, MCWESRGAN presenta una evolución sobre ESRGAN tanto en lo referido a los tiempos de entrenamiento como en las métricas de evaluación para imágenes satelitales.

3.3.3. Arquitectura SWCGAN

Tu et al. (2022) presentaron un enfoque novedoso para la superresolución de imágenes satelitales al introducir un nuevo modelo denominado SWCGAN, basado en GANs que combinó las fortalezas de los Swin transformers y las CNN. La motivación detrás de este trabajo surgió de las limitaciones de las CNN convencionales para capturar dependencias de largo alcance dentro de las imágenes, un requisito importante en las imágenes satelitales.

Los autores utilizaron para el bloque generador la auto-atención con ventanas desplazadas del Swin transformer, lo que permitió procesar eficazmente imágenes grandes sin necesidad de dividirlas en parches. En el generador se utilizó una capa convolucional para la extracción inicial de características superficiales. A continuación, se utilizó un bloque Swin transformer denso residual (RDSTB por sus siglas en inglés, residual dense swin transformer block) para extraer características profundas. Finalmente, se utilizó una etapa de upsampling hasta alcanzar un aumento de la resolución de la imagen en $\times 4$. Por otra parte, para el discriminador se utilizó una versión simplificada del Swin transformer que considera únicamente las primeras dos etapas del Swin transformer original. Los autores introdujeron en el discriminador el uso de RaD en lugar de un discriminador estándar.

Para validar el desempeño del método propuesto, se realizaron experimentos con el conjunto de datos de referencia UCMerced e imágenes satelitales. Los resultados demostraron que SWCGAN superó a los métodos existentes en algunas métricas, destacándose particularmente en la métrica LPIPS. Además, los estudios de ablación confirmaron la efectividad del bloque Swin transformer (STB por sus siglas en inglés, swin transformer block) dentro del RDSTB y justificaron la elección de la arquitectura. Los autores también señalaron que la inclusión del Swin transformer aumenta el tiempo de entrenamiento en comparación con los modelos basados en CNN estándar, aunque el modelo intermedio SWCG (entrenado solamente con la pérdida de contenido) resultó ser más rápido y obtener resultados similares respecto a otros modelos con más parámetros como RCAN.

En este artículo se realizaron contribuciones significativas al campo del procesamiento de imágenes satelitales al proponer una arquitectura GAN híbrida que aprovecha las capas convolucionales y los mecanismos de auto-atención basados en transformers para capturar dependencias de largo alcance en las imágenes y lograr mayor estabilidad en los resultados.

En resumen, el modelo SWCGAN desarrollado por Tu et al. (2022) representó un avance significativo en la superresolución de imágenes satelitales, ofreciendo una mayor precisión y robustez a través de una combinación novedosa de tecnologías.

3.4. Aplicación de GANs a la superresolución de DEMs

En esta sección se analizan los trabajos sobre DSRGAN y DEM-ESRGAN, dos arquitecturas de redes neuronales que aplicaron GANs a la superresolución de DEMs. Ambas arquitecturas adaptaron arquitecturas existentes como SRGAN y ESRGAN, para mejorar los resultados al utilizar DEMs como entrada. Se describen las modificaciones estructurales introducidas en los generadores y discriminadores, las funciones de pérdida utilizadas y los procedimientos de entrenamiento.

3.4.1. Arquitectura DSRGAN

Demiray et al. (2021) presentaron la DSRGAN, una arquitectura que adapta SRGAN para mejorar la resolución espacial de DEMs. Se muestra la arquitectura del generador y discriminador propuesto en la Figura 3.4.

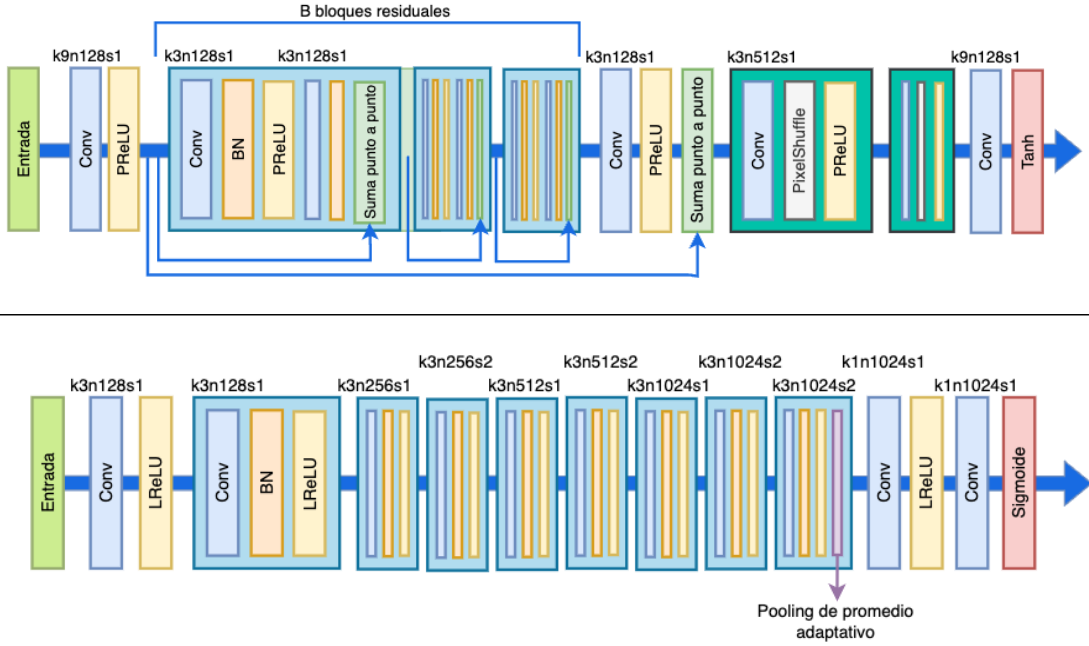


Figura 3.4: Representación de generador de DSRGAN (A) y Discriminador (B).

Al comparar la arquitectura DSRGAN con su modelo base SRGAN, se identifican diferencias fundamentales. Tanto en el generador como en el discriminador, cada convolución duplicó el número de filtros. Además, en la arquitectura del generador, la cantidad de bloques residuales se redujo a 8, la mitad en comparación con a los 16 de SRGAN. Se muestran estos cambios destacados en la Figura 3.5. Finalmente, en el generador se incorporó una función de activación de tangente hiperbólica al final de la arquitectura y para el discriminador se agregó en el último bloque una capa de pooling dinámico utilizando el promedio. La función de pérdida del generador se mantuvo igual que en SRGAN, exceptuando que en la content loss se dejó de utilizar la red VGG-19 para la extracción de características (feature extraction). La función de pérdida del discriminador se definió de igual manera que para SRGAN, utilizando entropía cruzada binaria entre lo que el discriminador distingue como real y falso.

Los resultados experimentales mostraron que la DSRGAN fue capaz de superar en MSE a los métodos de interpolación bicúbico y bilineal. Los autores también comentaron que DSRGAN tuvo un mejor desempeño en terrenos más planos que en terrenos con pendientes.

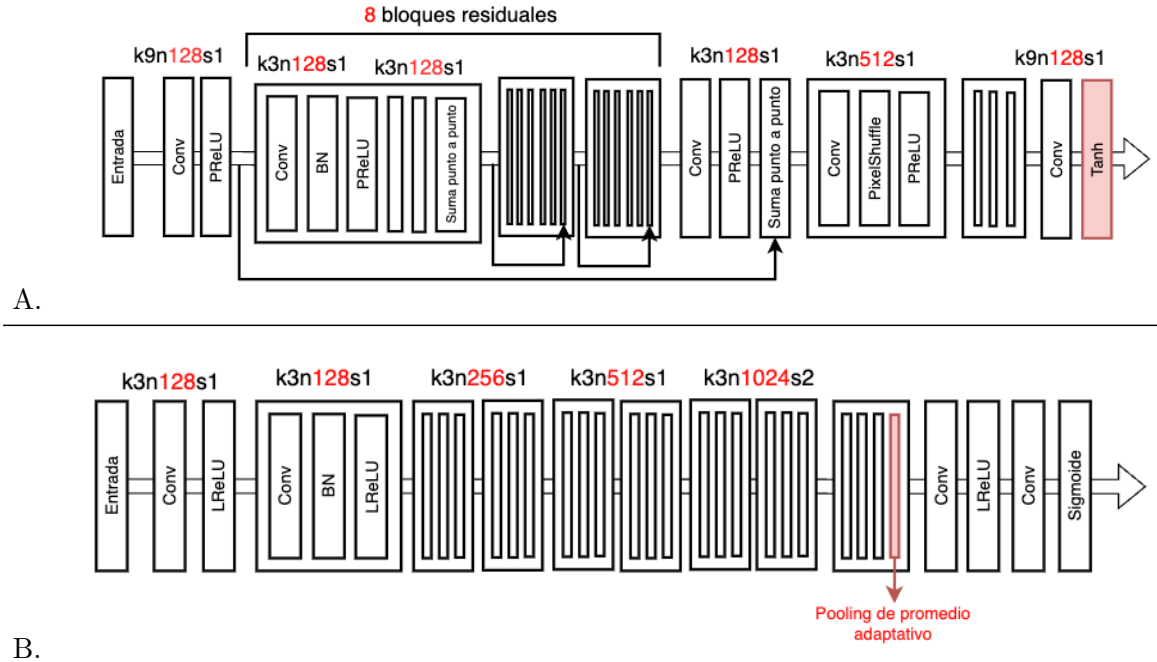


Figura 3.5: Representación de cambios de la DSRGAN con respecto a la SRGAN en las redes Generador (A) y Discriminador (B).

3.4.2. Arquitectura DEM-ESRGAN

Moreira et al. (2023) presentaron en su trabajo una manera satisfactoria de utilizar las GAN para la superresolución de DEMs extraídos de los satélites SRTM (imágenes con resolución de 30 metros) y ALOS PALSAR (imágenes con resolución de 12.5 metros).

Los autores utilizaron un conjunto de datos acotado de 70 pares de imágenes con una división 70-30 para el conjunto de entrenamiento y validación a los que no se les realizó ningún preprocesamiento fuera de un recorte en crops de tamaño 156×156 celdas. La GAN que utilizaron en el trabajo presentado se nombró «DEM-ESRGAN» y como su nombre indica, está basada en ESRGAN. Para diseñar DEM-ESRGAN modificaron ESRGAN para poder utilizar DEMs como entrada y obtener un factor de escalado predefinido. DEM-ESRGAN fue evaluada con las métricas PSNR, SSIM, NIQE, MSE y RMSE y comparada contra algoritmos de interpolación. El modelo de DEM-ESRGAN se mostró superior a estos algoritmos en las comparaciones de métricas numéricas. En el trabajo de Moreira et al. (2023), DEM-ESRGAN no fue comparada con otros métodos basados en aprendizaje profundo.

Los autores entrenaron la red con 10,000 épocas, luego con 50,000 épocas y por último con 100,000 épocas. Para cada uno de estos modelos se evaluaron la pendiente, los puntos críticos y las líneas estructurales generadas. Como resultado obtuvieron que la red con más épocas de entrenamiento mostró mejores resultados.

3.5. DSRT: aplicación de transformers a la superresolución de DEMs

Li et al. (2023) presentaron una arquitectura para la superresolución de DEMs utilizando la arquitectura transformer denominada DEM super-resolution Transformer (DSRT por sus siglas). En la DSRT se utilizaron cuatro módulos:

1. Modulo de extracción de características superficiales (shallow feature extraction): utiliza únicamente capas convolucionales para preservar datos de baja frecuencia de la entrada.

2. Modulo de extracción de características profundas (deep feature extraction): busca extraer información de alto nivel del DEM. Se compone de bloques GABs (Global Attention Blocks) secuenciales. Cada uno de estos bloques tiene múltiples capas de Global Attention Swin Transformer modificadas.

3. Modulo GeoTransform: capaz de generar DEMs en formato GeoTIFF nativamente.

4. Modulo de generación de DEM de alta resolución: bloque final que genera la salida en alta resolución.

Como función de pérdida para el DSRT, se utilizó una suma ponderada de varias funciones de pérdida con objetivos diferentes en una sola función denominada pérdida colaborativa (collaborative loss). En la collaborative loss se utilizaron cinco términos:

1. MAE celda a celda.

2. RMSE celda a celda.

3. DEM feature loss: refiere a una perceptual loss, en la que usan una extracción de características de la red VGG-16, similar a como se hace en SRGAN. Se computa la norma euclideana al cuadrado de las diferencias entre los feature maps obtenidos por la imagen real y la generada y se normaliza.

4. DEM edge loss: especializada en bordes. Primero se detectan los bordes de la imagen generada y de la real y luego se computa el promedio de las diferencias al cuadrado.

5. Global gradient loss: mide el promedio de las diferencias al cuadrado de los gradientes de cada imagen. Esta componente tiene un rol crucial a la hora de capturar características del terreno y en particular de sus bordes.

En el artículo se hicieron evaluaciones con tres resoluciones distintas y tres métricas distintas. También se evaluó el modelo en conjuntos de entrenamiento con terrenos de distintas características. En cada evaluación realizada se comparó a DSRT con SRGAN, ESRGAN, Tfasr y un modelo de interpolación bicúbica. DSRT fue el modelo con mejores resultados en la mayoría de las evaluaciones.

Capítulo 4

Metodología

En este capítulo se describe la metodología empleada en este proyecto de grado para resolver el problema de superresolución en DEMs, incluyendo la formulación del problema, las arquitecturas evaluadas, el proceso de entrenamiento y evaluación de los modelos y el preprocesamiento de los datos. Además, se detallan las configuraciones utilizadas en los modelos y el uso de ClusterUY para el entrenamiento.

4.1. Descripción del problema

El problema a resolver en el proyecto de grado consiste en la generación de DEMs de alta resolución para el territorio uruguayo a partir de datos disponibles en baja resolución. Los datos fueron proporcionados por el Instituto Geográfico Militar (IGM) e incluyen dos categorías: DEMs en alta resolución con una separación de 50 cm entre puntos y DEMs de baja resolución con una separación de 250 cm entre puntos. Los DEMs de baja resolución están disponibles por el IDEUY para la totalidad del territorio uruguayo pero los DEMs de alta resolución existen únicamente para zonas específicas en las que se realizaron vuelos con drones.

Con este proyecto de grado se pretende obtener una solución que permita generar DEMs de alta resolución a partir de los DEMs disponibles en baja resolución para todo el territorio uruguayo. La Figura 4.1 muestra una comparación visual entre un DEM de baja resolución (mitad izquierda) y un DEM de alta resolución (mitad derecha).

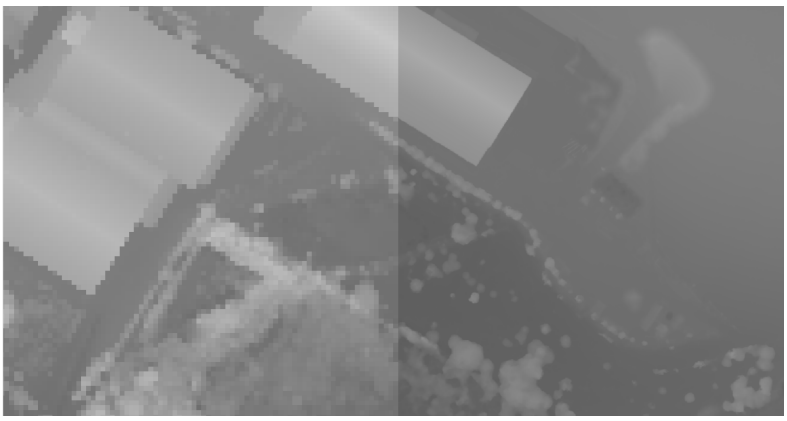


Figura 4.1: Comparación visual de DEMs de baja resolución con DEMs de alta resolución.

4.2. Arquitecturas evaluadas

Con el objetivo de desarrollar un modelo capaz de reconstruir los DEMs de alta resolución de manera óptima, se evaluaron cuatro arquitecturas de las mencionadas en el Capítulo 3: SRGAN, ESRGAN, DSRGAN y una variante de SRGAN. Entre estas arquitecturas, SRGAN presentó los mejores resultados en la comparación realizada por Zhang y Yu (2022). Los autores de esta comparación hicieron público el código utilizado en sus experimentos en la web `https://figshare.com/s/3fe0b1313938b0db994a?file=27988359`, incluyendo la definición de la arquitectura, el preprocesamiento de los datos, el entrenamiento de la red y los pesos de los modelos ya entrenados. Sin embargo, la versión de SRGAN utilizada por Zhang y Yu (2022) presenta modificaciones respecto a la versión original de SRGAN, por lo que fue considerada como una arquitectura diferente. Para distinguir ambas versiones de SRGAN, a partir de este punto se denominará SRGAN_{Z} a la variante modificada empleada por Zhang y Yu (2022) y SRGAN a la arquitectura original sin modificaciones. Además de estas dos arquitecturas, se evaluó la arquitectura ESRGAN, también considerada en las comparaciones de Zhang y Yu (2022), aunque con resultados inferiores a los de SRGAN. Por último, se seleccionó la arquitectura DSRGAN, que está diseñada como una mejora a SRGAN para trabajar con DEMs. Es relevante utilizar esta arquitectura, dado que en el trabajo que fue presentada no se la compara con ESRGAN ni con SRGAN.

La arquitectura DSRT, presentada por Li et al. (2023), reportó un desempeño de estado del arte en términos de RMSE en elevación, pendiente, aspecto y curvatura. Por esta razón, se intentó contactar a los autores con el fin de obtener el código fuente del modelo. Sin embargo no se obtuvo respuesta y dado que la arquitectura incluye componentes que se desvían de los objetivos de este proyecto de grado, se decidió no implementarla de forma independiente.

Para usar DEMs fue necesario adaptar las arquitecturas SRGAN y ESRGAN dado que están diseñadas para recibir imágenes como entrada y no DEMs. Además, SRGAN, ESRGAN y DSRGAN originalmente aumentan la resolución de la entrada en un número potencia de dos, por lo que también fue necesario ajustar las arquitecturas para permitir un incremento en un factor de cinco. Además, los DEMs pueden contener celdas sin datos, por lo que SRGAN, ESRGAN y DSRGAN también deben aprender a manejar estos casos de manera adecuada. En las siguientes secciones se describen las adaptaciones realizadas a cada arquitectura seleccionada para cumplir con los objetivos de este proyecto de grado. Además, en la Sección 4.2.4, dedicada a la arquitectura SRGAN_{Z}, se mencionan las diferencias entre esta variante y SRGAN original.

4.2.1. Arquitectura SRGAN

La arquitectura SRGAN, descrita en la Sección 3.2.1, fue adaptada a partir de una implementación disponible en la web `https://github.com/leftthomas/SRGAN` y se verificó que el código fuente fuera fiel a la descripción original de la arquitectura.

Dado que el objetivo en este proyecto de grado es aumentar la resolución en un factor de cinco, una modificación realizada a la arquitectura de SRGAN fue adaptar los upsample blocks para que lograran el aumento de resolución en un factor de cinco. Se realizaron ejecuciones concatenando upsampling blocks y también con un único upsampling block. Finalmente se decidió utilizar un único upsample block configurado para incrementar la resolución en un factor de cinco y esta adaptación también se realizó en el

resto de arquitecturas seleccionadas. Otra modificación necesaria fue adaptar la entrada y salida del generador y la entrada del discriminador de SRGAN para trabajar con un único canal, dado que la versión original de SRGAN está diseñada para recibir imágenes con tres canales de entrada. La última modificación realizada a SRGAN fue en la función de activación de la salida del generador. Esta función de activación debe ser capaz de representar valores en el rango $[0, 1]$ para las alturas, así como celdas vacías con el valor -1. Por esta razón, se reemplazó la función sigmoide utilizada originalmente por la función tangente hiperbólica, que tiene codominio en el rango $[-1, 1]$.

Con respecto al entrenamiento y siguiendo la metodología de Ledig et al. (2017), se llevó a cabo un preentrenamiento únicamente del generador. Por un lado, se denominó SRGAN «original» a la variante entrenada según la metodología propuesta por Ledig et al. (2017). Por otro lado, se denominó SRGAN «personalizada» a la variante en la que se incorporó la métrica MAE en la función de pérdida del generador durante el entrenamiento y utilizó exclusivamente MAE, en lugar de MSE, durante el preentrenamiento.

4.2.2. Arquitectura ESRGAN

La arquitectura ESRGAN, descrita en la Sección 3.2.2, es una evolución de la arquitectura SRGAN que demostró un mejor desempeño en la tarea de superresolución. Para la implementación de la arquitectura ESRGAN y el bucle de entrenamiento se utilizó como base el repositorio de github `https://github.com/eriklindernoren/PyTorch-GAN/tree/master/implementations/esrgan`, luego de verificar que su código fuera acorde a la descripción original de la arquitectura presentada por Wang et al. (2019).

Para adaptar ESRGAN al problema específico de superresolución en DEMs, fueron necesarias modificaciones. Al igual que SRGAN y DSRGAN, ESRGAN fue originalmente construida para aumentar en factores de 2 la resolución de las imágenes, en particular en un factor de 4. Por esta razón, el cambio más significativo a nivel de arquitectura fue la modificación de la etapa de upsampling para alcanzar el factor de cinco deseado. Se decidió reemplazar completamente el upsample block de ESRGAN por un único bloque que realice el reescalado $\times 5$.

Además de lo mencionado en la Sección 3.2.2, ESRGAN incorpora otras dos modificaciones respecto a SRGAN. La primera modificación consiste en la inclusión de la métrica MAE en la función de pérdida del modelo, particularmente en la pérdida de contenido. La segunda modificación es la introducción de un proceso denominado Network interpolation que consiste en entrenar únicamente el generador de ESRGAN utilizando PSNR como función de pérdida y posteriormente hacer fine-tuning al modelo incorporando al discriminador. Para este proyecto de grado también se entrenaron variantes de ESRGAN con un preentrenamiento basado en MAE en lugar de PSNR. Se llamó ESRGAN «original» a la variante entrenada con PSNR como en el trabajo de presentación de la arquitectura y se llamó ESRGAN «personalizada» a la variante preentrenada con la métrica MAE.

4.2.3. Arquitectura DSRGAN

Otra de las arquitecturas seleccionadas en este proyecto de grado fue DSRGAN descrita en la Sección 3.4.1. Esta arquitectura utiliza una red similar a SRGAN pero que difiere en algunos puntos clave, como se mencionó en la Sección 3.4.1 de trabajos re-

lacionados. Para la arquitectura DSRGAN no se encontró públicamente disponible una implementación, por lo que fue necesario reconstruirla a partir del trabajo de Demiray et al. (2021).

En el trabajo en el que se propone DSRGAN (Demiray et al., 2021), la ampliación de los datos se realiza en un factor de cuatro con respecto al tamaño inicial, mientras que en este proyecto de grado se requiere una ampliación de cinco veces el tamaño original de los DEMs. Debido a esta diferencia en la escala de aumento, la arquitectura original se vio alterada. En lugar de utilizar dos upsample blocks que duplican el tamaño de la imagen, se implementó un único bloque capaz de realizar directamente una ampliación de cinco veces el tamaño inicial. Debido a que la arquitectura del generador de DSRGAN ya agrega la función de tangente hiperbólica no fue necesario añadirla para representar las celdas sin datos con el valor -1.

Se realizó un entrenamiento similar al descrito en el trabajo presentación de DSRGAN (Demiray et al., 2021) y al modelo resultante se lo denominó DSRGAN «original». El trabajo de Demiray et al. (2021) no menciona cómo debería ser el preentrenamiento de la red y, por lo tanto, para DSRGAN «original» se utilizó el preentrenamiento establecido para SRGAN utilizando la métrica MSE como función de pérdida. Al igual que en el caso de SRGAN y ESRGAN, se evaluó una variante con modificaciones en la función de pérdida del generador y en el preentrenamiento del generador. En cuanto a la función de pérdida, se añadió la métrica MAE con un peso asociado a esta métrica que debe ser ajustado en la optimización hiperparamétrica. Respecto al preentrenamiento, se decidió utilizar MAE como función de pérdida. A esta versión de la red que incorpora la métrica MAE en el entrenamiento y preentrenamiento se la denominó DSRGAN «personalizada».

4.2.4. Arquitectura SRGAN$_Z$

Adicionalmente, se decidió evaluar el modelo SRGAN$_Z$ basado en SRGAN y publicado por Zhang y Yu (2022) en el repositorio público `https://figshare.com/s/3fe0b1313938b0db994a?file=27988359`. El modelo SRGAN$_Z$ fue entrenado con $25,088$ crops de 64×64 celdas obtenidos a partir de DEMs del Servicio Geológico de los Estados Unidos. Zhang y Yu (2022) explicaron que los crops utilizados en el entrenamiento fueron seleccionados aleatoriamente e incluyen una variedad de terrenos complejos con un gran rango de alturas (de 0.5 a 3,741 metros). Dado que la complejidad del relieve y los rangos de alturas del terreno uruguayo son considerablemente menores que las utilizadas en el trabajo original, se realizó un proceso de fine-tuning para mejorar el desempeño de SRGAN$_Z$ en el territorio uruguayo.

Llevar a cabo el proceso de fine-tuning en el modelo existente requirió ciertas modificaciones. Los cambios implementados fueron:

1. Escalado de la entrada: se escalan los crops de baja resolución previo al ingreso a la red para que sus dimensiones coincidan con las de los crops de alta resolución. Como resultado, a diferencia de SRGAN original, toda la red trabaja con las dimensiones de los crops de alta resolución. Esto implica que los upsample blocks no funcionan de la misma manera que en SRGAN, donde se utilizan para aumentar la resolución de los feature maps entrantes. En SRGAN$_Z$ se incorpora una capa convolucional con stride luego de la capa convolucional sub pixel (Shi et al., 2016) para mantener las mismas dimensiones a la entrada y salida de los upsample blocks.

2. Uso de Swish: en lugar de utilizar Parametric ReLU como función de activación en el generador y Leaky ReLU en el discriminador, se emplea Swish en ambos.

3. Salida del discriminador: se reemplaza la red feedforward de dos capas densas al final del discriminador por una capa convolucional que genera un solo feature map. Este feature map es luego procesado por una capa de average pooling que retorna un único valor.

En el trabajo en el que fue presentada, SRGAN_Z incluyó la métrica MAE en la función de pérdida del generador en el entrenamiento. Por lo tanto, el proceso de fine-tuning se realizó con un solo modelo a diferencia de SRGAN, ESRGAN y DSRGAN en las que se diferenció los modelos «original» y «personalizado».

4.3. Datos de entrenamiento

Los conjuntos de entrenamiento y evaluación fueron compuestos por DEMs de baja resolución junto con sus respectivas versiones en alta resolución, extraídos de distintas zonas de Uruguay. Particularmente se utilizaron DEMs del tipo DSM que contienen diversidad de terrenos y fenómenos geográficos dentro de Uruguay. En la Tabla 4.1 se listan las zonas que abarcan los DEMs con su respectiva área en kilómetros cuadrados (km^2) excluyendo regiones sin datos.

zona	área (km^2)
Abra de Castellanos	40.49
Durazno	17.10
Las Garzas	6.94
Punta Negra	5.22
City Golf	4.55
Isla Martín García	4.45
total	84.58

Tabla 4.1: Áreas de las zonas de donde se extrajeron los modelos digitales de elevación (DEM).

En esta sección se describe el procesamiento aplicado sobre los DEMs para utilizarlos en el entrenamiento y evaluación de los modelos seleccionados.

4.3.1. Preprocesamiento de DEMs

Algunos de los DEMs de alta resolución no presentaban una escala proporcional de 5 a 1 con exactitud respecto a los DEMs de baja resolución. Por esta razón, se decidió generar nuevos DEMs de baja resolución a partir de los de alta resolución y así garantizar una correspondencia precisa entre los DEMs de baja y alta resolución. La técnica utilizada para obtener los DEMs de baja resolución consistió en seleccionar filas y columnas alternadas de los DEMs de alta resolución. En particular, se tomaron una fila cada cinco filas, y una columna cada cinco columnas, comenzando siempre desde la

fila superior y la columna más a la izquierda. De esta manera se obtuvieron DEMs en los que el mapeo de baja resolución a alta resolución era conocido y exacto. A modo de ejemplo, si se considera que la celda (1, 1) de un DEM corresponde a la esquina superior izquierda, entonces la matriz formada por la celda (1, 1) en el DEM de baja resolución se corresponde con la matriz cuadrada de celdas cuyas esquinas son (1, 1), (5, 1), (1, 5) y (5, 5) en el DEM de alta resolución. En la Figura 4.2 se representa gráficamente la correspondencia de celdas entre un DEM de alta resolución y uno de baja resolución.

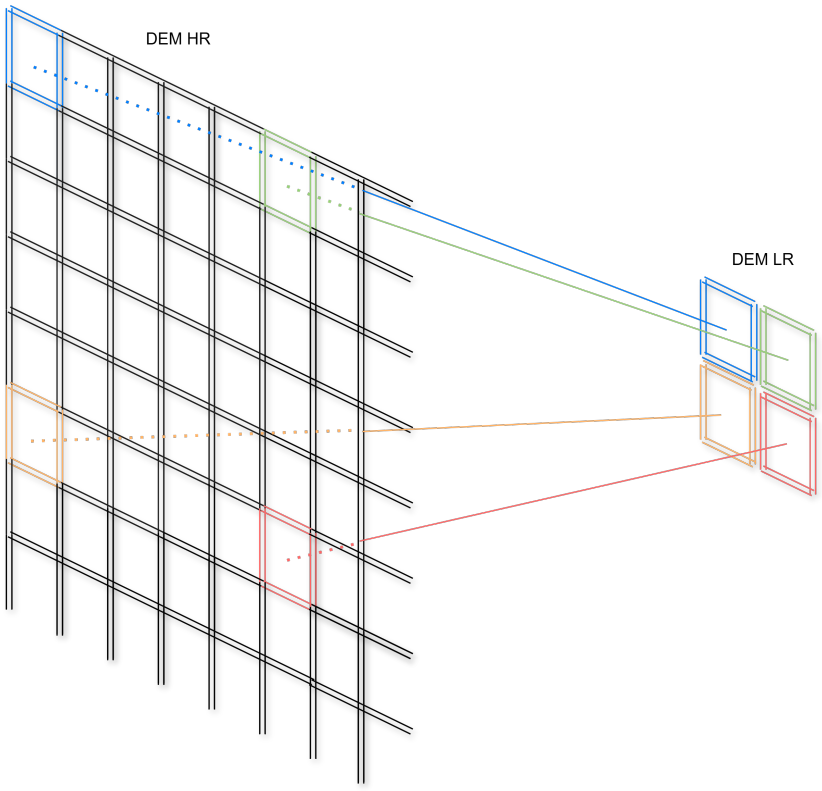


Figura 4.2: Correspondencia de celdas de un DEM de alta resolución con un DEM de baja resolución.

4.3.2. Recorte de DEMs

Para este proyecto de grado también se decidió recortar los DEMs en crops de igual manera que Zhang y Yu (2022) con el fin de reducir el tamaño de la entrada de los modelos y, en consecuencia, disminuir la memoria requerida para el entrenamiento y la inferencia. El tamaño de los crops se determinó en la etapa de análisis experimental. Para el recorte de crops se utilizó un stride que indicó la distancia entre el inicio de un crop y el inicio del siguiente en una dirección determinada (horizontal o vertical).

El valor del stride determinó el grado de superposición entre crops. Un stride menor al tamaño del crop significa que existe superposición y, cuanto menor sea el stride, mayor será la superposición. Por otra parte, cuanto más cercano sea el stride al tamaño del crop, menor será la superposición. Si el stride es igual o mayor al tamaño del crop, entonces los crops no se superponen en absoluto.

Dado que dos crops superpuestos no pueden pertenecer a conjuntos diferentes debido a la coincidencia parcial de sus celdas, se implementó una técnica de recorte que contempló esta restricción. Esta técnica asume que los crops finales tienen un tamaño de 16×16, 24×24 o 32×32. La técnica utilizada consta de tres etapas:

1. Recorte virtual de DEMs en crops de 96×96, denominados «crops base».

2. Selección del 80 % de los crops base para el conjunto de entrenamiento, un 10 % para el conjunto de validación y el 10 % restante para el conjunto de evaluación. Para asignar los crops base a cada conjunto, primero se ordenan los crops base de izquierda a derecha y de arriba a abajo dentro de cada DEM. Luego de ordenar los crops base, los primeros cuatro se asignan al conjunto de entrenamiento, el quinto se asigna al conjunto de evaluación, los próximos cuatro (del sexto al noveno) vuelven a asignarse al conjunto de entrenamiento y el décimo se asigna al conjunto de validación. Este patrón se repite para el resto de los crops base del DEM. De esta manera se garantiza que los conjuntos de datos estén distribuidos uniformemente dentro de cada DEM.

3. Recorte de los crops base en subcrops dentro cada conjunto (entrenamiento, validación y evaluación) al tamaño deseado. En caso de requerir superposición entre crops, esta superposición se genera entre subcrops de un mismo crop base y únicamente en el conjunto de entrenamiento.

Para recortar los crops base se recorre el DEM en ambas direcciones (horizontal y vertical) y se realizan los recortes correspondientes. El proceso finaliza cuando el siguiente crop base excede los límites del DEM. Esta situación provoca que los datos ubicados en los bordes del DEM sean excluidos de los crops base. Para evitar la pérdida de estos datos, previo a recortar el DEM se añadió un padding de celdas vacías del mismo ancho que los crops base. De esta manera, cuando un crop base excede las dimensiones del DEM con padding significa que dicho crop base no contiene datos.

Dado que los DEMs originales contenían regiones sin datos, al recortarlos se generaron tres tipos de crops: crops totalmente vacíos, crops con secciones vacías y crops sin secciones vacías. Las celdas vacías representan un 29.2 % del total de celdas en los DEMs utilizados, por lo que al emplear tamaños de crop relativamente pequeños en comparación con el tamaño de los DEMs, la proporción de crops completamente vacíos tiende a ser cercana a este porcentaje. Por esta razón, se decidió descartar los crops totalmente vacíos con el objetivo de reducir la cantidad de instancias de entrenamiento. Se consideró que el modelo es capaz de aprender a reconstruir regiones sin datos a partir de crops que contengan simultáneamente regiones con datos y regiones vacías. Luego de normalizar los valores de altura al intervalo $[0, 1]$, se asignó el valor distintivo -1 a las celdas sin datos para que los modelos pudieran identificarlas adecuadamente y aprender su reconstrucción.

4.3.3. Visualizacion de DEMs

En este proyecto de grado, los DEMs se visualizaron como imágenes a color mediante una transformación de escala a color, aunque originalmente contenían un único canal de datos. Los colores utilizados van desde el azul, representando las zonas más bajas del DEM, hasta el blanco, que indica las zonas más altas. En la Figura 4.3 se muestran algunos ejemplos de DEMs que fueron utilizados en este proyecto de grado.

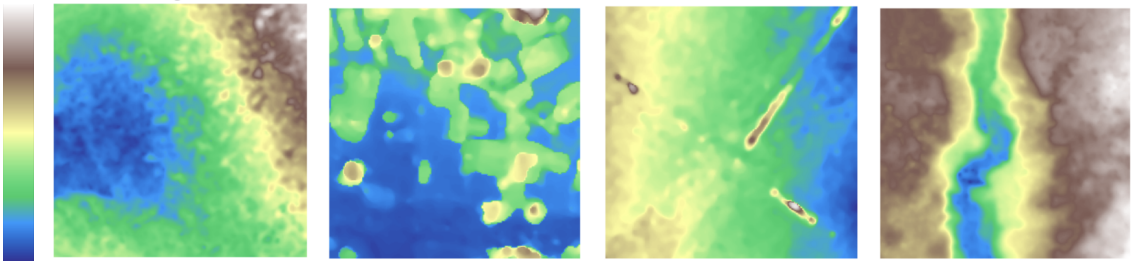


Figura 4.3: Representación visual de DEMs.

Si bien en este proyecto de grado no se hizo un uso intensivo de la herramienta QGIS, sí desempeñó un papel clave al facilitar las tareas de visualización de los DEMs en formato TIFF. Esto fue importante durante las etapas de construcción y procesamiento del conjunto de datos. QGIS permitió detectar zonas sin datos en los DEMs, validar visualmente el algoritmo de recorte de crops y verificar la conservación de metadatos.

4.3.4. Normalización de datos

Dado que los DEMs presentaron diferentes rangos de altura, se normalizaron al intervalo $[0, 1]$ para favorecer el aprendizaje de los modelos. Esta normalización se realizó mediante la Ecuación 4.1.

$$DEM_{LR} = \frac{DEM_{LR} - DEM_{LR_{min}}}{DEM_{LR_{max}} - DEM_{LR_{min}}} \tag{4.1}$$

Donde $DEM_{LR_{min}}$ y $DEM_{LR_{max}}$ representan las alturas mínima y máxima del DEM en baja resolución respectivamente. Para desnormalizar un DEM superresuelto se utilizaron las alturas mínima y máxima del DEM en baja resolución correspondiente. Tanto los cálculos de mínimo y máximo como la normalización de los DEMs se realizaron excluyendo las celdas sin datos, a las que se les asignó el valor -1.

4.4. Tecnologías utilizadas

Esta sección presenta las tecnologías más relevantes utilizadas en este proyecto de grado. El código fuente fue escrito en el lenguaje de programación Python (Van Rossum y Drake, 2009) en su versión 3.12 y se utilizaron tres librerías que desempeñaron un papel central en el desarrollo: PyTorch, Optuna y Rasterio.

Optuna

Optuna es un software de código abierto especializado en la optimización hiperparamétrica para modelos de aprendizaje automático. Las principales razones por las cuales se utilizó Optuna para la búsqueda de hiperparámetros en este proyecto de grado son:

- Facilidad de uso e instalación: su API de Python es sencilla e intuitiva lo que facilita su integración con el código ya existente de los modelos. Esto permitió implementar la búsqueda de hiperparámetros con solo unas pocas decenas de líneas de código.

- Paralelización sencilla: Optuna permite ejecutar búsquedas de hiperparámetros en paralelo mediante múltiples procesos sin necesidad de modificar el código fuente. Basta con lanzar varias ejecuciones en paralelo utilizando el mismo comando y el software gestiona automáticamente la concurrencia permitiendo que los procesos ejecuten en paralelo. Esta funcionalidad resultó útil debido a las limitaciones en la disponibilidad de las GPUs al momento de realizar las ejecuciones.

- Algoritmos de muestreo: la principal razón para utilizar herramientas como Optuna en lugar de una búsqueda de grilla (grid search) es la reducción progresiva del espacio de búsqueda. Esta reducción es realizada por los algoritmos de muestreo que ajustan dinámicamente el espacio de búsqueda en función de los resultados obtenidos en iteraciones previas. Optuna combina dos estrategias de muestreo para definir el espacio de búsqueda: el muestreo relacional y el muestreo independiente.

  - Muestreo relacional: esta estrategia considera las interdependencias entre los diferentes hiperparámetros. Para ello, utiliza la información de ensayos anteriores para guiar la selección de los valores de los hiperparámetros.
  - Muestreo independiente: en esta estrategia los hiperparámetros se seleccionan de manera independiente, sin considerar los valores de otros hiperparámetros. Es útil para reducir el sesgo y evitar dependencias complejas cuando se desconoce la relación entre los parámetros. Este enfoque es similar a la búsqueda aleatoria (random search), pero puede ajustarse con técnicas adicionales para mejorar la exploración del espacio de búsqueda.

- Posibilidad de utilizar algoritmos de pruning: en Optuna, el término pruning refiere al proceso mediante el cual se interrumpe de forma anticipada el entrenamiento de un modelo cuando no muestra resultados prometedores. Para decidir si el entrenamiento de un modelo debe ser interrumpido, la librería ofrece diversos algoritmos como MedianPruner, PercentilePruner o SuccessiveHalvingPruner. Cada uno de estos algoritmos aplica un criterio diferente para determinar si detener o no un entrenamiento, en función de los resultados parciales reportados. En este proyecto de grado se utilizó MedianPruner, que interrumpe el entrenamiento si su mejor resultado intermedio es inferior a la mediana de los resultados obtenidos en el mismo paso por los modelos evaluados previamente en la búsqueda de hiperparámetros.

- Optuna Dashboard: es una herramienta de Optuna que permite visualizar los resultados de la búsqueda de hiperparámetros en un tablero interactivo con una interfaz intuitiva. Además, es posible monitorear en tiempo real el avance de la búsqueda de hiperparámetros con representaciones gráficas.

PyTorch

Pytorch (Paszke et al., 2019) es un paquete para el lenguaje Python que proporciona dos características de alto nivel: operaciones con tensores optimizadas, diseñadas para aprovechar la capacidad de cómputo de la GPU y un conjunto de modelos de redes neuronales profundas ya implementados. Es uno de los paquetes más utilizados para la construcción y entrenamiento de redes neuronales en Python. Los principales componentes empleados en este proyecto de grado son:

- torch.Tensor: librería de tensores similar a Numpy pero con un mejor soporte para el uso de GPU.
- torch.autograd: librería de diferenciación automática para tensores con soporte para todas las operaciones diferenciables en torch. Permite calcular de manera eficiente y transparente las derivadas requeridas para el entrenamiento de los modelos mediante backprogrpagation.
- torch.nn: librería de redes neuronales integrada con autograd para facilitar y flexibilizar su implementación.
- torch.utils: librería para el manejo del conjunto de datos de entrenamiento, destacando principalmente la clase DataLoader.
- torchvision: paquete que proporciona conjuntos de datos populares, arquitecturas de modelos y transformaciones específicas relacionadas con el área de visión por computadora (computer vision).

En este proyecto de grado se utilizó Pytorch para la implementación de todas las arquitecturas evaluadas, así como en los scripts de entrenamiento y manejo de los conjuntos de datos. La principal razón para su uso fue que el código base utilizado para todas las arquitecturas ya estaba implementado en Pytorch, por lo que las modificaciones necesarias se realizaron empleando recursos del mismo paquete.

El uso de Pytorch permitió optimizar el tiempo de desarrollo de funciones y algoritmos dentro del proyecto de grado. Por ejemplo, resultaron útiles las implementaciones de las funciones de error disponibles en la librería torch.nn, tales como BCEWithLogitsLoss(), L1Loss(), MSELoss() y BCELoss(). También se utilizó una una red VGG preentrenada del paquete torchvision para la extracción de características de los DEMs (2.3.3).

Rasterio

Una etapa fundamental de este proyecto de grado fue la obtención y manipulación de datos. Los datos utilizados para el entrenamiento de los modelos son DEMs que se encuentran almacenados en formato GeoTIFF. Los archivos con formato GeoTIFF contienen, además de la matriz con valores de elevaciones, varios metadatos georreferenciales que pueden perderse si no se manipulan correctamente. Dado que Python no tiene soporte nativo para este tipo de datos, se decidió utilizar la librería Rasterio (Gillies et al., 2013) para la lectura y manipulación de los DEMs.

Rasterio es una librería de Python diseñada para simplificar la lectura y manipulación de datos ráster en Python. Si bien existe una API en Python de otra librería llamada GDAL (por sus siglas en inglés, Geospatial Data Abstraction Library) (GDAL/OGR contributors, 2025), se optó por utilizar Rasterio debido a su mayor simplicidad de uso con respecto a GDAL, que presentó dificultades durante las evaluaciones iniciales.

Las funciones más relevantes de Rasterio para el proyecto de grado fueron las relacionadas a la lectura y carga en memoria de los DEMs desde archivos con formato GeoTIFF. Para ello se emplea la función `rasterio.open()` que retorna un DataSetReader conteniendo el dato ráster cargado en memoria. La función de lectura `open()` admite varios argumentos que permiten ajustar la lectura del archivo. Por ejemplo, el argumento `nodata` permite especificar el valor que indica que una celda del ráster no contiene datos. También fue de utilidad la clase `Window` importada desde el módulo `rasterio.windows`

que permitió implementar de forma sencilla el recorte de los DEMs con distintos tamaños y solapamientos para los conjuntos de datos. Para generar un crop, se crea un objeto de tipo `Window` con el tamaño de crop deseado y se utiliza como argumento para la función `rasterio.windows.transform`. De esta forma se genera el crop que posteriormente se guarda junto con sus metadatos en un archivo con formato GeoTIFF para su uso durante el entrenamiento o evaluación de modelos.

4.5. Entrenamiento de los modelos

El entrenamiento de las arquitecturas descritas en la Sección 4.2 se realizó en dos etapas. En la primera etapa se identificaron las configuraciones de hiperparámetros que obtuvieron el mejor rendimiento en la métrica MAE para cada arquitectura mediante una búsqueda sistemática en el espacio de hiperparámetros. En la segunda etapa se llevó a cabo un entrenamiento exhaustivo utilizando las configuraciones óptimas obtenidas previamente. Estas dos etapas representan, respectivamente, una exploración del espacio de hiperparámetros y una explotación de las configuraciones óptimas. En esta sección se describen ambas etapas en detalle, se explica el procedimiento seguido en cada una y se introduce para cada arquitectura una variante en el entrenamiento de los modelos que difiere de los entrenamientos originales. Finalmente se describen dos técnicas de aumento de datos aplicadas en el conjunto de entrenamiento: la superposición y rotación de crops.

4.5.1. Entrenamientos A y B

En este proyecto de grado se implementaron dos variantes de entrenamiento para cada modelo. Por un lado, la variante denominada «entrenamiento A» consistió en entrenar con los crops normalizados en todo momento. Por otro lado, la variante «entrenamiento B» consistió en desnormalizar los crops y enmascarar las celdas sin datos para calcular el MSE y MAE de la función de pérdida en el preentrenamiento y el entrenamiento. Para distinguir entre modelos de una misma arquitectura entrenados con métodos distintos, se añade el sufijo *-A* o *-B* al nombre de la arquitectura, en referencia al entrenamiento correspondiente.

Para diferenciar las celdas vacías de las celdas con datos en el entrenamiento A, se estableció el umbral de decisión en -0.3. Los valores inferiores a -0.3 se consideran celdas vacías (se les asigna el valor -1), mientras que los valores iguales o superiores a -0.3 se consideran alturas válidas. Aunque el umbral podría fijarse en 0 de forma intuitiva, las evaluaciones realizadas demostraron que debido a pequeños errores, el modelo con el umbral en 0 clasifica incorrectamente un mayor número de celdas con datos. Para el entrenamiento B el umbral se modificó a -0.1 dado que demostró empíricamente tener mejores resultados respecto a utilizar -0.3.

Optimizar las métricas en crops normalizados y desnormalizados produjo resultados diferentes y esto se explicó con la definición matemática del proceso de desnormalización. El crop «SR» generado por el modelo se desnormaliza utilizando los valores mínimo y máximo del crop en baja resolución «LR» utilizado como entrada. Sin embargo, el crop objetivo «HR» se desnormaliza utilizando sus propios valores de mínimo y máximo, que en general difieren de los correspondientes al crop LR. Sean LR_N, SR_N y HR_N las versiones normalizadas de los crops LR, SR y HR respectivamente. Sean T_{LR} y T_{HR} las transformaciones de desnormalización basadas en los valores de máximo y mínimo de los

crops LR y HR respectivamente. En este contexto, al denotar los pesos del modelo como θ, optimizar la métrica MAE en los crops normalizados implica encontrar el valor de θ^* que satisfaga la Ecuación 4.2.

$$\theta^* = \arg\min_{\theta} \{|SR_N(\theta) - HR_N|\} \tag{4.2}$$

Los crops desnormalizados se expresan como se muestra en la Ecuación 4.3.

$$LR = T_{LR}(LR_N), \quad SR = T_{LR}(SR_N), \quad HR = T_{HR}(HR_N) \tag{4.3}$$

Por lo tanto, la optimización de la métrica MAE con los crops desnormalizados equivale a encontrar el valor de θ^* que cumpla con la Ecuación 4.4.

$$\theta^* = \arg\min_{\theta} \{|SR(\theta) - HR|\} = \arg\min_{\theta} \{|T_{LR}(SR_N(\theta)) - T_{HR}(HR_N)|\} \tag{4.4}$$

El valor de θ^* que satisface la Ecuación 4.2 únicamente coincide con el valor de θ^* que satisface la Ecuación 4.4 cuando los valores de máximo y mínimo del crop LR son iguales a los valores de máximo y mínimo del crop HR respectivamente, lo que en general no ocurre. Por lo tanto optimizar un modelo utilizando los crops normalizados no es siempre equivalente a hacerlo con los crops desnormalizados, dado que las transformaciones T_{LR} y T_{HR} suelen ser distintas. En general, el entrenamiento con crops normalizados produce resultados diferentes al entrenamiento con crops desnormalizados.

Además de las diferencias en la normalización mencionadas en el párrafo anterior, en el entrenamiento A se incluyen las celdas sin datos en el cálculo de las métricas MAE y MSE. Esto da lugar a que errores demasiado altos en las celdas sin datos lleven al modelo a un desbalance en el entrenamiento. Cuando el modelo predice incorrectamente una celda sin datos como si tuviera datos, el error absoluto en esa celda es mayor o igual a 1 (considerando crops normalizados), pues el valor objetivo en una celda sin datos es -1, mientras que el valor inferido se encuentra en el rango de 0 a 1. En contraste, para las celdas con datos, el error máximo ocurre cuando el modelo predice el valor mínimo (0) y el valor real es el máximo (1), o viceversa, con lo que el máximo error posible en valor absoluto es 1.

Por lo tanto, el problema surge al predecir incorrectamente un valor de altura a una celda sin datos, pues el error calculado para esa celda resulta ser mayor que el máximo error posible al predecir celdas con datos. En consecuencia, el modelo asigna una importancia desproporcionadamente alta a la predicción correcta de celdas sin datos, en detrimento de la precisión en las celdas que sí contienen datos. El problema se soluciona utilizando el entrenamiento B, dado que en este enfoque las celdas sin datos no se incluyen en el cálculo de las métricas MAE y MSE.

4.5.2. Búsqueda de hiperparámetros

En la etapa de búsqueda de hiperparámetros el objetivo fue identificar las configuraciones de hiperparámetros que permitieran a los modelos minimizar la métrica MAE en el conjunto de validación. Las arquitecturas SRGAN, SRGAN$_{\text{Z}}$, DSRGAN y ESRGAN fueron entrenadas siguiendo la metodología descrita para cada arquitectura en la Sección 4.2.

En el preentrenamiento de todos los modelos se utilizó un learning rate de 5×10^{-5} para el optimizador del generador. Este valor fue fijado en base a evaluaciones preliminares en las que demostró un mejor desempeño por sobre otros valores de learning rate. Dado que se prefijó el único hiperparámetro del preentrenamiento, la optimización se centró exclusivamente en los hiperparámetros correspondientes al entrenamiento de las arquitecturas. Los hiperparámetros seleccionados para la optimización en el entrenamiento son:

- Tamaño de batch: corresponde al tamaño de batch utilizado en cada iteración del entrenamiento del modelo.
- Learning rate de los optimizadores: se utilizó el algoritmo de optimización Adam tanto para el generador como para el discriminador. El learning rate se trató como un hiperparámetro independiente en ambos optimizadores.
- Pesos de los componentes de las funciones de pérdida: pesos con los que se ponderaron la content loss, adversarial loss, MAE loss y MSE loss dentro de la función de pérdida final de cada modelo. Estos pesos fueron considerados como hiperparámetros a optimizar en las arquitecturas en las que correspondía.
- Tamaño de crop: dimensión de los crops utilizados durante el entrenamiento. Entrenar con un tamaño de crop más pequeño permite que la red aprenda características locales, mientras que un tamaño mayor de crop puede ayudar a capturar características globales, aunque también añade complejidad al entrenamiento de la red.

Para cada hiperparámetro se definió un rango de valores a utilizar por el software Optuna durante la optimización. Optuna eligió los valores para cada hiperparámetro dentro de los rangos definidos y luego de 50 combinaciones diferentes identificó la mejor combinación de hiperparámetros. Para que una combinación se considere mejor que el resto, el modelo entrenado con dichos hiperparámetros debe alcanzar un MAE menor en el conjunto de validación en comparación con el resto de las configuraciones probadas. La Tabla 4.2 muestra los rangos de valores utilizados para cada hiperparámetro. Los extremos de los rangos se seleccionaron basándose en los valores de la implementación original para cada arquitectura, agregando un margen de exploración adicional. La única excepción fue el peso asociado al MAE loss, un componente no incluido en las implementaciones originales. Para el peso asociado al MAE loss se definió un rango similar al peso asociado al MSE loss, pero con un extremo superior restringido para evitar que la métrica MAE predomine sobre los demás términos de la función de pérdida.

En la búsqueda de hiperparámetros, dos de los hiperparámetros recibieron un tratamiento diferenciado: el tamaño de crop y el peso de la métrica MAE en la función de pérdida. Por un lado, utilizar diferentes tamaños de crops como hiperparámetro implicó que los conjuntos de entrenamiento fueran diferentes, por lo que se realizó una optimización específica para cada tamaño (16×16, 24×24 y 32×32). Por otro lado, la métrica MAE no estaba incluida como componente de la función de pérdida en las arquitecturas originales. Sin embargo, debido a la mejora en el rendimiento observada en evaluaciones preliminares, se decidió realizar optimizaciones adicionales incluyendo la métrica MAE como componente de la función de pérdida que, como se mencionó en la Sección 4.2, la incorporación de este componente conformó a los denominados modelos personalizados.

hiperparámetro	rango / valores
tamaño de batch	[8, 16, 32]
learning rate del generador	$(1 \times 10^{-8}, 1 \times 10^{-3})$
learning rate del discriminador	$(1 \times 10^{-8}, 1 \times 10^{-3})$
peso de la content loss	$(1 \times 10^{-5}, 1 \times 10^{-2})$
peso de la adversarial loss	$(1 \times 10^{-5}, 1 \times 10^{-2})$
peso de la MSE loss	$(1 \times 10^{-5}, 1)$
peso de la MAE loss	$(1 \times 10^{-5}, 5 \times 10^{-1})$
tamaño de los crops	[16×16, 24×24, 32×32]

Tabla 4.2: Rangos de valores utilizados para la optimización de hiperparámetros.

Para cada uno de los tamaños de crops se llevaron a cabo dos búsquedas de hiperparámetros diferentes: una para el modelo original y otra para el modelo personalizado, lo que resultó en un total de seis optimizaciones de hiperparámetros por modelo. Adicionalmente, se realizó una optimización utilizando el entrenamiento B mencionado en la sección 4.5.1 para cada caso. Por lo tanto, se realizaron seis optimizaciones de hiperparámetros por modelo con entrenamiento A y otras seis con entrenamiento B, incrementando a doce el número total de optimizaciones realizadas por cada arquitectura.

Para cada optimización en las arquitecturas SRGAN, DSRGAN y ESRGAN, se realizaron 50 intentos con distintas combinaciones de hiperparámetros. Para cada combinación de hiperparámetros se realizaron 15 épocas de preentrenamiento y 25 épocas de entrenamiento, con el objetivo de equilibrar explotación, exploración y tiempo total de búsqueda. Los crops utilizados para entrenar los modelos durante la búsqueda de hiperparámetros no presentaron solapamiento con el propósito de balancear el tiempo de entrenamiento y la información contenida en los datos. En el caso de SRGAN_Z, se probaron 30 combinaciones de hiperparámetros con 15 épocas de entrenamiento cada una. Para SRGAN_Z se probaron menos combinaciones y se entrenaron menos épocas que en las demás arquitecturas pues se trata de un modelo ya entrenado y la demora en cada época fue significativamente mayor a la demora de cada época en las demás arquitecturas.

4.5.3. Entrenamiento en profundidad

Una vez finalizada la optimización de hiperparámetros, se realizó un entrenamiento en profundidad con el modelo que obtuvo mejores resultados para cada arquitectura. El entrenamiento en profundidad para las arquitecturas SRGAN, ESRGAN y DSRGAN consistió en 15 épocas de preentrenamiento del generador seguidas de 100 épocas de entrenamiento de la GAN. En el caso de la arquitectura SRGAN_Z no se realizaron las 15 épocas de preentrenamiento, dado que el modelo utilizado contaba con un entrenamiento previo de 100 épocas y los tiempos de ejecución fueron significativamente mayores respecto al resto de las arquitecturas. Para simplificar la nomenclatura, desde este punto se denomina entrenamiento al proceso completo, que incluye las fases de preentrenamiento y entrenamiento del modelo. Con el fin de tener significancia estadística en los resultados y de aprovechar al máximo los modelos, cada entrenamiento se repitió cinco veces.

4.5.4. Aumento de datos

Agregar nuevos datos al conjunto de entrenamiento puede mejorar el rendimiento de los modelos. Sin embargo, cuando no se dispone de nuevos datos o son muy costosos de obtener, es posible aplicar técnicas para generar datos artificialmente y así aumentar la diversidad del conjunto de entrenamiento. Las técnicas de aumento de datos se aplican sobre los datos existentes con el objetivo de crear nuevas muestras de forma artificial.

En este proyecto de grado se utilizaron dos técnicas de aumento de datos. La primera técnica se basó en la aplicación de rotaciones (en múltiplos de 90 grados) y volteos horizontales o verticales sobre los crops, manteniendo su estructura original. Con esta primera técnica se buscó que el modelo fuera capaz de aprender patrones independientemente de la posición específica dentro del crop. La segunda técnica consistió en superponer los crops del conjunto de entrenamiento. Con esta técnica se lograron incluir patrones ubicados en los bordes de los crops que, de otro modo, quedarían separados por la división entre ellos. En la segunda técnica se utilizó para los crops un stride (explicado en la Sección 4.3.2) del 50 % del tamaño del crop.

La segunda técnica de aumento de datos implicó un incremento en la cantidad de instancias en el conjunto de entrenamiento debido a que se almacenaron los crops superpuestos. Para mantener una cantidad comparable de iteraciones con respecto a los entrenamientos anteriores, se redujo proporcionalmente el número de épocas de preentrenamiento y entrenamiento. La cantidad de épocas se determinó durante el diseño del experimento, en función exclusivamente del tamaño de crop seleccionado para el entrenamiento con esta técnica.

Una vez fijado el tamaño de crop, la cantidad de subcrops C que se generan sin aplicar stride, sobre un crop base de 96×96 celdas se calcula mediante la Ecuación 4.5, donde TC representa el tamaño del crop seleccionado.

$$C = \left(\frac{96}{TC}\right)^2 \tag{4.5}$$

Al aplicar un stride del 50 %, la cantidad de subcrops dentro del mismo crop base CS aumenta y se calcula mediante la Ecuación 4.6.

$$CS = \left(\left(\frac{96}{TC}\right) \cdot 2 - 1\right)^2 \tag{4.6}$$

El factor de incremento en la cantidad de subcrops generados, denotado por Z, se obtiene con la Ecuación 4.7.

$$Z = \frac{CS}{C} \tag{4.7}$$

Finalmente, para mantener la equivalencia en el número total de iteraciones, se ajustaron las épocas de entrenamiento dividiéndolas por este factor Z. Así, el número de épocas de preentrenamiento con stride se redujo a $\frac{15}{Z}$ y el número de épocas de entrenamiento a $\frac{100}{Z}$.

4.6. Evaluación de los modelos

Para la evaluación de los modelos utilizados en este proyecto de grado se presentaron los resultados de las métricas seleccionadas en el conjunto de validación y se compararon con los resultados obtenidos al evaluar los métodos tradicionales de interpolación mencionados en la Sección 2.1.2. Además, con el objetivo de seleccionar los mejores modelos dentro de los evaluados, se realizó una comparación entre ellos.

4.6.1. Método de ventana deslizante

Para realizar la comparación entre los modelos y los métodos de interpolación, se utilizó un conjunto de datos que permitió evaluar cada modelo de manera independiente del tamaño de crop con el cual fue entrenado. La metodología utilizada para recortar los crops inicialmente garantizó la preservación de información en cada conjunto de datos (entrenamiento, validación y evaluación) para diferentes tamaños de crop. En la evaluación y comparación de los modelos se decidió emplear los crops base de tamaño 96×96, mencionados en la Sección 4.3.2

El uso de los crops base de tamaño 96×96 implicó que los modelos aplicaran una técnica de ventana deslizante para generar subcrops, ya que los modelos se entrenaron con crops más pequeños. Además, utilizar crops de tamaño 96×96 brindó un escenario de evaluación más realista, debido a que los DEMs completos generalmente poseen dimensiones superiores a los crops utilizados durante el entrenamiento. El procedimiento empleado para evaluar cada modelo fue idéntico al proceso final aplicado a un DEM completo: dividir el crop base en subcrops, realizar la superresolución por separado en cada uno de ellos, y luego reconstruir el crop base con la resolución aumentada.

La técnica de ventana deslizante genera subcrops con superposición entre crops contiguos de forma horizontal o vertical. Como consecuencia, algunas celdas del DEM original reciben más de una predicción debido a su inclusión en múltiples subcrops. Para resolver la existencia de múltiples predicciones por celda, cada subcrop se multiplicó por una matriz de pesos. En esta matriz, las celdas cercanas al centro poseen pesos mayores en comparación con aquellas ubicadas cerca de los bordes debido a que en el centro del subcrop el modelo cuenta con mayor información contextual. Luego de multiplicar los subcrops por la matriz de pesos, se realizó una suma ponderada de los valores predichos para cada celda del crop base. Finalmente, el valor obtenido se dividió entre la suma total de los pesos aplicados. La matriz de pesos se calculó utilizando la Ecuación 4.8.

$$w_{i,j} = 1 - \frac{\sqrt{(i-c)^2 + (j-c)^2}}{\sqrt{2} \cdot c} \tag{4.8}$$

En la Ecuación 4.8, i y j indican la fila y columna de la celda dentro del subcrop respectivamente y c representa la mitad del tamaño del subcrop. Esta ecuación asigna un peso menor a las celdas alejadas del centro.

La Figura 4.4 muestra un ejemplo del método descrito en el párrafo anterior. En la figura se presenta un crop base con líneas negras indicando divisiones de subcrops de tamaño 32×32 con superposición de 16 celdas. Dos subcrops están resaltados en naranja y rojo, indicando mayor intensidad de color según la matriz de pesos. El punto negro situado en el borde del subcrop naranja y cerca del centro del subcrop rojo, recibe mayor influencia de la predicción realizada en el subcrop rojo respecto a la predicción realizada en el subcrop naranja.

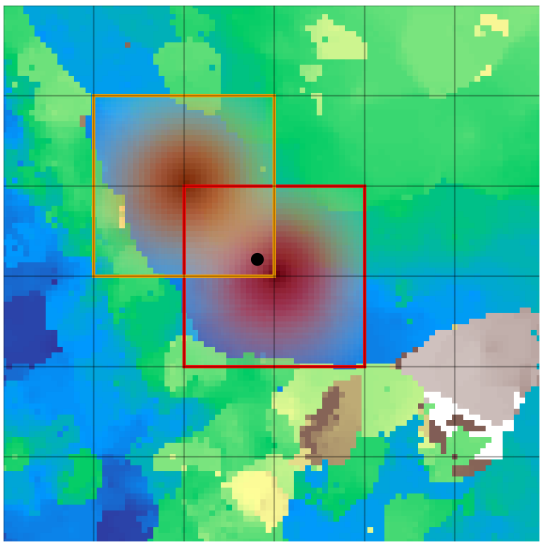


Figura 4.4: Representación gráfica de dos crops con su respectiva matriz de pesos.

4.6.2. Métricas

Para evaluar los resultados de los modelos seleccionados se adoptó un enfoque cuantitativo en el que se utilizaron las métricas que han sido tendencia en los trabajos del área de superresolución mencionados en la Sección 3. En la búsqueda de hiperparámetros se calcularon las métricas MAE, RMSE y LE90 para seleccionar los mejores modelos. Adicionalmente se reportaron las métricas max error, precision y recall.

Las métricas precision y recall se incorporaron al conjunto de métricas utilizadas debido a que los DEMs, a diferencia de las imágenes, tienen celdas vacías. La precision y el recall permiten evaluar el rendimiento del modelos tomando en cuenta las celdas vacías. Una precision baja significa que muchas celdas sin datos fueron clasificadas erróneamente como celdas con datos. Un recall bajo significa que hay muchas celdas sin datos que no fueron clasificadas como tal y en cambio se les asignó un valor de altura. Mantener valores superiores en ambas métricas en comparación con las interpolaciones clásicas supone un desafío adicional al realizar superresolución sobre DEMs.

Debido a la poca diferencia de elevación en el territorio uruguayo las métricas geográficas no aportan información relevante por lo que se decidió no implementarlas. En el caso de las redes de drenaje, no es adecuado incluir la evaluación de estas en el territorio uruguayo dado que en terrenos de baja elevación o en DEMs pequeños no se pueden identificar con certeza las redes de drenaje.

4.7. Entrenamiento de los modelos en ClusterUY

En general, el entrenamiento de modelos de aprendizaje automático requiere una capacidad de cómputo elevada. Para el entrenamiento de redes neuronales, el uso de GPUs permite paralelizar los cálculos y, en consecuencia, acelerar el entrenamiento. Por esta razón, se utilizó la plataforma de computación de alto desempeño Centro Nacional de Supercomputación (ClusterUY) (Nesmachnow e Iturriaga, 2019).

ClusterUy cuenta con una capacidad equivalente a 10,000 computadoras tradicionales y los recursos que ofrece a los usuarios son:

- 2,240 núcleos de cómputo CPU de los cuales 1,120 son núcleos Intel Xeon-Gold 6138 2.00GHz, 96 son núcleos AMD EPYC 7642 2.30GHz y 1,024 son AMD EPYC 7763 3.50Ghz
- 6 TB de memoria RAM
- 100,352 núcleos de cómputo GPU Nvidia Tesla P100 con 12Gb de memoria, Nvidia Ampere A100 de 40Gb y Nvidia A40 de 48G interconectados por una red de alta velocidad Ethernet de 10 Gbps

Para la ejecución de los programas en ClusterUY, se crean trabajos que ejecutan scripts en shell, donde se especifican los recursos requeridos y los comandos a ejecutar. Al ejecutar un trabajo, ClusterUY asigna un nodo con los recursos solicitados en el script, permitiendo la ejecución de la tarea. En los entrenamientos de las redes neuronales se utilizaron los tres tipos de GPUs disponibles: NVIDIA P100, NVIDIA A100 y NVIDIA A40 además de 16GB de almacenamiento rápido. Los administradores de ClusterUY recomiendan utilizar el almacenamiento rápido temporal para manejar grandes volúmenes de datos, ya que el almacenamiento predeterminado es considerablemente más lento. Para hacer uso del espacio de almacenamiento temporal, es necesario transferir los datos a utilizar durante el entrenamiento desde el almacenamiento predeterminado. Una vez finalizado el trabajo, el espacio de almacenamiento temporal reservado es eliminado automáticamente. Dado el gran volumen de datos disponible para el entrenamiento y la baja velocidad de escritura y lectura en el almacenamiento predeterminado, se optó por comprimir los conjuntos de crops en un archivo ZIP. Cada vez que se utilizaban los archivos para un entrenamiento, se transferían desde el almacenamiento predeterminado al almacenamiento temporal donde se descomprimían. Dado que el almacenamiento rápido es temporal, los resultados generados por los programas se guardaron en el almacenamiento predeterminado.

Capítulo 5

Análisis experimental

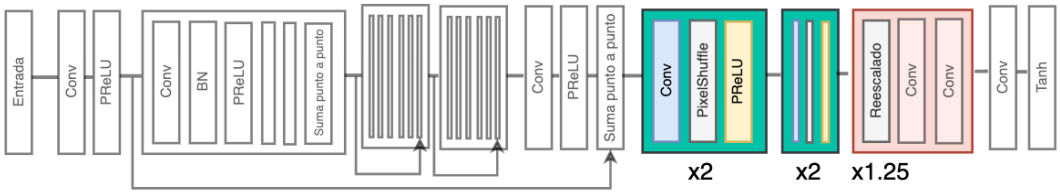
En este capítulo se exponen y analizan los resultados de los experimentos realizados. En primer lugar se detallan las adaptaciones implementadas en el generador de cada modelo. Luego se presentan y se comparan los resultados del proceso de búsqueda de hiperparámetros y del entrenamiento final de cada modelo. Finalmente, se describen las ejecuciones realizadas con la técnica de aumento de datos aplicada al conjunto de entrenamiento.

5.1. Adaptación del generador

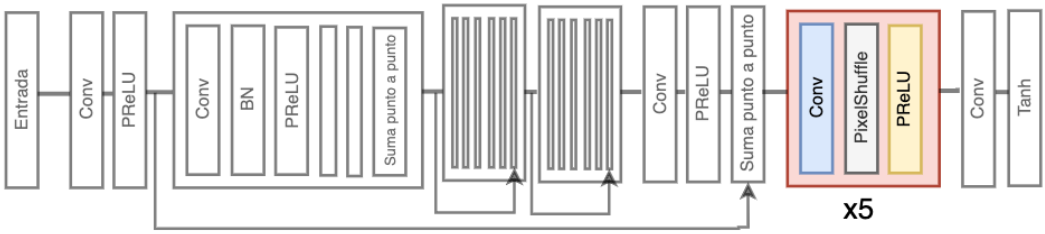
En este proyecto de grado se adaptaron las arquitecturas propuestas en la Sección 4.2 para alcanzar el incremento de resolución requerido. Como se indicó en la Sección 4.1 se necesitó aumentar cinco veces la resolución, por lo que la salida de cada modelo debe tener cinco veces la resolución de la entrada. Esta restricción generó una dificultad dado que los generadores evaluados utilizaban dos upsampling blocks, cada uno encargado de duplicar la resolución de la entrada. Al emplear únicamente dos upsampling blocks, la salida final tenía una resolución cuatro veces mayor que la entrada.

Se identificaron dos posibles soluciones al problema de la diferencia entre el factor de escalado de los generadores originales y el factor de escalado requerido por el proyecto de grado. La primera solución propuesta consistió en concatenar al generador original una capa de aumento de escala (upscaling) encargada de incrementar la resolución por un factor de 1,25, alcanzando así la ampliación total de $\times 5$. La capa de upscaling recibe como entrada la salida de los upsampling blocks y luego de la capa de upscaling se agregaron dos capas convolucionales adicionales para ajustar las características obtenidas tras el escalado de factor 1,25. La segunda solución al problema de la diferencia en el factor de escalado consistió en eliminar los upsampling blocks originales y utilizar un único upsampling block modificado para incrementar la resolución en un factor de cinco. La Figura 5.1 presenta gráficamente ambas soluciones aplicadas a la arquitectura SRGAN.

Para determinar cuál de las dos soluciones era la más adecuada se implementó cada solución en una variante de la red SRGAN, pues es la arquitectura base de las arquitecturas evaluadas. Se entrenó cada variante con diez épocas de preentrenamiento y treinta épocas adicionales de entrenamiento manteniendo la combinación de hiperparámetros utilizada en la red SRGAN original. Para decidir cuál de las dos variantes presentaba mejores resultados se evaluaron las métricas cuantitativas MAE, RMSE, max error y LE90.



(a) Upsample block de 1.25



(b) Upsample block de 5

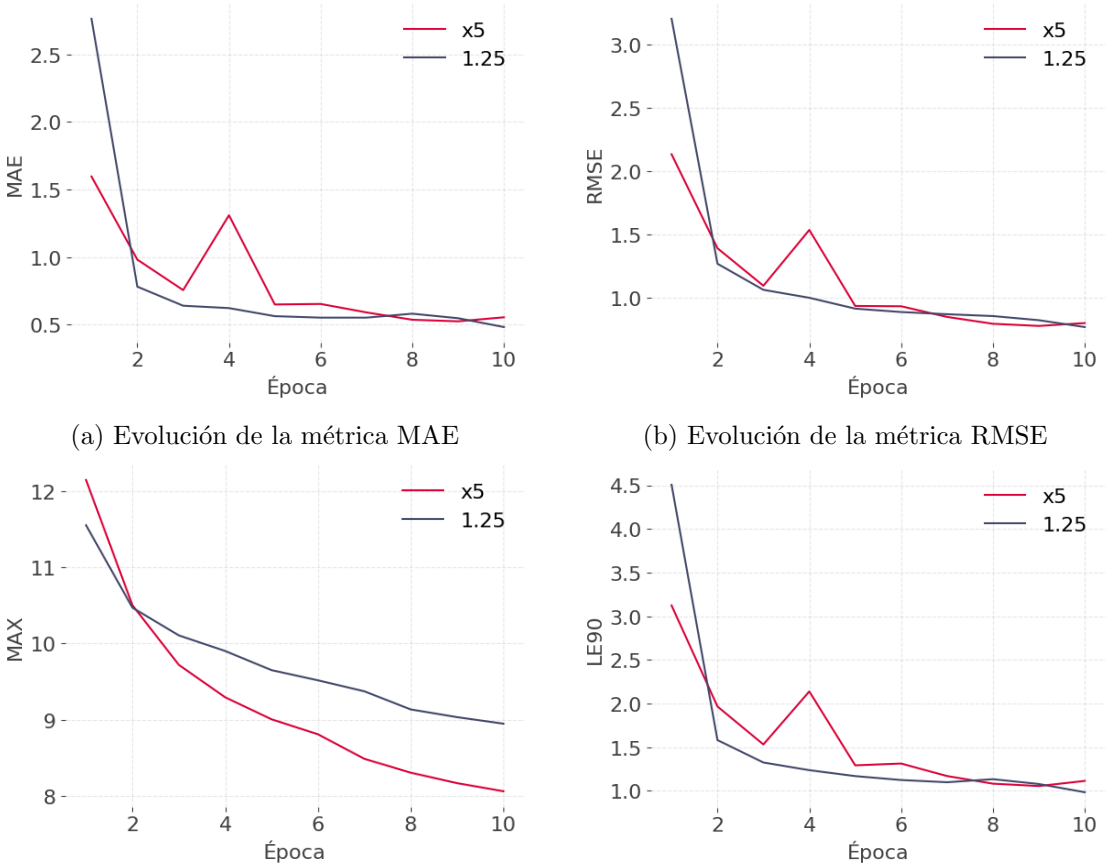
Figura 5.1: Representación de dos arquitecturas de generador propuestas.

La Figura 5.2 presenta los valores de las métricas MAE, RMSE, max error y LE90 obtenidos en el conjunto de validación durante el preentrenamiento para ambas variantes de SRGAN.

Ambas variantes de SRGAN mostraron un comportamiento similar en las métricas MAE, RMSE y LE90. Sin embargo, la variante con un único upsampling block con escalado $\times 5$ presentó un 10,1 % menos de error en la métrica max error.

La Figura 5.3 muestra los resultados obtenidos de las treinta épocas adicionales de entrenamiento posteriores al preentrenamiento.

Ambas variantes presentaron resultados similares en las métricas MAE, RMSE, max error y LE90. Sin embargo, la variante con un único upsampling block $\times 5$ obtuvo los valores más bajos en cada métrica. Si se considera el menor valor alcanzado por cada variante durante el entrenamiento, la variante con un único upsampling block $\times 5$ obtuvo una mejora de 6,0 % en MAE, 3.6 % en RMSE, 3.6 % en max error y 4.1 % en LE90. Se decidió por lo tanto implementar un único upsampling block con factor de escalado $\times 5$ para todas las arquitecturas evaluadas en el proyecto de grado dado que es la solución que presentó mejores resultados.



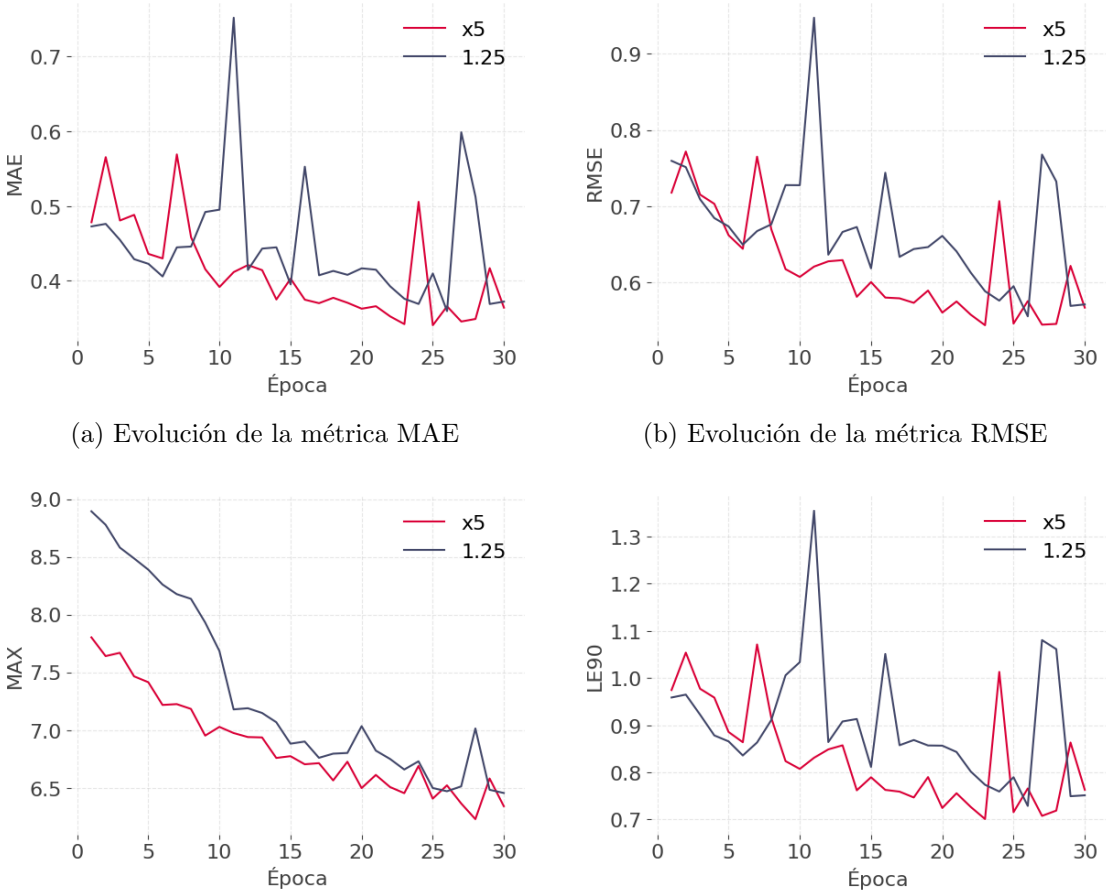
(a) Evolución de la métrica MAE

(b) Evolución de la métrica RMSE

(c) Evolución de la métrica max error

(d) Evolución de la métrica LE90

Figura 5.2: Métricas de evaluación del preentrenamiento de la SRGAN (10 épocas) con distintas configuraciones de upsample blocks.



(a) Evolución de la métrica MAE

(b) Evolución de la métrica RMSE

(c) Evolución de la métrica max error

(d) Evolución de la métrica LE90

Figura 5.3: Métricas de evaluación para el entrenamiento de 30 épocas para SRGAN con distintas configuraciones de upsample blocks.

5.2. Búsqueda de hiperparámetros

En esta sección se presenta un análisis de tiempos de entrenamiento para los modelos estudiados y los resultados de las evaluaciones de los mejores modelos tras las optimizaciones realizadas, junto con un análisis comparativo basado en la metodología descrita en la Sección 4.6.

5.2.1. Organización de los resultados

Para cada arquitectura evaluada, los resultados se organizaron según entrenamiento A o entrenamiento B que fueron descritos en la Sección 4.5. Para cada tipo de entrenamiento, se reportaron los modelos con mejores resultados entrenados con cada tamaño de crop (16×16, 24×24 y 32×32) tanto en la variante del modelo personalizada como en el modelo original, lo que resultó en 6 modelos por tipo de entrenamiento. Para cada uno de los mejores modelos se reportaron los hiperparámetros utilizados y los valores de las métricas evaluadas en el conjunto de validación.

En las tablas de hiperparámetros los valores en las columnas w_{ADV}, w_{CONT}, w_{MAE} y w_{MSE} corresponden a los pesos asignados a los componentes adversarial loss, content loss, MAE loss y MSE loss de la función de pérdida del generador respectivamente. Las columnas LR_D y LR_G corresponden a los valores de los learning rates utilizados para el discriminador y el generador respectivamente.

Para visualizar el rendimiento de las distintas combinaciones de hiperparámetros se presentó un resumen de los resultados obtenidos durante las optimizaciones en un diagrama de caja (boxplot) para cada arquitectura y tipo de entrenamiento. Cada boxplot corresponde al MAE final de todos los intentos de combinaciones de hiperparámetros obtenidos luego de entrenar cada modelo con el tamaño de crop indicado en el eje vertical. La variante del modelo personalizado u original se indica en el boxplot mediante colores en la leyenda y separados con una línea punteada.

Adicionalmente, para cada arquitectura y tipo de entrenamiento se incluyó una tabla que presenta un crop en baja resolución, su versión en alta resolución y sus versiones superresueltas por cada modelo con el fin de evaluar la calidad visual de los resultados. El crop seleccionado contiene cambios abruptos de altura que producen bordes finos en la representación visual y permiten observar las diferencias visuales entre los crops generados por los modelos. El crop seleccionado tiene un tamaño de 48×48 en baja resolución, lo que requiere que cada modelo aplique la técnica de ventana deslizante para generar su versión superresuelta. Para cada crop superresuelto se reportan las métricas MAE y RMSE.

Se utilizaron tres métricas principales para seleccionar los mejores modelos: MAE, RMSE y LE90 en orden de importancia. En casos donde estas métricas resultaron similares entre dos modelos, se utilizó el max error, la precision y el recall como criterios adicionales.

5.2.2. Análisis de tiempos de entrenamiento

En esta sección se presenta un análisis comparativo del tiempo de entrenamiento requerido por cada una de las arquitecturas evaluadas. Se evaluó la duración de una época de entrenamiento para cada modelo utilizando diferentes tamaños de crop, con el objetivo de cuantificar de manera precisa el costo temporal asociado a cada configuración.

Las ejecuciones se realizaron con un tamaño de batch de ocho, que representa el caso más lento en el proceso de entrenamiento, permitiendo así obtener el tiempo en el peor escenario.

Como se mencionó en la Sección 4.5.2, en la búsqueda de hiperparámetros del modelo de la arquitectura SRGAN_Z (Zhang y Yu, 2022) se optó por entrenar un menor número de épocas por intento y evaluar un menor número de combinaciones de hiperparámetros en comparación con los modelos de las demás arquitecturas. Esta decisión se debió a que el modelo demandó un tiempo de cómputo significativamente mayor en comparación con ESRGAN, SRGAN y DSRGAN.

La Tabla 5.1 muestra la duración en segundos del entrenamiento de una época para cada arquitectura evaluada y en cada tamaño de crop utilizado. Para cada configuración se realizaron diez mediciones independientes utilizando una GPU NVIDIA A40.

arquitectura	tamaño de crop	tiempo mínimo (s)	tiempo máximo (s)	tiempo promedio (s)	desviación estándar
SRGAN	16	208.47	241.94	216.18	11.29
	24	144.06	147.23	145.72	0.96
	32	122.20	124.58	122.85	0.67
ESRGAN	16	609.34	738.54	648.84	42.79
	24	338.61	347.96	341.11	2.50
	32	267.58	269.46	268.09	0.52
DSRGAN	16	267.33	275.04	270.68	2.47
	24	204.43	206.44	205.55	0.72
	32	180.43	182.43	181.72	0.56
SRGAN_Z	16	1305.79	1310.49	1307.37	1.27
	24	1290.81	1294.05	1292.58	0.99
	32	1267.26	1274.74	1268.64	2.07

Tabla 5.1: Duración promedio en segundos de una época de entrenamiento para cada modelo probado.

La Tabla 5.1 muestra que la arquitectura más rápida en todos los tamaños de crop utilizados para entrenar fue SRGAN. La segunda arquitectura más rápida fue DSRGAN que al compararla con SRGAN resultó ser 28 % más lenta en crops de 16×16, 41 % más lenta en crops de 24×24 y 47 % más lenta en crops de 32×32. ESRGAN fue 193 % más lenta que SRGAN en crops de 16×16, 135 % más lenta en crops de 24×24 y 118 % más lenta en crops de 32×32. Por lo tanto, ESRGAN requirió hasta 3 veces más tiempo de entrenamiento que SRGAN. El modelo SRGAN_Z presentó los mayores tiempos de entrenamiento entre todas las arquitecturas evaluadas. Respecto a SRGAN, SRGAN_Z fue 526 % más lento en crops de 16×16, 796 % más lento en crops de 24×24 y 937 % más lento en crops de 32×32.

Una razón por la que se decidió entrenar menos épocas en la búsqueda de hiperparámetros en los modelos de la arquitectura SRGAN_Z fue que realizar 25 épocas de entrenamiento y 50 intentos durante la búsqueda de hiperparámetros para esta arquitectura utilizando crops de 16×16, habría implicado en el peor de los casos un total de

$1,305 \times 25 \times 50 = 1,631,250$ segundos (equivalentes a 18.9 días) de cómputo en una GPU A40. Esta estimación se refiere únicamente a un tamaño de crop, sin considerar que dicha operación debía repetirse para cada uno de los tamaños evaluados. Además, a diferencia de las demás arquitecturas, en SRGAN_Z el aumento del tamaño del crop no produjo una disminución significativa en el tiempo de entrenamiento por época, como se evidenció en la Tabla 5.1. Por lo tanto, para la búsqueda de hiperparámetros en SRGAN_Z, se decidió entrenar un número de épocas que logró un equilibrio entre el tiempo de entrenamiento y la calidad de resultados.

5.2.3. Resultados obtenidos con SRGAN

En esta subsección se presentan los resultados obtenidos en la búsqueda de hiperparámetros con la arquitectura SRGAN. Se reportan los resultados correspondientes al entrenamiento A seguidos por los resultados obtenidos con el entrenamiento B.

Entrenamiento A

La Figura 5.4 presenta los boxplots correspondientes al MAE para los modelos SRGAN-A. La Tabla 5.2 muestra los resultados de los mejores modelos de cada optimización con SRGAN-A y la Tabla 5.3 detalla los hiperparámetros asociados. Dado que la métrica MAE es añadido a la función de pérdida únicamente para los modelos personalizados, no se incluye como hiperparámetro en los modelos originales.

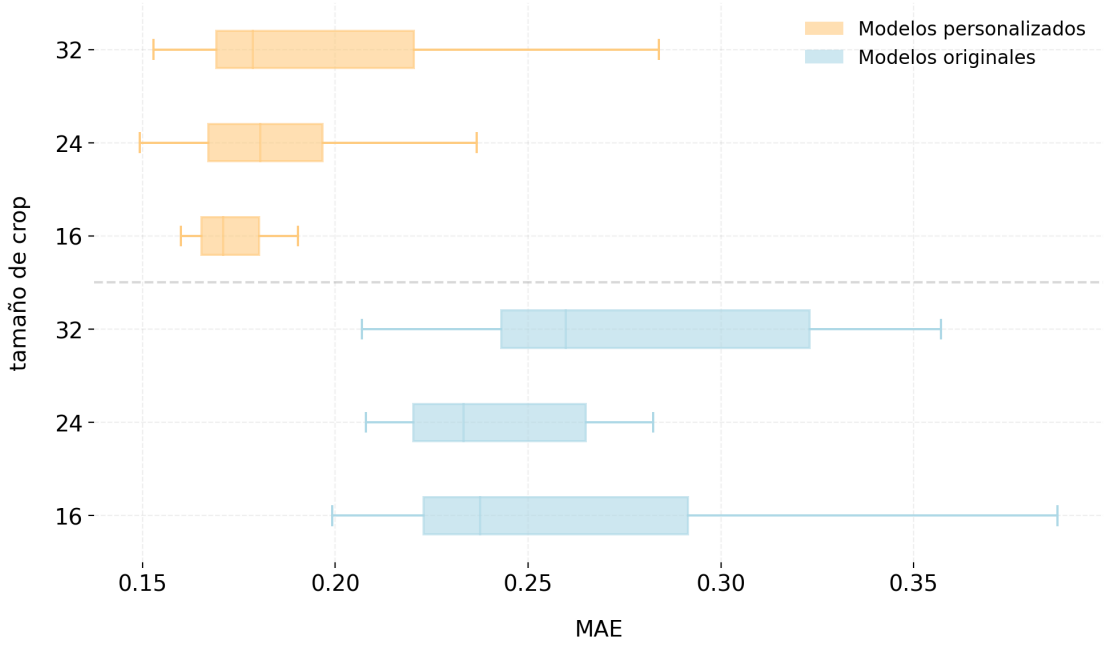


Figura 5.4: Boxplots de los resultados de las optimizaciones con SRGAN-A

La Figura 5.4 muestra la superioridad de los modelos personalizados frente a los modelos originales en la métrica MAE. Entre los modelos personalizados, el modelo entrenado con crops de tamaño 24×24 obtuvo la mediana más alta y el límite inferior más bajo. Además, la mediana del modelo personalizado entrenado con crops de tamaño

24×24 resultó más centrada respecto a los demás modelos, lo que sugiere que los valores de MAE obtenidos durante la optimización de este modelo se distribuyeron de forma más uniforme.

modelo	tamaño de crop	MAE	RMSE	max error	LE90	precision	recall
original	16	0.1715	0.3538	8.6029	0.3110	0.9948	**0.9995**
	24	0.1811	0.3552	**8.0689**	0.3140	0.9956	0.9994
	32	0.1802	0.3662	8.1803	0.3221	**0.9957**	0.9991
personalizado	16	0.1387	0.3378	8.8233	0.2347	0.9947	0.9991
	24	**0.1323**	0.3463	9.0198	**0.2158**	0.9952	0.9991
	32	0.1373	**0.3368**	8.4471	0.2299	0.9949	0.9991

Tabla 5.2: Mejores modelos de optimización de hiperparámetros con SRGAN-A.

La Tabla 5.2 muestra resultados consistentes con los boxplots presentados en la Figura 5.4. Entre los modelos personalizados el que presentó mejores resultados fue el modelo entrenado con crops de 24×24, dado que obtuvo los menores valores para las métricas MAE y LE90. Dentro de los modelos originales se destaca como mejor modelo el entrenado con crops de 16×16, dado que obtuvo los menores valores para las métricas MAE, RMSE y LE90.

El mejor modelo personalizado de SRGAN-A en comparación al mejor modelo original de SRGAN-A mostró una reducción del 24.2 % en MAE, del 2.0 % en RMSE y del 31.8 % en LE90. En max error, precision y recall los modelos originales mostraron en promedio un mejor resultado comparado con los modelos personalizados. Dados los resultados obtenidos en las tres métricas principales, se consideraron los modelos personalizados de SRGAN-A superiores a los modelos originales de SRGAN-A.

modelo	tamaño de crop	tamaño de batch	LR_D	LR_G	w_{ADV}	w_{CONT}	w_{MAE}	w_{MSE}
original	16	8	1.1×10^{-6}	4.8×10^{-4}	1.1×10^{-4}	2.2×10^{-5}	-	4.2×10^{-1}
	24	16	3.5×10^{-8}	1.5×10^{-6}	3.9×10^{-4}	4.1×10^{-4}	-	2.3×10^{-1}
	32	8	4.8×10^{-5}	3.0×10^{-8}	2.8×10^{-5}	1.0×10^{-3}	-	1.0×10^{-2}
personalizado	16	8	6.6×10^{-6}	7.2×10^{-8}	1.2×10^{-4}	4.5×10^{-4}	2.3×10^{-1}	5.2×10^{-2}
	24	16	3.0×10^{-6}	2.3×10^{-4}	1.4×10^{-4}	2.5×10^{-3}	4.2×10^{-1}	7.6×10^{-3}
	32	8	9.6×10^{-8}	1.1×10^{-8}	4.2×10^{-5}	7.6×10^{-4}	4.8×10^{-1}	1.0×10^{-5}

Tabla 5.3: Mejores hiperparámetros para SRGAN-A.

Respecto a los hiperparámetros de los modelos reportados en la Tabla 5.3, se extrajeron tres conclusiones relevantes. En primer lugar, los tamaños de batch más pequeños resultaron en un mejor rendimiento en términos de MAE, dado que en cuatro de las seis optimizaciones realizadas el mejor tamaño de batch encontrado fue de 8, mientras que en las dos restantes fue de 16. Ningún modelo presentó un tamaño de batch de 32 como el óptimo encontrado. En segundo lugar, el peso asociado a la MSE loss disminuyó

a medida que aumentó el tamaño de crop utilizado durante el entrenamiento. Además, el peso asociado a la MSE loss predominó en los modelos originales, mientras que en los modelos personalizados predominó el peso asignado a la MAE loss. Finalmente, en los modelos personalizados el peso asociado a la MAE loss aumentó conforme lo hizo el tamaño de crop utilizado en el entrenamiento del modelo.

La Tabla 5.4 muestra el crop seleccionado generado por cada uno de los modelos SRGAN analizados.

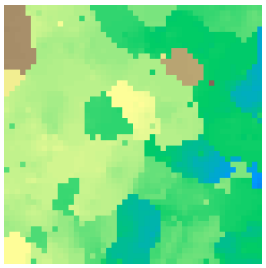
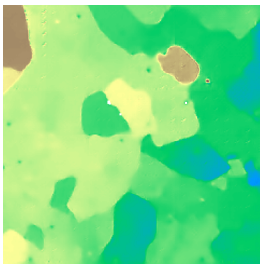
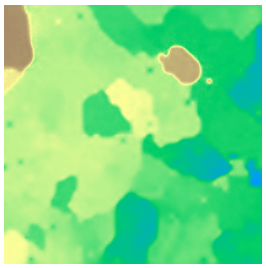
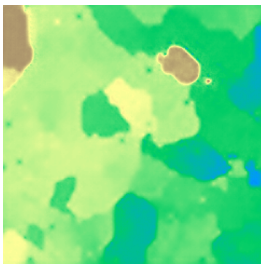
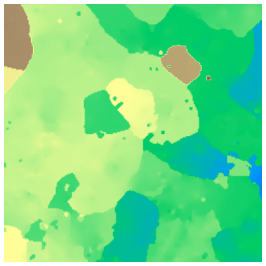
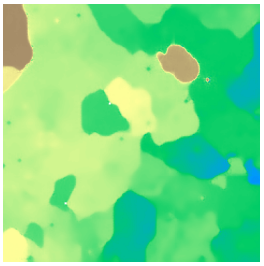
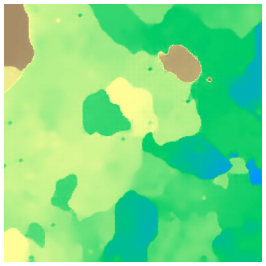
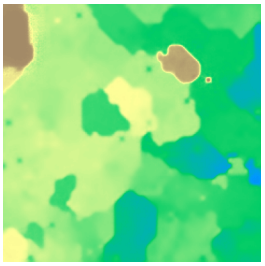
baja resolución	16×16 original	24×24 original	32×32 original
			
	MAE: 0.3783 RMSE: 0.8149	MAE: 0.3585 RMSE: 0.7734	MAE: 0.3720 RMSE: 0.7909
alta resolución	**16×16 personalizado**	**24×24 personalizado**	**32×32 personalizado**
			
	MAE: 0.2628 RMSE: 0.7813	MAE: 0.2681 RMSE: 0.8270	MAE: 0.2806 RMSE: 0.7800

Tabla 5.4: Comparación de un crop generado por los seis modelos reportados de SRGAN-A.

Los modelos SRGAN con entrenamiento A presentaron patrones con artefactos en los crops generados que fueron más frecuentes en los modelos originales. Por ejemplo, el crop generado por el modelo original de 16×16 presentado en la Tabla 5.4 presentó un patrón repetitivo en la parte superior izquierda. Este patrón se clasifica como artefacto ya que no hay indicios en el crop de baja resolución que justifiquen su existencia. En contraste, los modelos originales de 24×24 y 32×32 no presentaron artefactos tan evidentes. Los modelos personalizados, en comparación con los originales, destacaron por su capacidad de reproducir bordes finos con mayor precisión, lo que fue particularmente notorio en los modelos personalizados de 16×16 y 24×24. Los crops generados por ambos modelos de 16×16 presentaron algunas celdas sin datos, un error que no presentaron los crops generados por los modelos de 24×24 y 32×32. Al igual que en los resultados presentados en la Tabla 5.2, la Tabla 5.4 muestra la superioridad de los modelos personalizados en la métrica MAE con respecto a los originales.

Entrenamiento B

La Figura 5.5 presenta los boxplots correspondientes al MAE para los modelos SRGAN-B. La Tabla 5.5 muestra los resultados de los mejores modelos de cada optimización con SRGAN-B y la Tabla 5.6 detalla los hiperparámetros asociados.

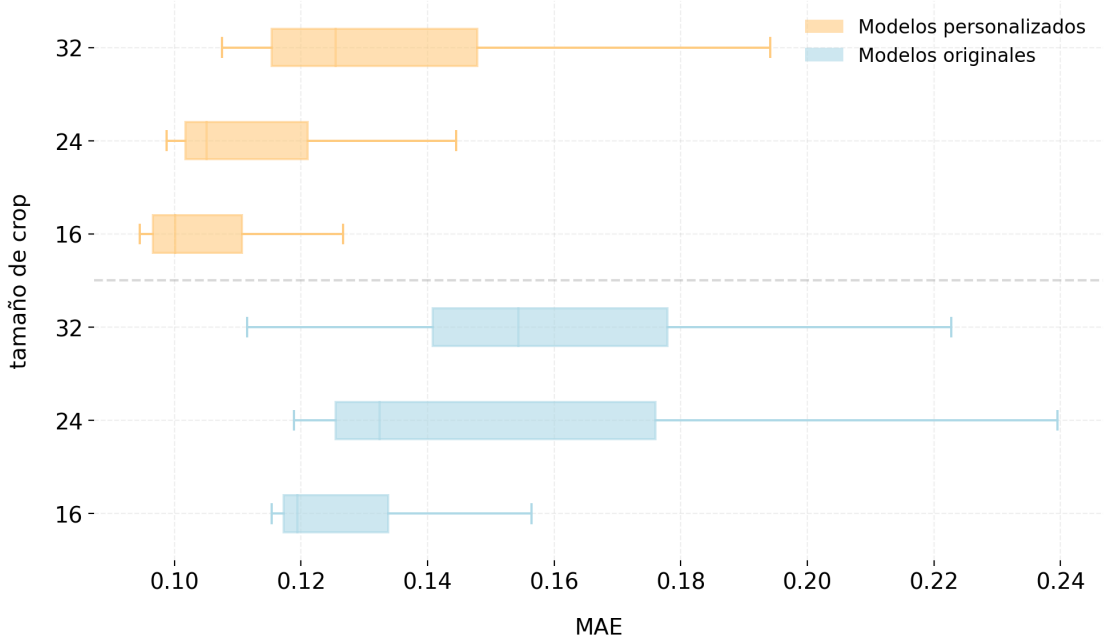


Figura 5.5: Boxplots de los resultados de las optimizaciones con SRGAN-B

Al igual que sucedió con SRGAN-A, en SRGAN-B los modelos personalizados mostraron una clara superioridad respecto a los modelos originales en la etapa de búsqueda de hiperparámetros. En el caso de SRGAN-B, en los modelos personalizados y en los modelos originales, los modelos entrenados con tamaños de crop más pequeños obtuvieron un mejor rendimiento en términos de MAE comparado con los modelos entrenados con tamaños de crop más grandes.

modelo	tamaño de crop	MAE	RMSE	max error	LE90	precision	recall
original	16	0.0996	**0.2881**	7.7282	0.1767	0.9943	0.9957
	24	0.1049	0.2942	**7.6624**	0.1851	0.9953	0.9948
	32	0.1182	0.3076	7.6974	0.2135	**0.9960**	0.9957
personalizado	16	**0.0832**	0.2943	8.3444	**0.1339**	0.9943	0.9985
	24	0.0886	0.2985	8.4486	0.1437	0.9942	**0.9986**
	32	0.0976	0.3121	8.4598	0.1591	0.9911	0.9985

Tabla 5.5: Mejores modelos de optimización de hiperparámetros con SRGAN-B.

Los resultados de la Tabla 5.5 confirmaron que los modelos entrenados con tamaños de crop menores obtuvieron un mejor rendimiento en las tres métricas principales en comparación con los modelos entrenados con tamaños de crop mayores. En particular, el modelo original entrenado con crops de 16×16 fue el que mejor rendimiento obtuvo dentro de los modelos originales. Lo mismo ocurrió con los modelos personalizados en donde el modelo entrenado con crops de 16×16 fue el que obtuvo mejor resultado en las tres métricas principales.

El mejor modelo de SRGAN-B personalizado en comparación con el mejor modelo original presentó una reducción del 16.1 % en MAE, del 26.7 % en LE90 y un aumento en RMSE del 3.1 %. En cuanto al max error y la precision, los modelos originales de SRGAN-B obtuvieron en promedio un mejor resultado comparado con los modelos personalizados de SRGAN-B. No obstante, en términos de recall los modelos personalizados de SRGAN-B superaron a los originales de SRGAN-B. Dados los resultados obtenidos en las tres métricas principales, se consideraron los modelos personalizados de SRGAN-B superiores a los modelos originales de SRGAN-B.

modelo	tamaño de crop	tamaño de batch	LR_D	LR_G	w_{ADV}	w_{CONT}	w_{MSE}	w_{MAE}
original	16	8	2.8×10^{-7}	6.2×10^{-7}	1.1×10^{-5}	8.9×10^{-3}	1.6×10^{-1}	-
	24	8	4.8×10^{-4}	4.0×10^{-5}	8.7×10^{-5}	1.4×10^{-3}	1.8×10^{-1}	-
	32	8	2.5×10^{-4}	4.6×10^{-7}	1.5×10^{-4}	7.9×10^{-4}	1.1×10^{-2}	-
personalizado	16	8	1.8×10^{-5}	4.9×10^{-6}	1.1×10^{-4}	8.1×10^{-4}	1.2×10^{-3}	3.0×10^{-1}
	24	8	8.5×10^{-8}	7.6×10^{-6}	6.4×10^{-5}	1.5×10^{-4}	9.4×10^{-5}	2.2×10^{-1}
	32	8	1.1×10^{-4}	2.6×10^{-5}	1.2×10^{-4}	5.8×10^{-5}	1.6×10^{-4}	2.5×10^{-1}

Tabla 5.6: Mejores hiperparámetros para SRGAN-B.

Los seis mejores modelos de SRGAN-B fueron entrenados con un tamaño de batch de 8, lo que reafirmó la superioridad del entrenamiento con tamaños de batch menores en SRGAN. En los modelos personalizados el w_{MAE} fue el peso que tuvo mayor magnitud, a diferencia de los modelos originales donde el w_{MSE} fue el peso con mayor magnitud.

La Tabla 5.7 muestra el crop seleccionado superresuelto por cada uno de los modelos SRGAN-B analizados.

Los crops generados con SRGAN-B presentaron menos artefactos que los generados con SRGAN-A. En términos de MAE, todos los crops generados con SRGAN-B obtuvieron mejores resultados que los correspondientes en SRGAN-A. En términos de RMSE la diferencia no es tan pronunciada aunque SRGAN-B sigue mostrando un rendimiento superior.

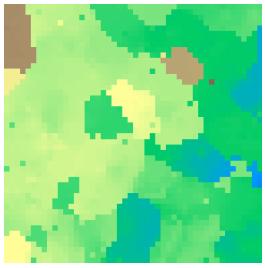
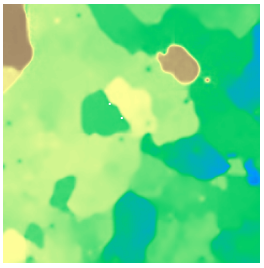
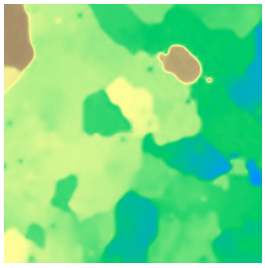
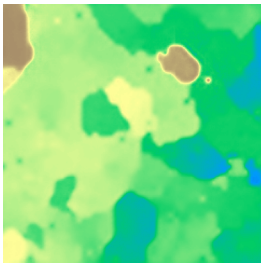
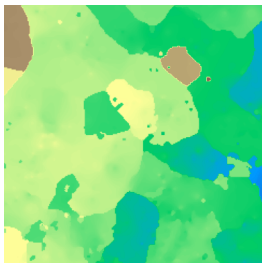
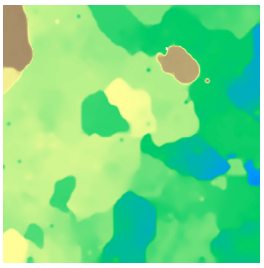
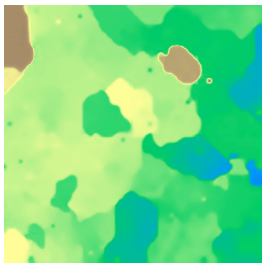
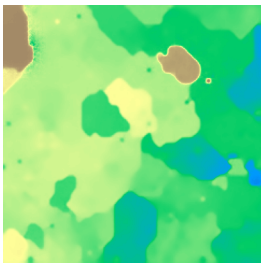
baja resolución	16×16 original	24×24 original	32×32 original
			
	MAE: 0.2568 RMSE: 0.7496	MAE: 0.2643 RMSE: 0.7525	MAE: 0.2902 RMSE: 0.7549
alta resolución	16×16 personalizado	24×24 personalizado	32×32 personalizado
			
	MAE: 0.1989 RMSE: 0.7734	MAE: 0.2168 RMSE: 0.7790	MAE: 0.2355 RMSE: 0.7903

Tabla 5.7: Comparación de un crop generado por los seis modelos reportados de SRGAN-B.

5.2.4. Resultados obtenidos con ESRGAN

En esta subsección se presentan los resultados obtenidos con ESRGAN. Se reportan los resultados correspondientes al entrenamiento A seguidos por los resultados obtenidos con el entrenamiento B.

Entrenamiento A

La Figura 5.6 presenta los boxplots correspondientes al MAE para los modelos ESRGAN-A. La Tabla 5.8 muestra los resultados de los mejores modelos de cada optimización con ESRGAN-A y la Tabla 5.9 detalla los hiperparámetros asociados. En ESRGAN, la única diferencia entre los modelos originales y los modelos personalizados radica en la función de pérdida empleada durante el preentrenamiento.

La Figura 5.6 y la Tabla 5.8 muestran que las diferencias entre los modelos originales y los personalizados en ESRGAN-A no fueron tan amplias como en SRGAN. A pesar de que los modelos originales obtuvieron los mejores resultados para cada métrica, las comparaciones uno a uno con los modelos personalizados muestran diferencias poco significativas. Por ejemplo, el modelo personalizado entrenado con crops de 16×16 superó al modelo original con el mismo tamaño de crop en todas las métricas principales, lo que evidenció que los modelos personalizados pueden ofrecer un rendimiento competitivo frente a los originales. Se consideró el modelo original entrenado con crops de 32×32 como el mejor modelo de ESRGAN-A, debido a que superó a todos los demás modelos en MAE y en LE90, dos de las tres métricas principales.

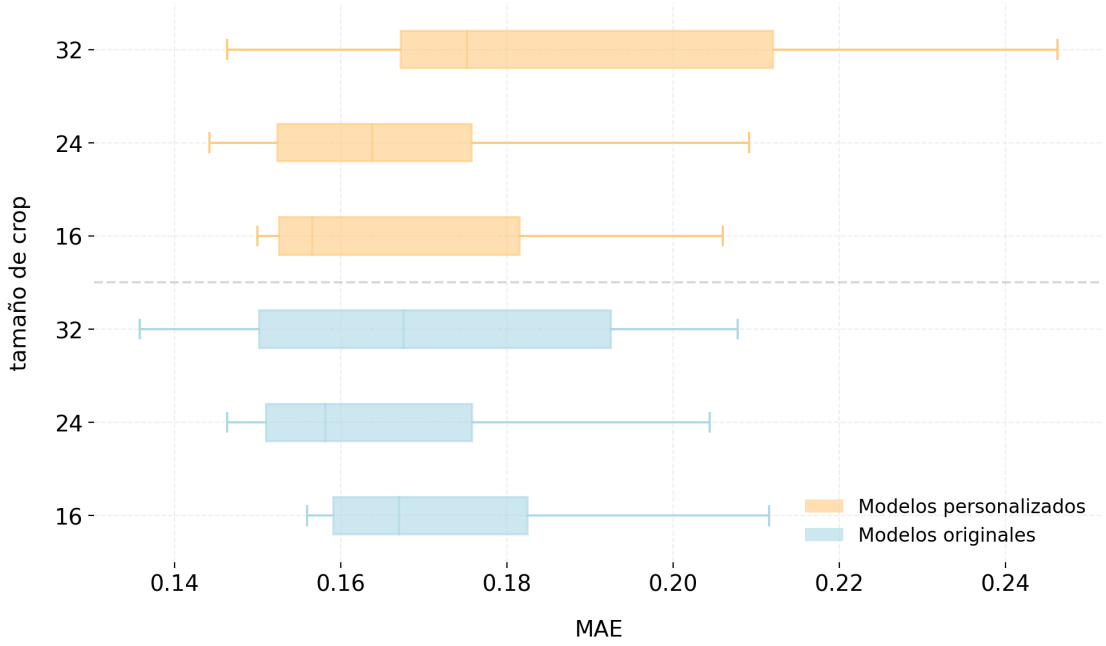


Figura 5.6: Boxplots de los resultados de las optimizaciones con ESRGAN-A

modelo	tamaño de crop	MAE	RMSE	max error	LE90	precision	recall
original	16	0.1346	0.3447	9.7169	0.2213	0.9948	**0.9991**
	24	0.1309	**0.3244**	**8.4561**	0.2180	0.9957	0.9990
	32	**0.1261**	0.3333	9.0126	**0.2017**	**0.9959**	0.9988
personalizado	16	0.1306	0.3344	8.5197	0.2138	0.9952	0.9989
	24	0.1298	0.3379	8.9253	0.2093	0.9957	0.9989
	32	0.1460	0.3598	11.7099	0.2347	0.9957	0.9988

Tabla 5.8: Mejores modelos de optimización de hiperparámetros con ESRGAN-A evaluados en el conjunto de validación

La Tabla 5.9 muestra cómo el peso del MAE fue predominante en los modelos originales. Sin embargo, en los modelos de ESRGAN-A personalizados este fenómeno no ocurrió. De hecho, el modelo ESRGAN-A personalizado entrenado con crops de 24×24 se posicionó como el segundo mejor modelo de ESRGAN-A, con un w_{MAE} significativamente más bajo.

La Tabla 5.10 muestra una comparación de los modelos ESRGAN-A con el mismo crop que en la Tabla 5.4 de SRGAN-A.

modelo	tamaño de crop	tamaño de batch	LR_D	LR_G	w_{ADV}	w_{CONT}	w_{MAE}
original	16	16	3.6×10^{-7}	1.7×10^{-6}	1.1×10^{-4}	1.9×10^{-3}	9.6×10^{-1}
	24	32	2.8×10^{-8}	1.8×10^{-6}	9.1×10^{-5}	1.5×10^{-5}	1.2×10^{-1}
	32	16	2.1×10^{-8}	5.8×10^{-6}	6.1×10^{-4}	2.7×10^{-4}	3.9×10^{-2}
personalizado	16	32	7.4×10^{-6}	3.1×10^{-7}	2.1×10^{-5}	1.7×10^{-5}	6.5×10^{-1}
	24	8	5.8×10^{-8}	8.2×10^{-8}	5.4×10^{-3}	1.3×10^{-3}	2.7×10^{-4}
	32	8	7.5×10^{-8}	4.0×10^{-7}	5.3×10^{-3}	1.6×10^{-3}	3.8×10^{-5}

Tabla 5.9: Mejores hiperparámetros para ESRGAN-A.

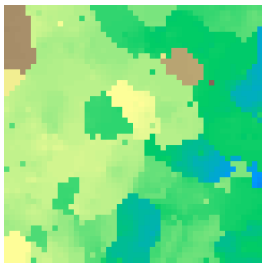
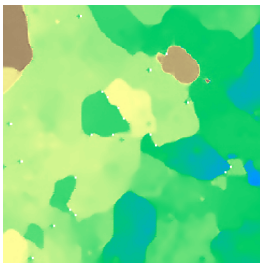
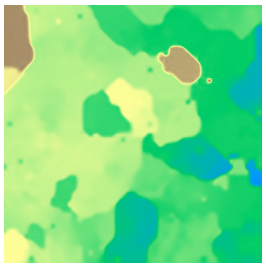
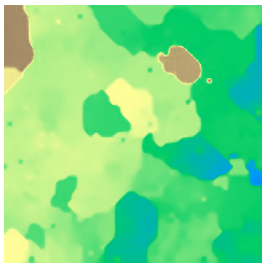
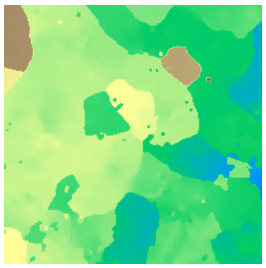
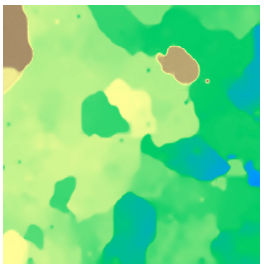
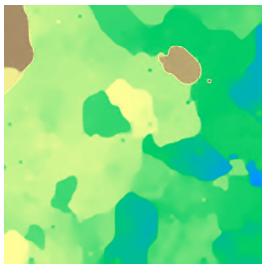
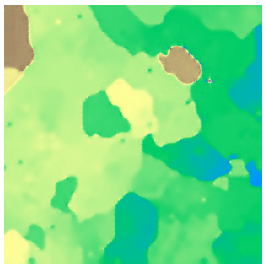
baja resolución	16×16 original	24×24 original	32×32 original
			
	MAE: 0.2510 RMSE: 0.8196	MAE: 0.2553 RMSE: 0.7561	MAE: 0.2551 RMSE: 0.7907
alta resolución	16×16 personalizado	24×24 personalizado	32×32 personalizado
			
	MAE: 0.2332 RMSE: 0.7706	MAE: 0.2375 RMSE: 0.7935	MAE: 0.3111 RMSE: 0.8479

Tabla 5.10: Comparación de un crop superresuelto por los seis modelos reportados de ESRGAN-A.

La diferencia visual entre los modelos personalizados y los originales no es notoria. El único artefacto visible se encontró en el crop generado por el modelo 32×32 personalizado, ubicado alrededor del área marrón cercana a la esquina superior derecha. Al igual que con SRGAN-A, surgió el problema de aparición de celdas clasificadas incorrectamente como vacías en el crop generado por el modelo 16×16 original. Sin embargo, este problema no se presentó en el crop generado por el modelo personalizado de 16×16. En relación con las métricas presentadas en la Tabla 5.10, no se hallaron diferencias significativas entre los modelos personalizados y los originales, ni una tendencia clara en los modelos

entrenados con distintos tamaños de crop. Estos hallazgos fueron consistentes con los resultados mostrados en la Tabla 5.8.

Entrenamiento B

La Figura 5.7 presenta los boxplots correspondientes al MAE para los modelos ESRGAN-B. La Tabla 5.11 muestra los resultados de los mejores modelos de cada optimización con ESRGAN-B y la Tabla 5.12 detalla los hiperparámetros asociados.

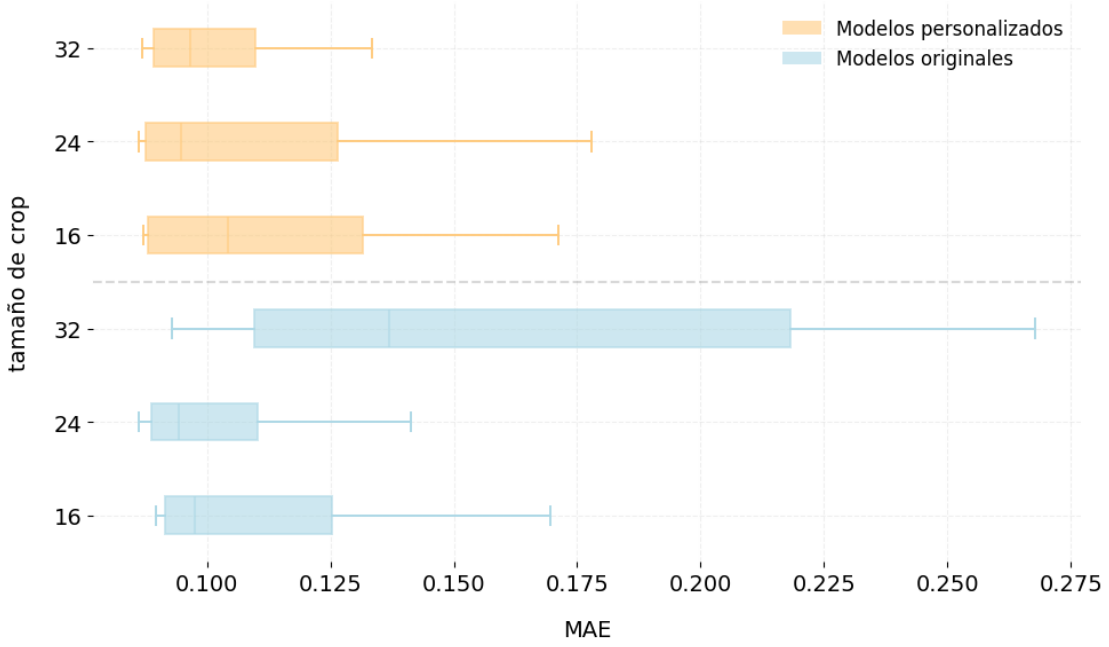


Figura 5.7: Boxplots de los resultados de las optimizaciones con ESRGAN-B

Al igual que en ESRGAN-A, los resultados de las optimizaciones de hiperparámetros fueron similares entre los modelos originales y personalizados. En particular, la optimización con crops de 32×32 del modelo original presentó un rango intercuartílico más amplio en comparación con sus pares en ESRGAN-B. Esto indica una mayor dispersión en los resultados para la métrica MAE en el modelo ESRGAN-B original entrenado con crops de 32×32.

Los modelos originales y personalizados de ESRGAN-B mostraron resultados muy similares en las tres métricas principales. El único modelo con un rendimiento significativamente inferior al del resto en las tres métricas principales fue el modelo original de ESRGAN-B entrenado con crops de 32×32, lo que concuerda con los boxplots de la Figura 5.7.

El mejor modelo de ESRGAN-B personalizado en comparación con el mejor modelo de ESRGAN-B original presentó una reducción de 2.5 % en MAE y de 2.0 % en LE90. No obstante, el RMSE del modelo personalizado fue 1.6 % mayor que el del modelo original. Dados los resultados obtenidos se concluyó que la diferencia entre los modelos originales y personalizados en ESRGAN-B no fue significativa.

En cuanto a los hiperparámetros presentados en la Tabla 5.12, se mantuvo la tendencia observada en SRGAN-B, pues los modelos presentaron un w_{MAE} de mayor magnitud

modelo	tamaño de crop	MAE	RMSE	max error	LE90	precision	recall
original	16	0.0788	**0.2846**	**8.2606**	0.1269	**0.9967**	0.9851
	24	0.0778	0.2870	8.4945	0.1236	**0.9967**	0.9669
	32	0.0856	0.3089	8.9489	0.1303	0.9965	**0.9959**
personalizado	16	**0.0768**	0.2893	8.4573	**0.1211**	0.9966	0.9906
	24	0.0780	0.2967	8.5617	0.1222	0.8882	0.0249
	32	0.0805	0.2936	8.3237	0.1266	**0.9967**	0.9895

Tabla 5.11: Mejores modelos de optimización de hiperparámetros con ESRGAN-B evaluados en el conjunto de validación

modelo	tamaño de crop	tamaño de batch	LR_D	LR_G	w_{ADV}	w_{CONT}	w_{MAE}
original	16	16	2.7×10^{-8}	9.4×10^{-6}	8.5×10^{-5}	2.0×10^{-5}	9.0×10^{-2}
	24	8	8.5×10^{-8}	1.7×10^{-5}	2.8×10^{-5}	5.7×10^{-5}	6.4×10^{-2}
	32	8	2.4×10^{-7}	2.9×10^{-4}	6.1×10^{-3}	2.5×10^{-4}	3.1×10^{-2}
personalizado	16	8	3.6×10^{-4}	8.1×10^{-7}	4.5×10^{-5}	6.3×10^{-4}	9.1×10^{-1}
	24	8	2.7×10^{-7}	6.1×10^{-6}	4.9×10^{-5}	3.6×10^{-4}	1.4×10^{-1}
	32	8	3.2×10^{-8}	9.4×10^{-6}	2.6×10^{-3}	1.1×10^{-5}	2.9×10^{-1}

Tabla 5.12: Mejores hiperparámetros para ESRGAN-B

respecto al resto de pesos. En particular, el modelo personalizado de 16×16 tuvo el mayor peso en w_{MAE} con un valor de 0,91, lo que explica su superioridad en esta métrica.

La Tabla 5.13 muestra el crop seleccionado generado por cada uno de los modelos de ESRGAN-B analizados.

En cuanto a los crops generados, los modelos ESRGAN-B superaron a los modelos ESRGAN-A cuantitativa y cualitativamente. Además, los crops generados no presentaron artefactos, patrones evidentes ni celdas erróneamente clasificadas como sin datos.

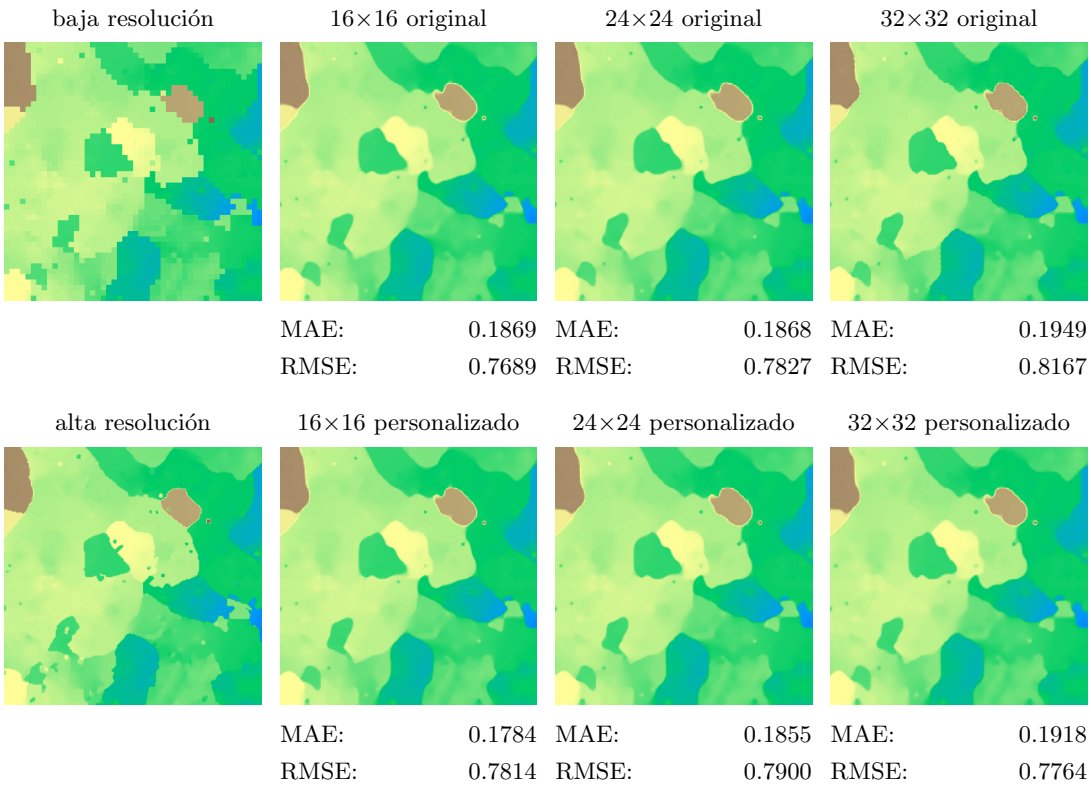


Tabla 5.13: Comparación de un crop superresuelto por los seis modelos reportados de ESRGAN-B.

5.2.5. Resultados obtenidos con DSRGAN

En esta subsección se presentan los resultados obtenidos con DSRGAN. Se reportan los resultados correspondientes al entrenamiento A seguidos por los resultados obtenidos con el entrenamiento B.

Entrenamiento A

La Figura 5.8 presenta los boxplots correspondientes al MAE para los modelos DSRGAN-A.

La Tabla 5.14 muestra los resultados de los mejores modelos de cada optimización con DSRGAN-A y la Tabla 5.15 detalla los hiperparámetros asociados. Al igual que sucedió con SRGAN, DSRGAN no incluye la métrica MAE en la función de pérdida, por lo que esta métrica fue incluida como un hiperparámetro únicamente en los modelos personalizados. A diferencia de las arquitecturas SRGAN y ESRGAN, en DSRGAN la content loss y el MSE son equivalentes dado que esta arquitectura no utiliza VGG para la pérdida de contenido, sino que utiliza el MSE.

Los modelos personalizados de DSRGAN-A mostraron un rendimiento significativamente superior al de los modelos originales de la misma arquitectura en las métricas MAE, RMSE y LE90. La Figura 5.8 muestra que los peores modelos personalizados tuvieron un mejor rendimiento en la métrica MAE que la mediana de los mejores modelos originales entrenados con el mismo tamaño de crop.

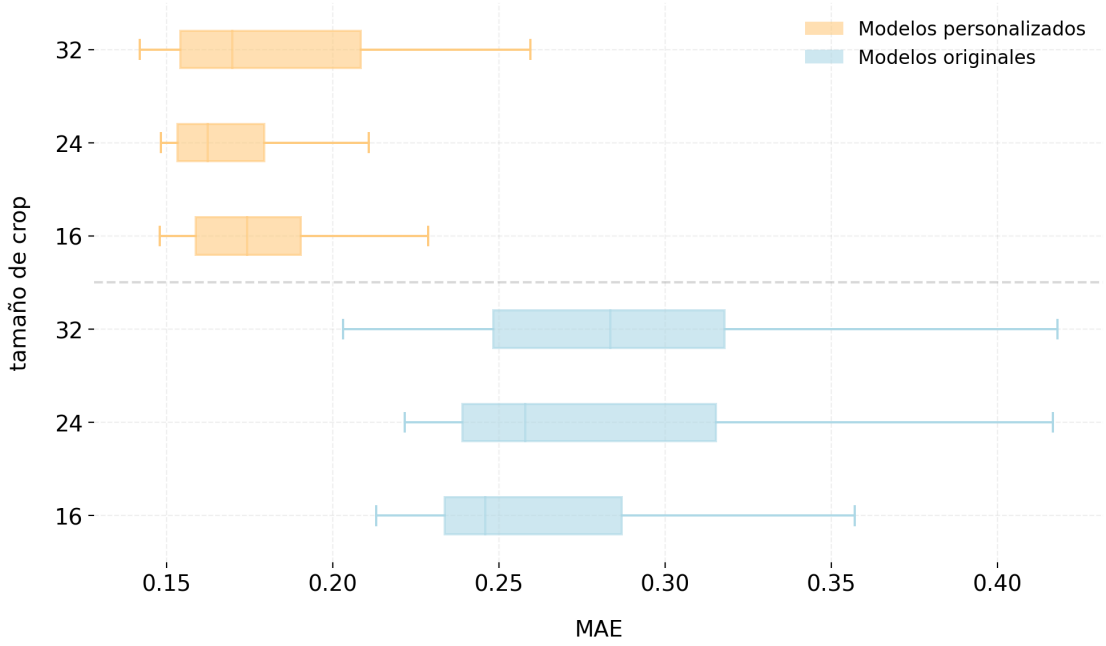


Figura 5.8: Boxplots de los resultados de las optimizaciones con DSRGAN-A

modelo	tamaño de crop	MAE	RMSE	max error	LE90	precision	recall
original	16	0.1852	0.3670	**8.2923**	0.3425	0.9945	**0.9993**
	24	0.1926	0.3914	9.5469	0.3467	0.9929	0.9993
	32	0.1823	0.3835	9.7295	0.3243	0.9934	0.9992
personalizado	16	**0.1268**	0.3347	8.9733	**0.2089**	0.9941	0.9991
	24	0.1308	0.3301	8.4938	0.2161	**0.9953**	0.9991
	32	0.1284	**0.3284**	8.7166	0.2096	0.9951	0.9991

Tabla 5.14: Mejores modelos de optimización de hiperparámetros con DSRGAN-A.

modelo	tamaño de crop	tamaño de batch	LR_D	LR_G	w_{ADV}	w_{CONT}	w_{MAE}
original	16	32	2.0×10^{-6}	1.4×10^{-8}	4.8×10^{-4}	6.3×10^{-3}	-
	24	32	6.3×10^{-8}	7.3×10^{-7}	1.6×10^{-4}	1.7×10^{-3}	-
	32	16	6.1×10^{-8}	6.6×10^{-7}	2.6×10^{-4}	8.1×10^{-3}	-
personalizado	16	8	2.5×10^{-8}	7.2×10^{-6}	7.4×10^{-3}	6.3×10^{-5}	1.0×10^{-1}
	24	16	1.5×10^{-8}	4.0×10^{-6}	2.4×10^{-4}	6.5×10^{-5}	2.6×10^{-1}
	32	16	3.2×10^{-8}	4.6×10^{-8}	1.8×10^{-4}	8.8×10^{-3}	1.6×10^{-1}

Tabla 5.15: Mejores hiperparámetros para DSRGAN-A.

El modelo personalizado con mejor desempeño fue el entrenado con crops de 16×16, que alcanzó los valores más bajos en MAE y en LE90 entre todos los modelos de DSRGAN-A. El modelo con mejor desempeño dentro de los modelos originales también fue el entrenado con crops de 16×16. En comparación con el modelo original de 16×16, el modelo personalizado de 16×16 presentó una mejora del 32.2 % en MAE, del 7.9 % en RMSE y del 40.4 % en LE90. Dados los resultados obtenidos en las tres métricas principales, se consideraron los modelos personalizados de DSRGAN-A superiores a los modelos originales de DSRGAN-A.

Respecto a los hiperparámetros de los modelos DSRGAN-A, la Tabla 5.15 muestra que los mejores modelos personalizados utilizaron un tamaño de batch menor en comparación con los mejores modelos originales. Sin embargo, en el resto de los hiperparámetros no se identificaron relaciones claras entre los diferentes tamaños de crops ni entre las arquitecturas personalizadas y originales.

La Tabla 5.16 muestra una comparación de los modelos DSRGAN-A con el mismo crop que se utilizó en SRGAN y en ESRGAN.

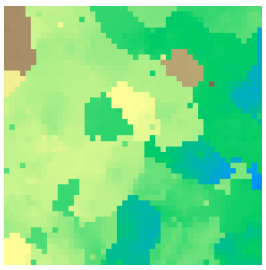
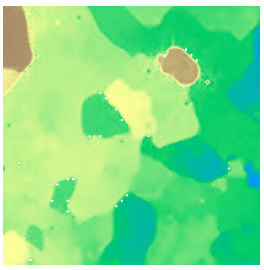
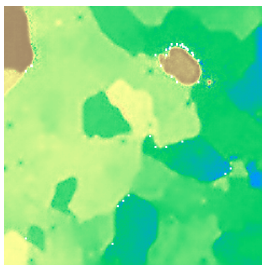
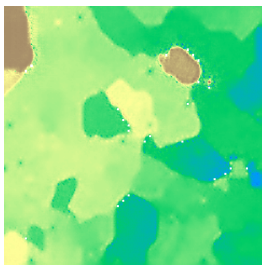
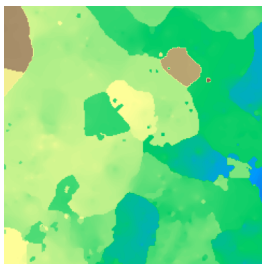
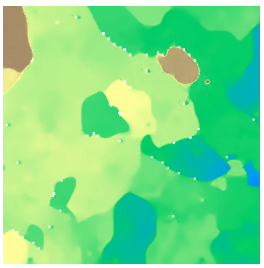
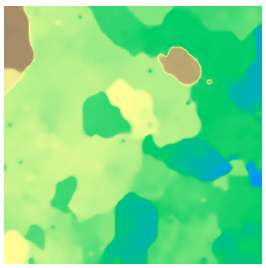
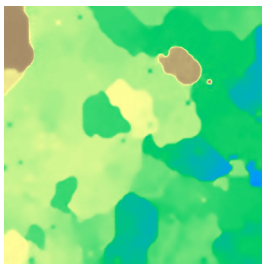
baja resolución	16×16 original	24×24 original	32×32 original
			
	MAE: 0.3902 RMSE: 0.8163	MAE: 0.3788 RMSE: 0.8371	MAE: 0.3649 RMSE: 0.8343
alta resolución	**16×16 personalizado**	**24×24 personalizado**	**32×32 personalizado**
			
	MAE: 0.2569 RMSE: 0.8020	MAE: 0.2482 RMSE: 0.7742	MAE: 0.2540 RMSE: 0.7737

Tabla 5.16: Comparación de un crop superresuelto por los seis modelos reportados de DSRGAN-A.

Al igual que en SRGAN, la Tabla 5.16 muestra una superioridad visual de los modelos personalizados respecto a los modelos originales, lo que es coherente con las métricas reportadas en la Tabla 5.14. Se observan artefactos en los tres crops generados por los modelos originales. En particular, en la esquina superior izquierda de los crops generados por los modelos originales se observa un patrón que no se encuentra en el crop en baja resolución ni tampoco en los crops generados por los modelos personalizados. Además, en los crops generados por los tres modelos originales se observan celdas erróneamente

detectadas como vacías. En los crops generados por los modelos personalizados solo se observan celdas erróneamente detectadas como vacías en el modelo entrenado con crops de tamaño 16×16.

Entrenamiento B

La Figura 5.9 presenta los boxplots correspondientes al MAE para los modelos DSRGAN-B. La Tabla 5.17 muestra los resultados de los mejores modelos de cada optimización con DSRGAN-B y la Tabla 5.18 detalla los hiperparámetros asociados.

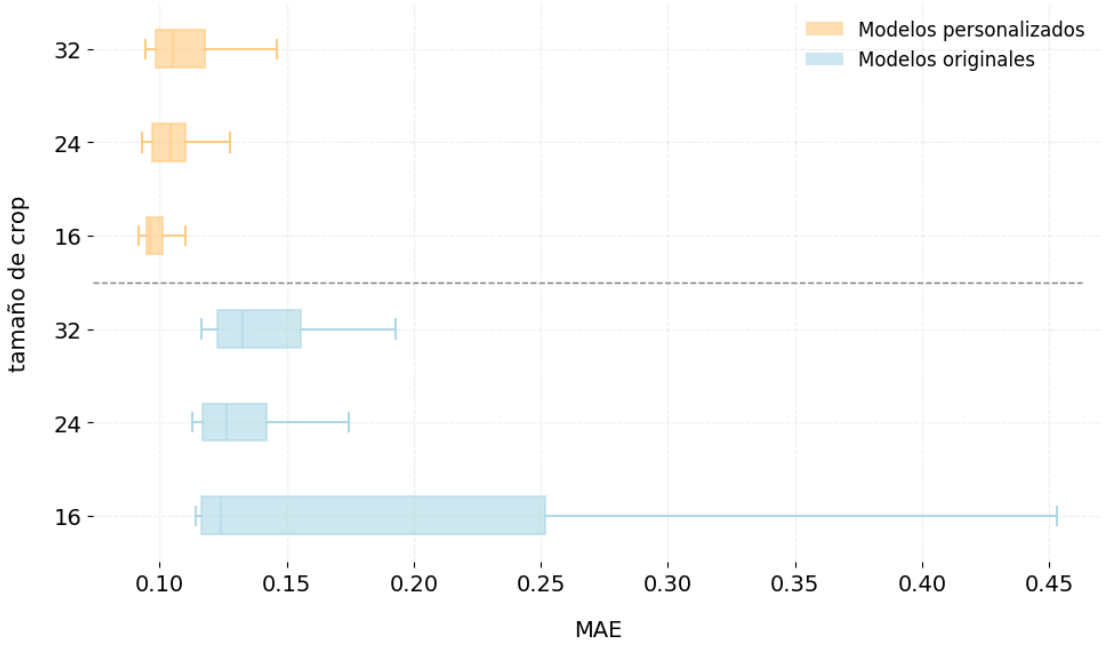


Figura 5.9: Boxplots de los resultados de las optimizaciones con DSRGAN-B

modelo	tamaño de crop	MAE	RMSE	max error	LE90	precision	recall
original	16	0.0989	0.2870	7.8697	0.1778	0.9900	0.9958
	24	0.0994	**0.2867**	7.7746	0.1765	0.9951	0.9962
	32	0.1055	0.2938	**7.7385**	0.1856	**0.9959**	0.9883
personalizado	16	**0.0810**	0.2945	8.3531	**0.1286**	0.9952	0.9977
	24	0.0836	0.2983	8.4794	0.1335	0.9946	**0.9986**
	32	0.0867	0.2995	8.5488	0.1386	0.9946	**0.9986**

Tabla 5.17: Mejores modelos de optimización de hiperparámetros con DSRGAN-B.

Al igual que en DSRGAN-A, los modelos personalizados mostraron un rendimiento superior respecto a los originales en la búsqueda de hiperparámetros en términos de MAE. En términos de RMSE, los modelos originales fueron superiores a los modelos personalizados y en LE90 los modelos personalizados fueron superiores.

Dentro de los modelos personalizados de DSRGAN-B, el mejor modelo fue el entrenado con crops de 16×16 dado que obtuvo el mejor resultado en las tres métricas principales. De manera similar, entre los modelos originales de DSRGAN-B el mejor fue el modelo entrenado con crops de 16×16, dado que fue superior en las métricas MAE y RMSE. El mejor modelo personalizado de DSRGAN-B presentó una reducción de 18.1 % en MAE, un aumento de 3.7 % en RMSE y una reducción de 30.9 % en LE90 en comparación con el mejor modelo original de DSRGAN-B. Dados los resultados obtenidos en dos de las tres métricas principales, se consideraron los modelos personalizados de DSRGAN-B superiores a los modelos originales de DSRGAN-B.

modelo	tamaño de crop	tamaño de batch	LR_D	LR_G	w_{ADV}	w_{CONT}	w_{MAE}
original	16	8	1.1×10^{-8}	5.1×10^{-7}	1.1×10^{-4}	8.4×10^{-3}	-
	24	8	2.7×10^{-8}	8.2×10^{-7}	1.5×10^{-4}	1.4×10^{-3}	-
	32	8	1.3×10^{-8}	6.4×10^{-7}	3.9×10^{-4}	1.5×10^{-3}	-
personalizado	16	8	5.0×10^{-8}	1.3×10^{-5}	1.2×10^{-4}	9.7×10^{-5}	1.1×10^{-1}
	24	8	2.1×10^{-6}	4.3×10^{-6}	2.6×10^{-3}	2.5×10^{-5}	9.1×10^{-2}
	32	8	1.7×10^{-8}	8.6×10^{-6}	4.9×10^{-3}	1.0×10^{-4}	2.8×10^{-1}

Tabla 5.18: Mejores hiperparámetros para DSRGAN-B.

Se destaca que para los modelos personalizados y originales el tamaño de batch óptimo encontrado es 8. En los modelos originales, el peso de mayor magnitud es el w_{CONT}, mientras que en los modelos personalizados el peso de mayor magnitud fue w_{MAE}. Además, en todos los modelos el learning rate del generador fue mayor que el del discriminador.

La Figura 5.19 muestra la superioridad cualitativa y cuantitativa de los modelos DSRGAN-B personalizados en comparación con los modelos DSRGAN-B originales. Los crops generados por DSRGAN-B superaron ampliamente a los generados con DSRGAN-A en términos de MAE, RMSE y calidad visual. DSRGAN-A presentó una mayor cantidad de celdas erróneamente etiquetadas como sin datos en comparación con las arquitecturas SRGAN y ESRGAN. Esta clasificación errónea de celdas se redujo con DSRGAN-B, dado que únicamente en el modelo original entrenado con crops de 16×16 se distinguieron celdas clasificadas erróneamente como sin datos.

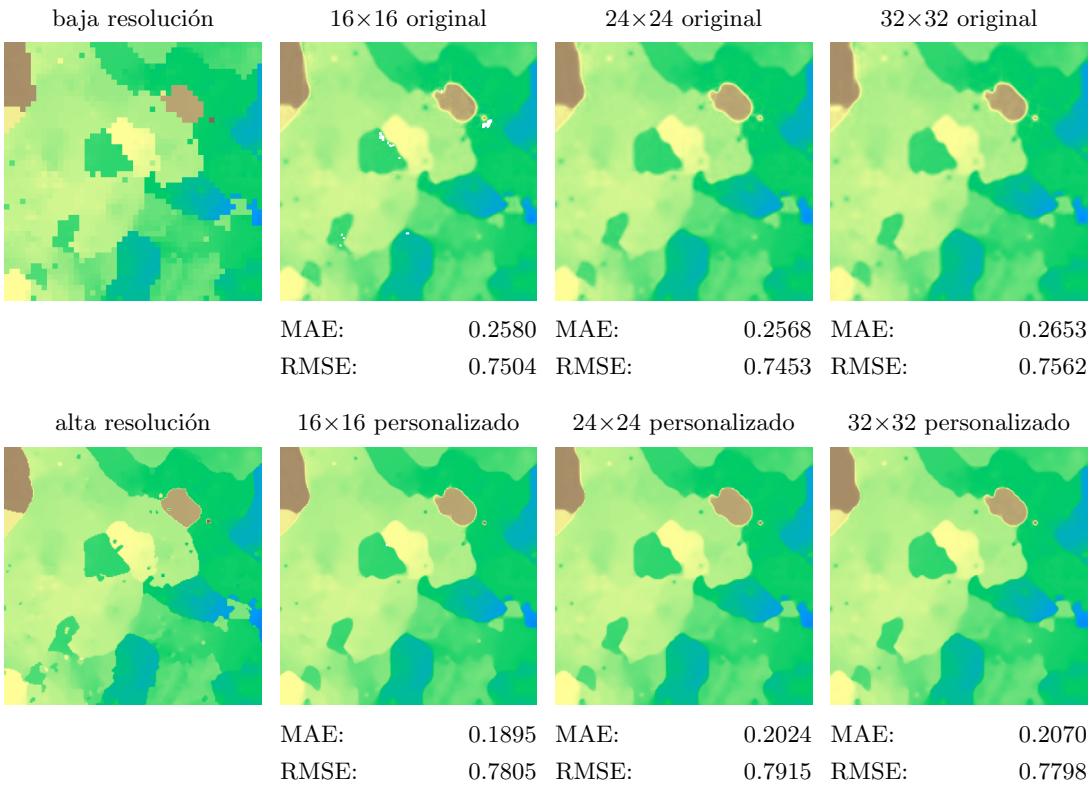


Tabla 5.19: Comparación de un crop superresuelto por los seis modelos reportados de DSRGAN-B.

5.2.6. Resultados obtenidos con SRGAN$_Z$

Por último, se realizó una búsqueda de los mejores hiperparámetros al modelo SRGAN$_Z$. Dado el análisis de tiempos de la Sección 5.1, se decidió realizar menos intentos y menos épocas en cada intento, concretamente 30 intentos de 15 épocas cada uno. No se modificó la arquitectura como en SRGAN, ESRGAN y DSRGAN, por lo que en los análisis no se diferencia entre modelos originales y personalizados.

Entrenamiento A

En la Figura 5.10 se muestran los boxplots correspondientes a la distribución de la métrica MAE evaluada en las diferentes configuraciones evaluadas para SRGAN$_Z$-A. En la Tabla 5.20 se muestran los resultados de los mejores modelos de cada optimización con SRGAN$_Z$-A y la Tabla 5.21 detalla los hiperparámetros asociados.

El modelo entrenado con crops de tamaño 32×32 fue el que obtuvo el mejor desempeño en la métrica MAE y también en LE90, dos de las tres métricas consideradas principales. Por lo tanto se consideró que el modelo entrenado con crops de tamaño 32×32 fue el de mejores resultados para SRGAN$_Z$-A.

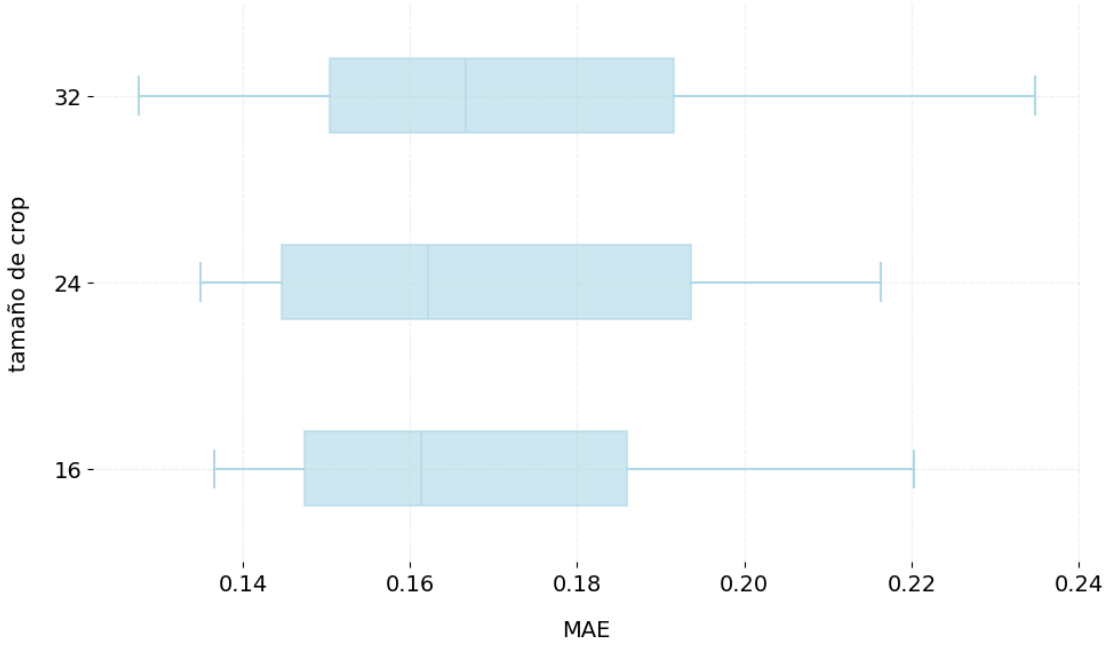


Figura 5.10: Boxplots de los resultados de las optimizaciones con SRGAN$_Z$-A.

tamaño de crop	MAE	RMSE	max error	LE90	precision	recall
16	0.1220	0.3292	**8.4844**	0.2003	0.9953	**0.9989**
24	0.1237	**0.3265**	8.6073	0.2044	0.9958	0.9988
32	**0.1172**	0.3303	9.0545	**0.1869**	**0.9960**	0.9987

Tabla 5.20: Mejores modelos de optimización de hiperparámetros con SRGAN$_Z$-A.

tamaño de crop	tamaño de batch	LR_D	LR_G	w_{ADV}	w_{CONT}	w_{MSE}	w_{MAE}
16	8	2.0×10^{-7}	5.1×10^{-4}	1.5×10^{-5}	9.9×10^{-4}	5.8×10^{-3}	8.4×10^{-2}
24	16	3.7×10^{-7}	2.0×10^{-4}	3.1×10^{-5}	4.4×10^{-5}	1.1×10^{-5}	4.3×10^{-4}
32	8	7.1×10^{-8}	1.2×10^{-4}	1.7×10^{-4}	2.2×10^{-3}	1.5×10^{-5}	1.3×10^{-2}

Tabla 5.21: Mejores hiperparámetros para SRGAN$_Z$-A.

Entrenamiento B

En la Figura 5.11 se muestran los boxplots correspondientes a la distribución de la métrica MAE evaluada en las diferentes configuraciones evaluadas para SRGAN$_Z$-B. En la Tabla 5.22 se muestran los resultados de los mejores modelos de cada optimización con SRGAN$_Z$-B y la Tabla 5.23 detalla los hiperparámetros asociados.

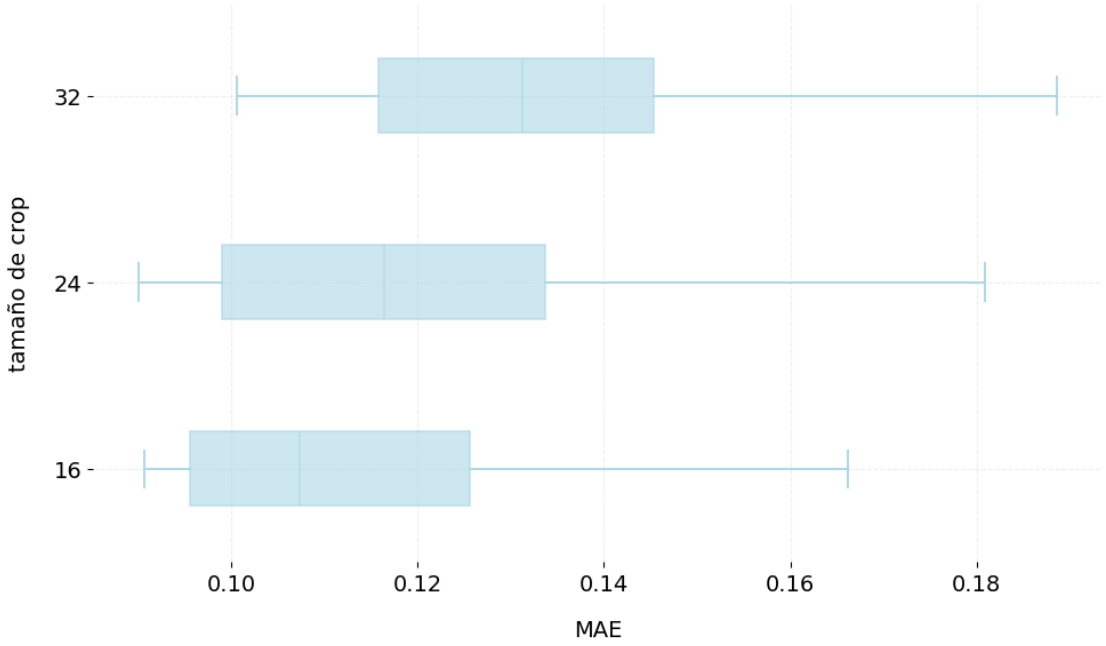


Figura 5.11: Boxplots de los resultados de las optimizaciones con SRGAN$_Z$-B

tamaño de crop	MAE	RMSE	max error	LE90	precision	recall
16	**0.0794**	0.3066	8.9950	**0.1256**	**0.9966**	0.9894
24	0.0811	**0.3032**	8.6374	0.1285	0.9965	**0.9951**
32	0.0956	0.3048	**8.4068**	0.1534	0.9965	0.9947

Tabla 5.22: Mejores modelos de optimización de hiperparámetros con SRGAN$_Z$-B.

tamaño de crop	tamaño de batch	LR_D	LR_G	w_{ADV}	w_{CONT}	w_{MAE}	w_{MSE}
16	16	2.1×10^{-6}	8.3×10^{-5}	2.0×10^{-4}	1.7×10^{-3}	9.5×10^{-2}	9.0×10^{-5}
24	32	6.7×10^{-8}	9.1×10^{-4}	1.1×10^{-4}	1.0×10^{-3}	4.7×10^{-1}	4.2×10^{-3}
32	16	3.2×10^{-4}	3.4×10^{-5}	2.0×10^{-5}	1.2×10^{-4}	4.2×10^{-1}	1.8×10^{-3}

Tabla 5.23: Mejores hiperparámetros para SRGAN$_Z$-B.

En los boxplots de SRGAN$_Z$-B, las medianas y los cuartiles se ubicaron en valores de MAE más bajos que los reportados para el entrenamiento A en la Figura 5.10.

Al comparar los resultados de la Tabla 5.22 con los resultados de la Tabla 5.20 se concluyó que el modelo SRGAN$_Z$-B entrenado con crops de 16×16 fue el mejor para el entrenamiento B y el modelo SRGAN$_Z$-A entrenado con crops de 32×32 fue el mejor para el entrenamiento A. El modelo SRGAN$_Z$-B entrenado con crops de 16×16 fue 32 % superior en MAE, 7 % en RMSE y 33 % en LE90 respecto a SRGAN$_Z$-A entrenado con crops de 32×32.

En la Tabla 5.22 se muestra que el modelo SRGAN$_Z$-B entrenado con crops de 16×16 fue el modelo que obtuvo el menor MAE y LE90, dos de las tres métricas principales. Por lo tanto, se consideró el modelo SRGAN$_Z$-B entrenado con crops de 16×16 como el mejor para la arquitectura SRGAN$_Z$.

5.3. Entrenamiento en profundidad

En esta sección se presentan los resultados de los entrenamientos en profundidad de los mejores modelos de cada arquitectura y tipo de entrenamiento. Los entrenamientos siguieron la metodología detallada en la Sección 4.5.3. Para cada modelo se realizaron cinco entrenamientos de 100 épocas cada uno, tanto para el tipo de entrenamiento A como para el entrenamiento B. Se reportó la evolución del MAE en los conjuntos de validación y entrenamiento así como la evolución de la función de pérdida del generador en cada uno de los cinco entrenamientos. Además de los valores reportados para cada entrenamiento, se incluyó el promedio como referencia. Los resultados correspondientes se organizaron en subsecciones específicas para cada arquitectura.

5.3.1. Entrenamiento de SRGAN

La Figura 5.12 y la Figura 5.13 muestran la evolución de la métrica MAE en el conjunto de validación y en el conjunto de entrenamiento respectivamente durante los cinco entrenamientos de SRGAN-A. Además, la Figura 5.14 presenta el resultado de la función de pérdida del generador en cada época.

Durante los cinco entrenamientos, la evolución de la métrica MAE mostró una tendencia similar, por este motivo se utilizó el promedio de los valores obtenidos como referencia para los análisis. En el conjunto de validación, la métrica MAE mejoró progresivamente hasta la época 41, punto en el que se alcanzó el mejor rendimiento promedio. Luego de la época 41, el valor de la métrica MAE aumentó de forma gradual hasta estabilizarse en una meseta alrededor de la época 90. Este aumento de la métrica MAE en el conjunto de validación no se debió a un sobreajuste a los datos de entrenamiento, pues la métrica MAE en el conjunto de entrenamiento siguió una tendencia similar a la tendencia en validación.

En la Figura 5.14 se muestra cómo la pérdida del generador mejoró a lo largo de todo el entrenamiento. Sin embargo, la métrica MAE en el conjunto de validación aumentó durante la mayor parte del entrenamiento. Esta contradicción se explica porque las métricas MAE y MSE en la función de pérdida del generador se calcularon utilizando los crops normalizados mientras que las métricas MAE y MSE sobre el conjunto de validación se calcularon con crops desnormalizados. Aunque el objetivo de la optimización fue similar, realizar la optimización en base a los crops normalizados no fue equivalente a realizar la optimización en base a los crops desnormalizados como se explicó en la Sección 4.5.1.

Como el objetivo de este proyecto de grado fue minimizar el error en los datos en su forma original (DEMs desnormalizados), se decidió incorporar el entrenamiento B a las evaluaciones realizadas.

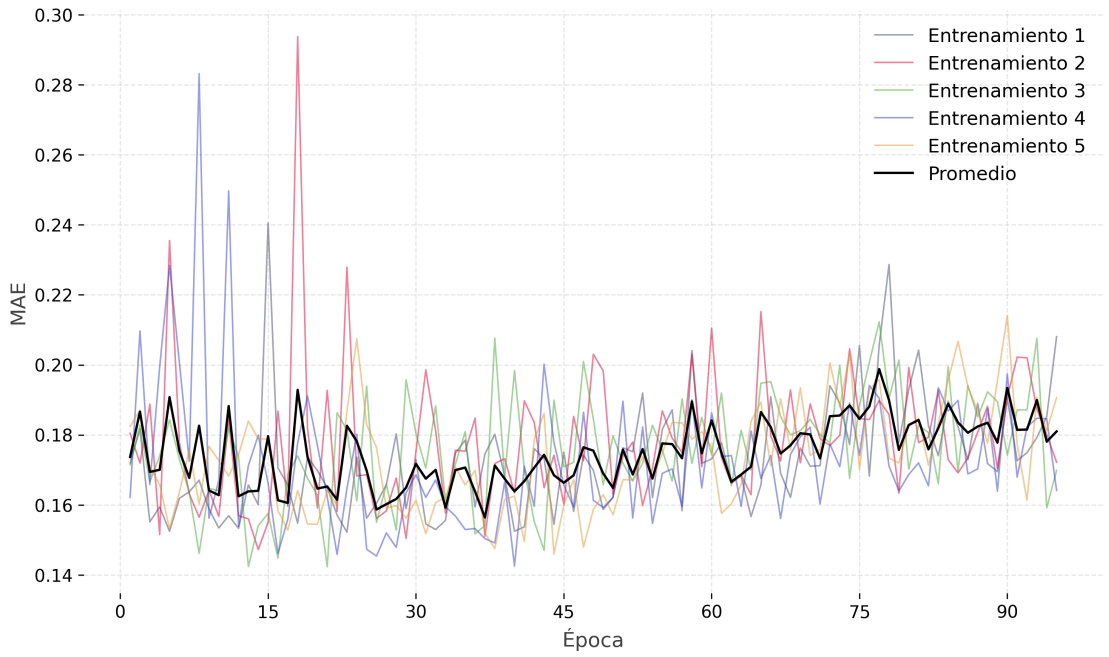


Figura 5.12: Evolución de la métrica MAE en el conjunto de validación para SRGAN-A.

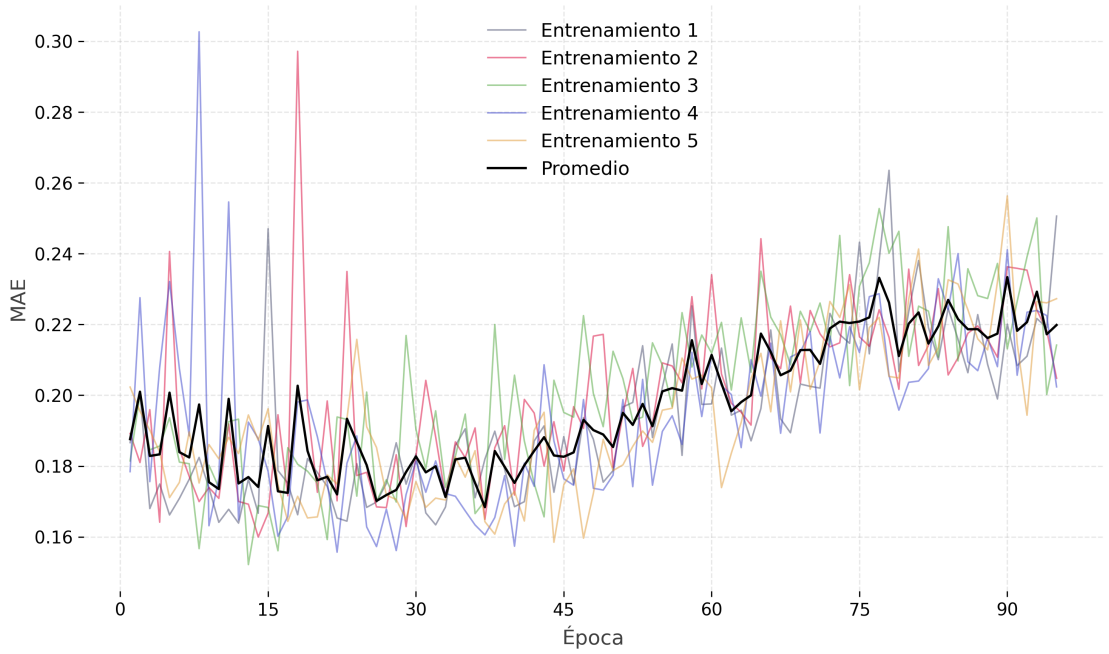


Figura 5.13: Evolución de la métrica MAE en el conjunto de entrenamiento para SRGAN-A.

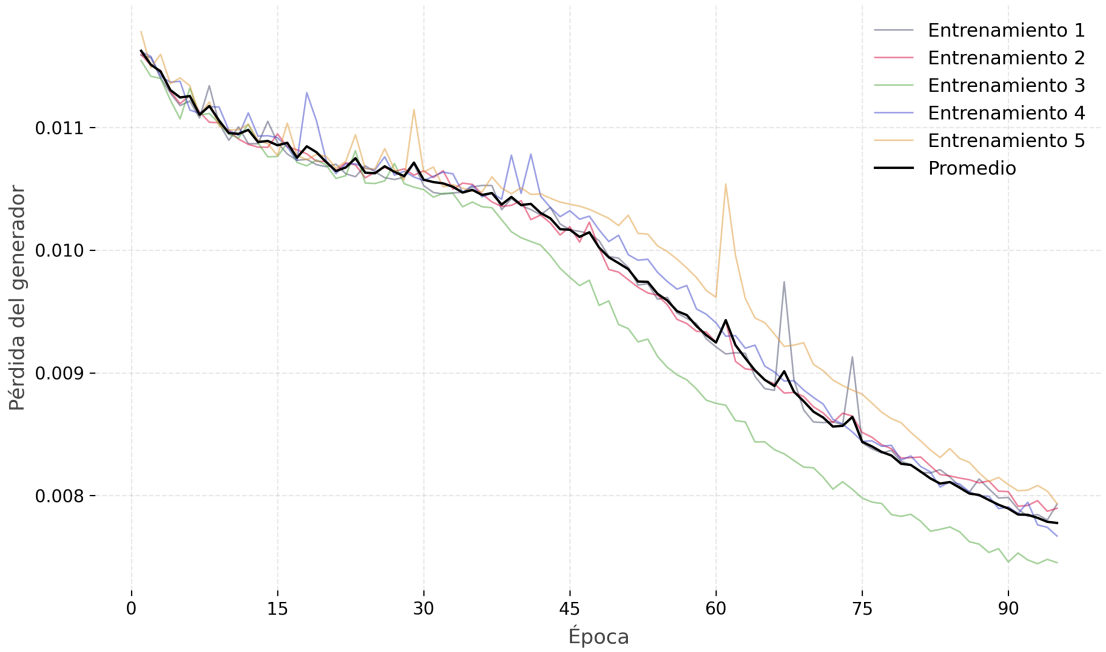


Figura 5.14: Evolución de la pérdida del generador en el conjunto de entrenamiento para SRGAN-A.

Las Figuras 5.15 y 5.16 muestran la evolución de la métrica MAE en los conjuntos de validación y entrenamiento respectivamente para SRGAN-B. La Figura 5.17 presenta el promedio de la pérdida del generador calculado en cada época para SRGAN-B.

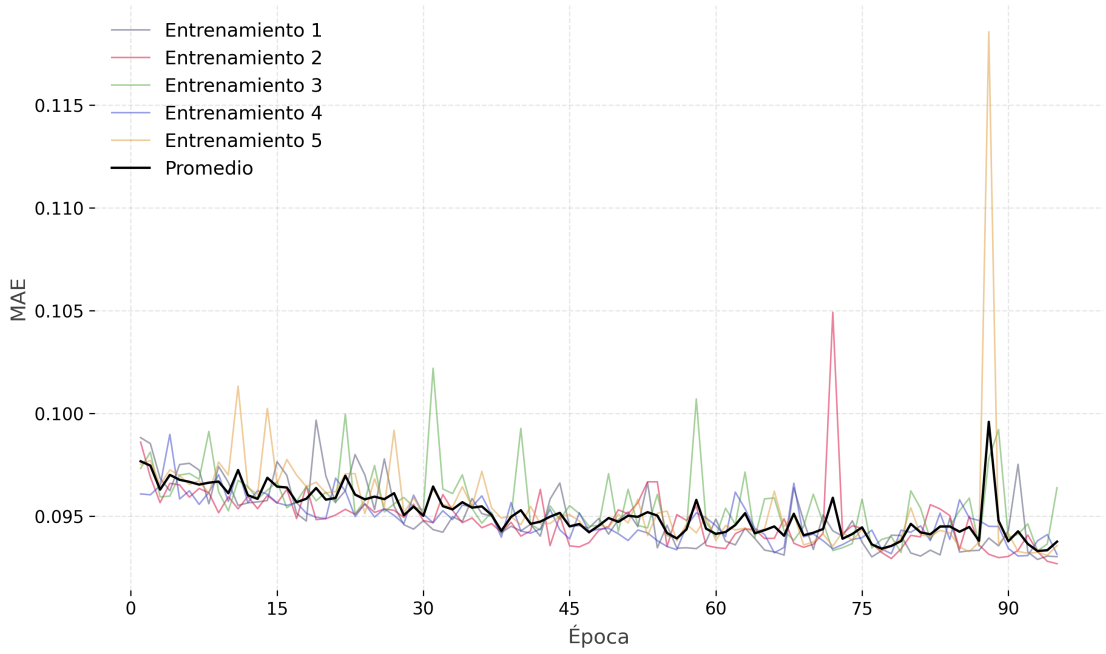


Figura 5.15: Evolución de la métrica MAE en el conjunto de validación para SRGAN-B.

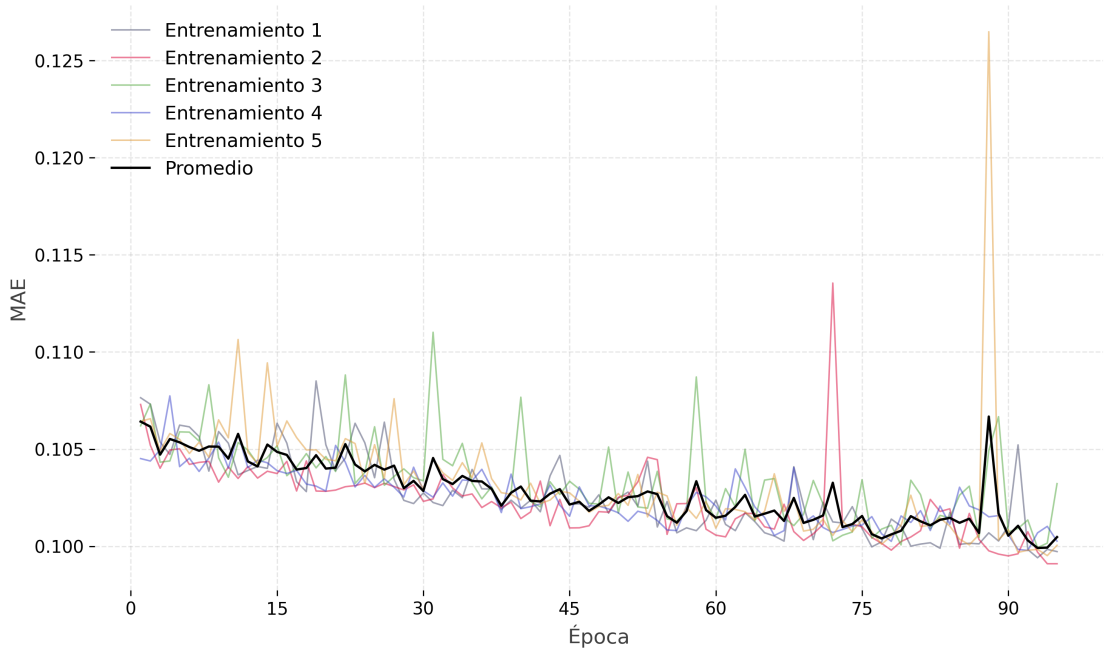


Figura 5.16: Evolución de la métrica MAE en el conjunto de entrenamiento para SRGAN-B.

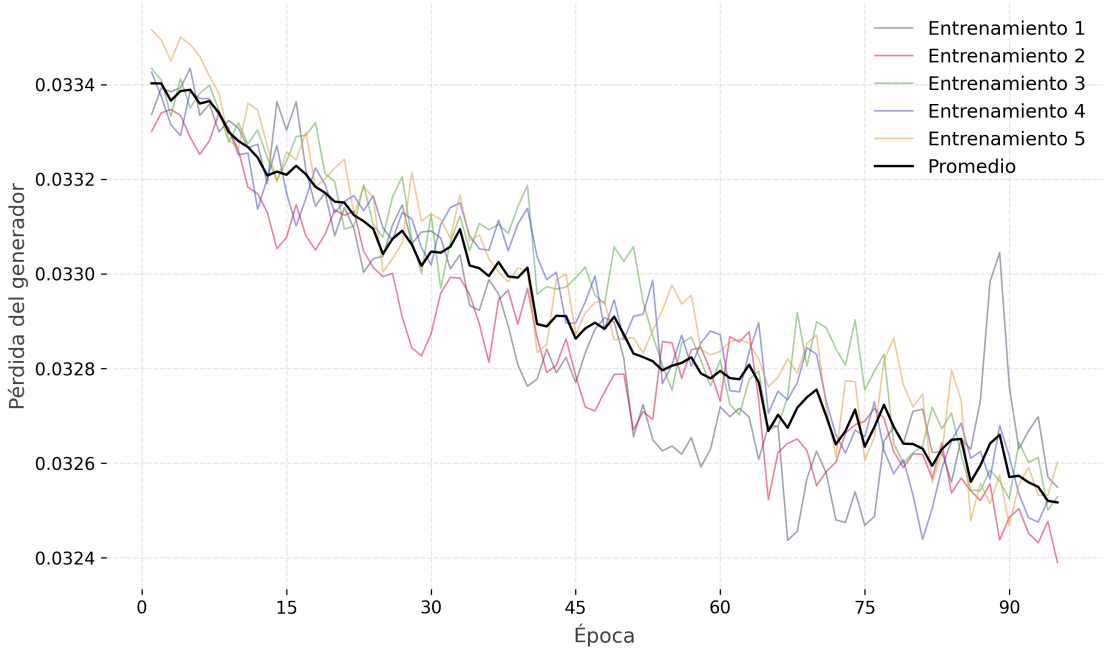


Figura 5.17: Evolución de la pérdida del generador en el conjunto de entrenamiento para SRGAN-B.

Durante los entrenamientos de SRGAN-B la métrica MAE disminuyó en el conjunto de entrenamiento y también en el conjunto de validación. En el conjunto de entrenamiento, la métrica MAE decreció sin estancarse hasta la época 100. En el conjunto de

validación, a diferencia de lo que ocurrió en el conjunto de entrenamiento, la disminución de la métrica MAE se desaceleró levemente en las últimas épocas. La función de pérdida del generador presentó un descenso sostenido con el transcurso de las épocas. A diferencia de SRGAN-A, en SRGAN-B la reducción de la función de pérdida del generador estuvo acompañada por una disminución de la métrica MAE en el conjunto de entrenamiento.

En términos absolutos, el entrenamiento B consiguió mejores resultados que el entrenamiento A. El mejor MAE promedio conseguido en el conjunto de validación fue de 0.0933 para el entrenamiento B, contra 0.156 conseguido en el entrenamiento A, lo que representa 40.2 % de mejora.

5.3.2. Entrenamiento de ESRGAN

En las Figuras 5.18 y 5.19 se gráfica la evolución de la métrica MAE en los conjuntos de validación y entrenamiento respectivamente. En la Figura 5.20 se muestra el resultado de la función de pérdida del generador en cada época durante los cinco entrenamientos de ESRGAN-A.

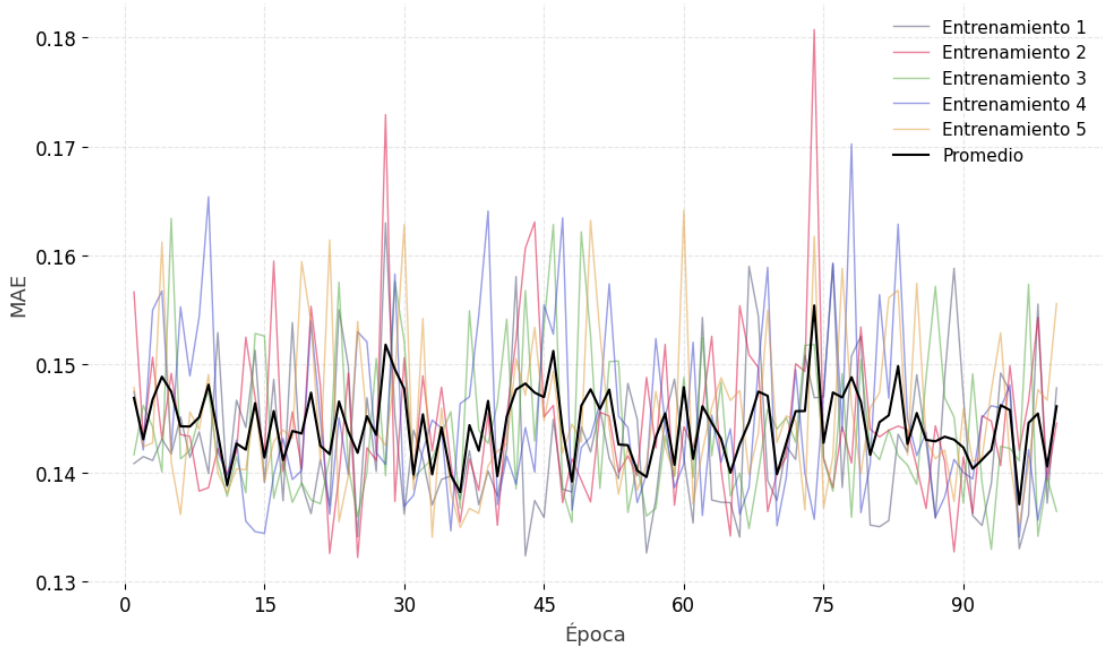


Figura 5.18: Evolución de la métrica MAE en el conjunto de validación para ESRGAN-A

Las Figuras 5.18 y 5.19 no muestran una tendencia decreciente de la métrica MAE en el conjunto de validación ni en el conjunto de entrenamiento, sino que la métrica MAE osciló en torno a un valor constante. Si bien el valor de la función de pérdida disminuyó, el modelo no mejoró en términos de MAE.

La Figura 5.20 muestra una tendencia descendiente que confirmó un correcto proceso de aprendizaje. Lo más importante para el problema estudiado en este proyecto de grado fue evaluar si este proceso de aprendizaje se correspondía con una mejoría en términos de la métrica MAE.

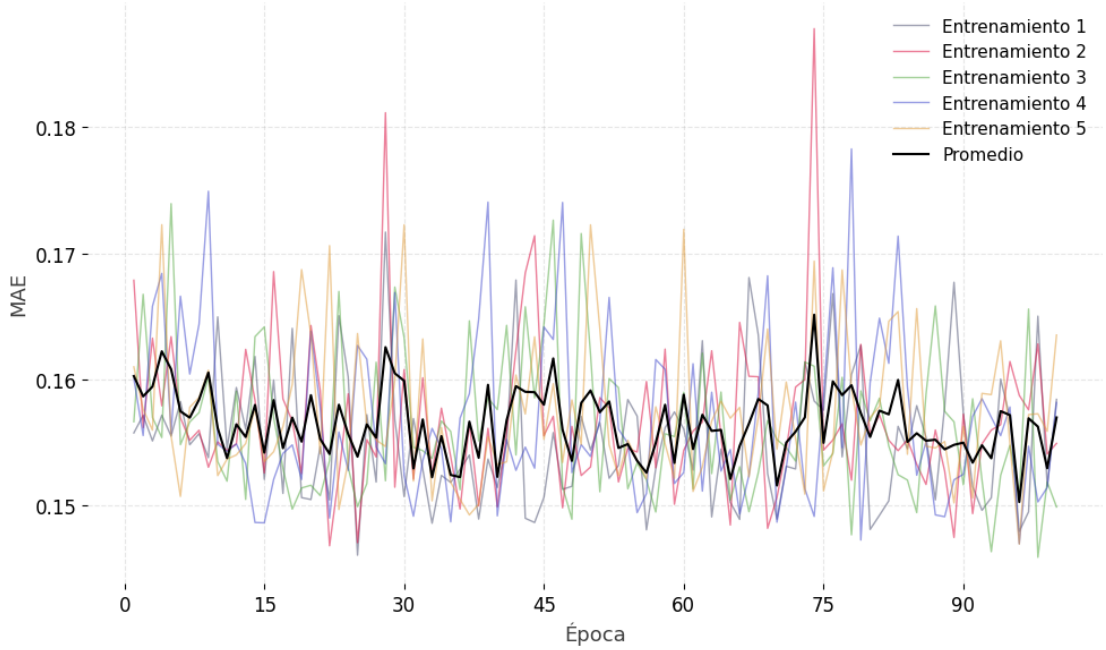


Figura 5.19: Evolución de la métrica MAE en el conjunto de entrenamiento para ESRGAN-A

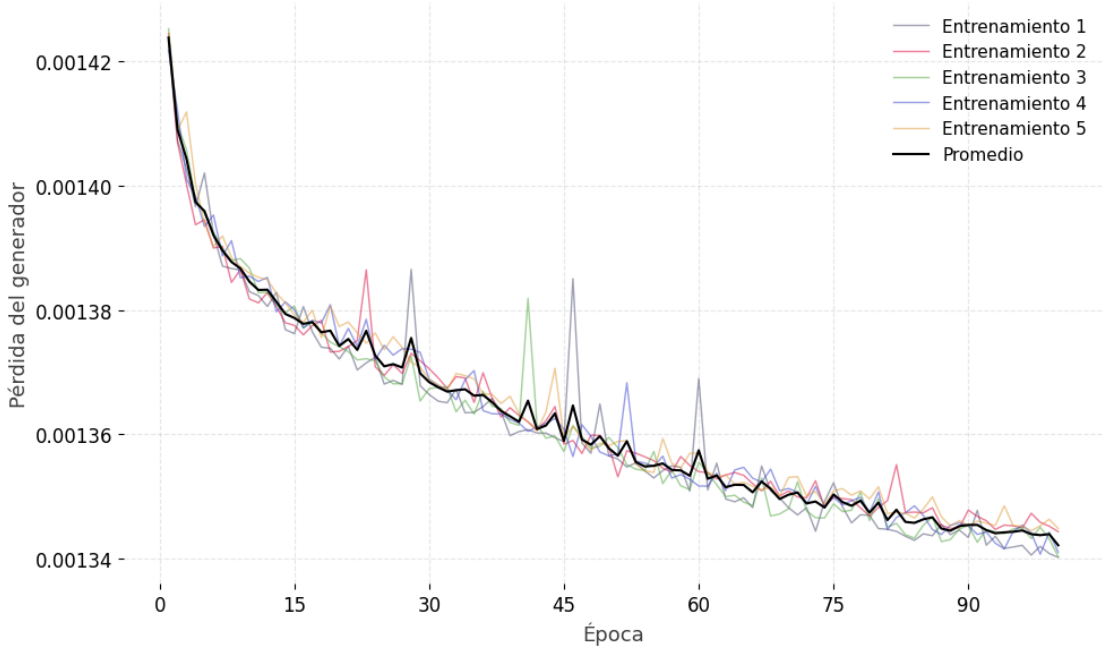


Figura 5.20: Evolución de la pérdida del generador en el conjunto de entrenamiento para ESRGAN-A

Al igual que para SRGAN, en ESRGAN se implementó la variante ESRGAN-B con el tipo de entrenamiento B. La Figura 5.21 muestra la evolución de la métrica MAE en el conjunto de validación y la Figura 5.22 muestra la evolución de la métrica MAE en el conjunto de entrenamiento. La Figura 5.23 muestra la evolución de la función de pérdida del generador en cada época durante los cinco entrenamientos con ESRGAN-B.

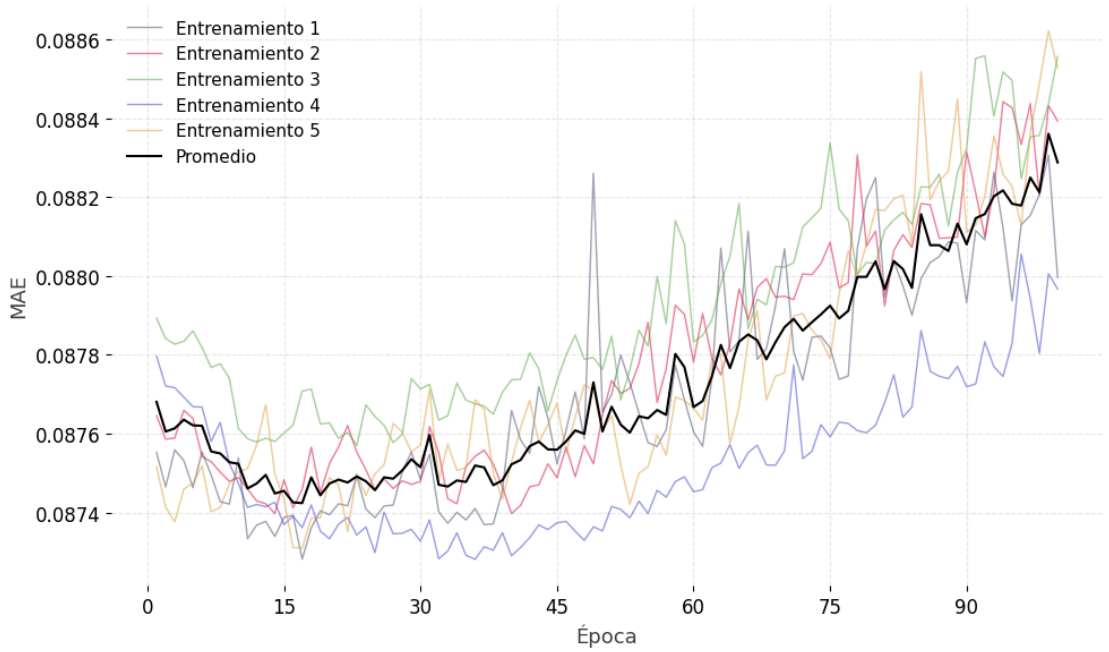


Figura 5.21: Evolución de la métrica MAE en el conjunto de validación para ESRGAN-B

A diferencia de lo que sucedió en el entrenamiento A, en ESRGAN-B hubo una tendencia decreciente de la métrica MAE en el conjunto de entrenamiento. Sin embargo, en el conjunto de validación la métrica MAE decreció hasta la época 17 y luego ascendió hasta la época 100. El ascenso de la métrica MAE en validación se debió a un sobreajuste del modelo a los datos de entrenamiento.

La Figura 5.23 muestra un decrecimiento menos acelerado y más oscilante respecto al entrenamiento A. La pérdida de ESRGAN-B en promedio, se mantuvo casi constante hasta la época 20 y luego comenzó a descender de forma notoria.

A pesar del problema de sobreajuste presentado, en términos absolutos el entrenamiento B consiguió mejores resultados. El mejor MAE promedio en el conjunto de validación fue 0.0874 para el entrenamiento B, en comparación con 0.137 para el entrenamiento A, que representa 36.2 % de mejora.

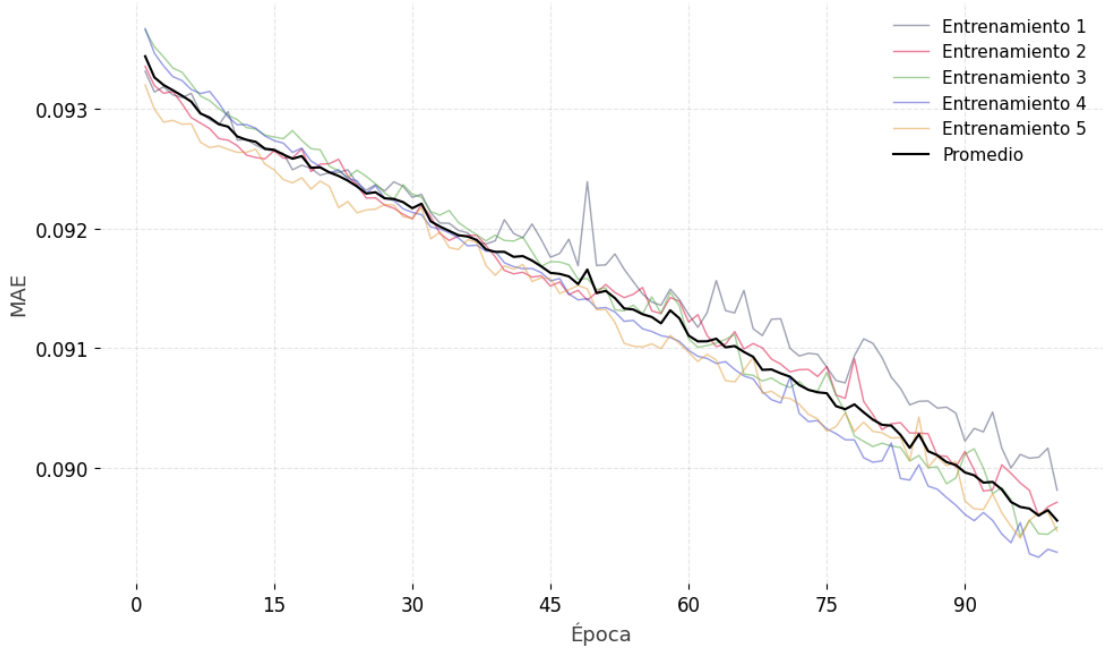


Figura 5.22: Evolución de la métrica MAE en el conjunto de entrenamiento para ESRGAN-B

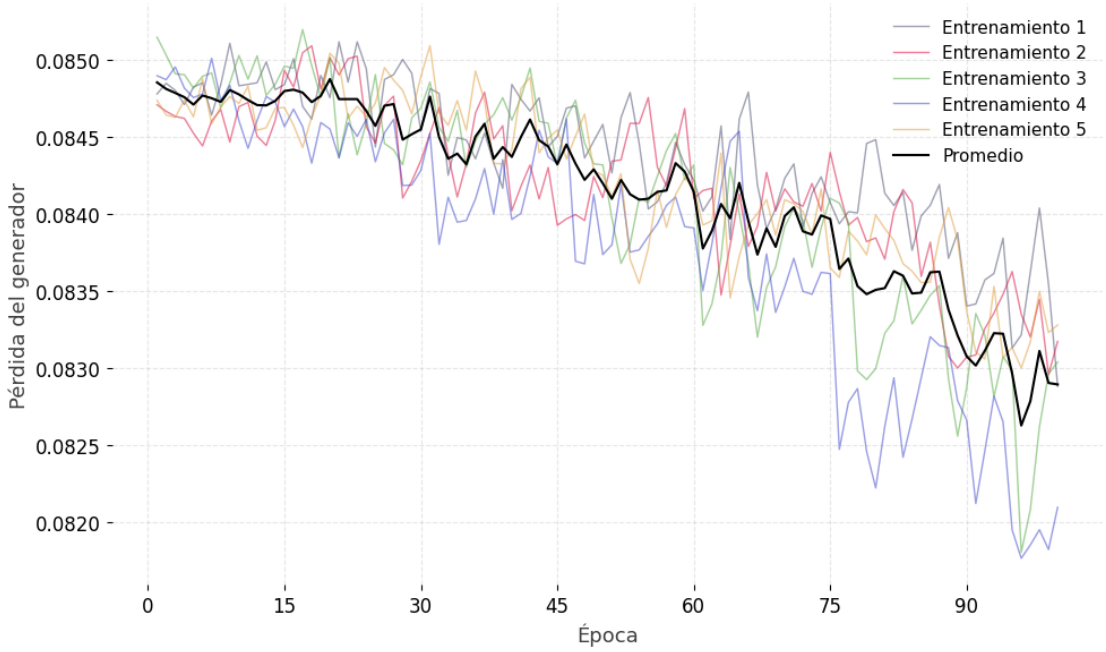


Figura 5.23: Evolución de la pérdida del generador en el conjunto de entrenamiento para ESRGAN-B

5.3.3. Entrenamiento de DSRGAN

En la Figura 5.24 y la Figura 5.25 se muestra la evolución del MAE en el conjunto de validación y en el conjunto de entrenamiento respectivamente durante los cinco entrenamientos de DSRGAN-A. Además, la Figura 5.26 muestra el resultado de la función pérdida del generador en cada época.

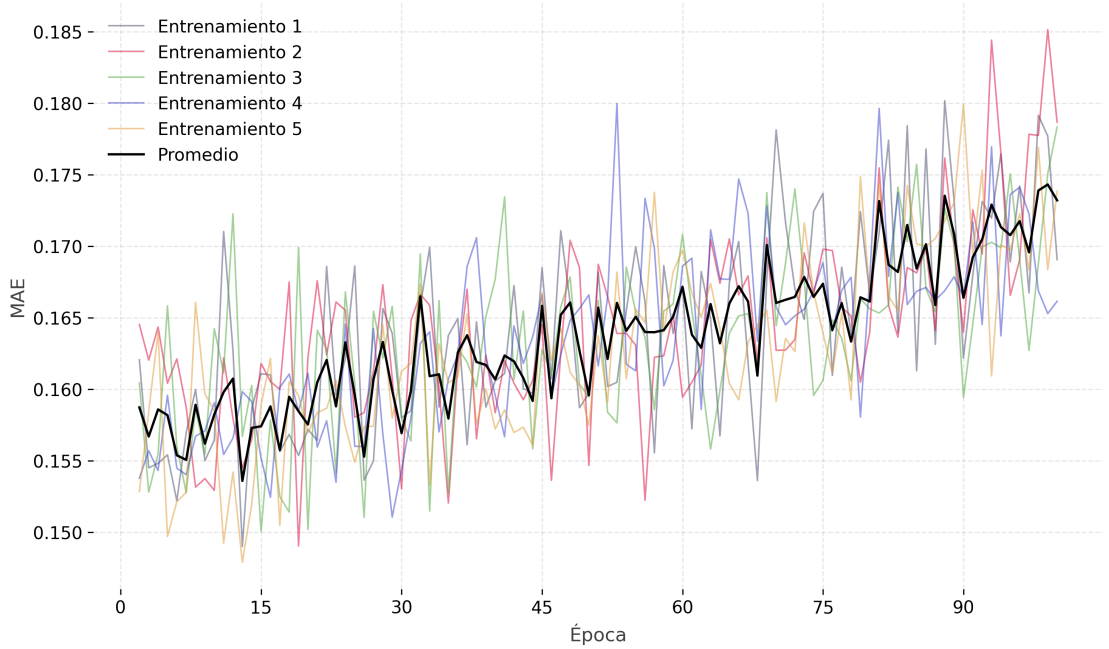


Figura 5.24: Evolución de la métrica MAE en el conjunto de validación para DSRGAN-A.

La evolución del MAE mostró una tendencia similar en los cinco entrenamientos y por lo tanto se utilizó el promedio de los valores obtenidos para los análisis. El promedio del MAE entre los entrenamientos en el conjunto de validación no presentó una mejora significativa sobre lo obtenido en el preentrenamiento. A diferencia de lo que ocurrió con SRGAN, luego de las primeras 13 época el MAE en el conjunto de validación no solo no se estabilizó en ningún valor sino que aumentó consistentemente hasta la época 100. El MAE en el conjunto de entrenamiento se comportó de manera similar a lo que sucedió en el conjunto de validación, donde el mejor valor de MAE se alcanzó en la época 13 y este valor empeoró posteriormente. Al igual que en SRGAN, el aumento del MAE, a pesar de la disminución de la función de pérdida del generador, se explica por la normalización de los DEMs durante el entrenamiento.

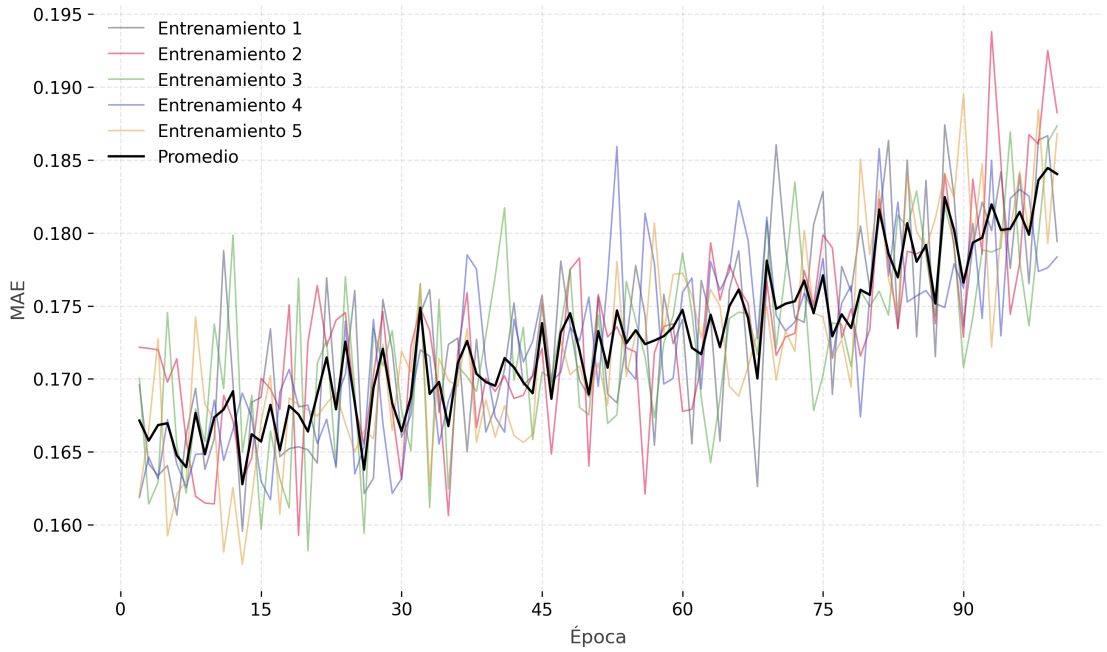


Figura 5.25: Evolución de la métrica MAE en el conjunto de entrenamiento para DSRGAN-A.

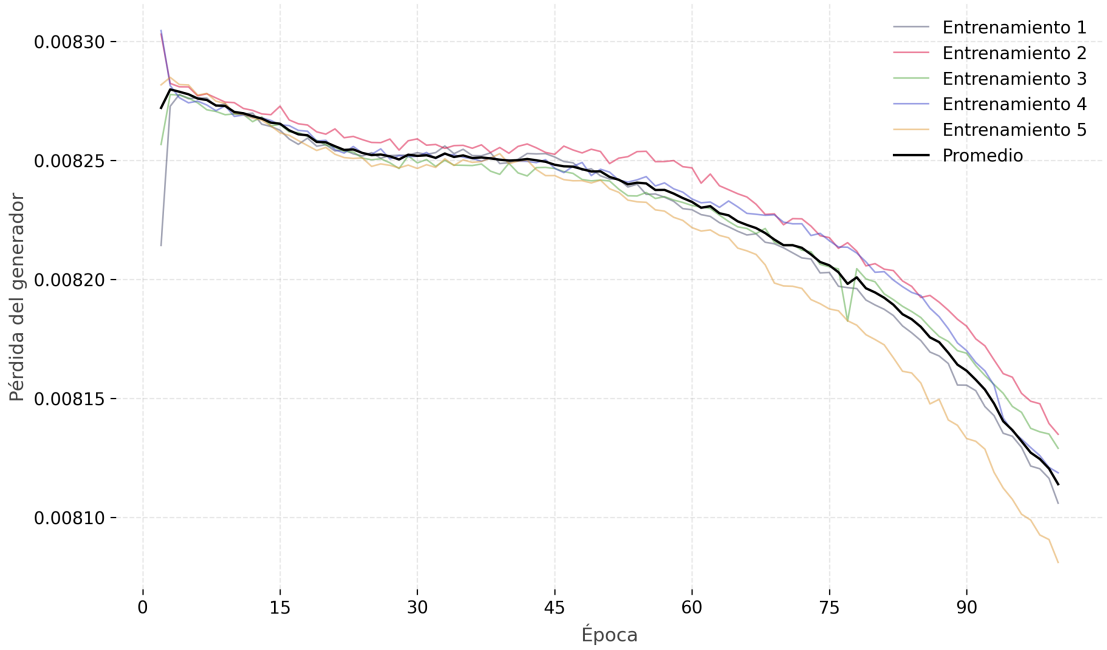


Figura 5.26: Evolución de la pérdida del generador en el conjunto de entrenamiento para DSRGAN-A.

Para DSRGAN-B también se entrenó al modelo durante 100 épocas posteriores al preentrenamiento y se incorporaron las modificaciones en la normalización correspondientes al entrenamiento B. La Figura 5.27 muestra el valor del MAE en el conjunto de validación para cada época de entrenamiento y en la Figura 5.28 se muestra el MAE en el conjunto de entrenamiento para cada época. Finalmente, en la Figura 5.29 se presenta el valor de la función de pérdida del generador para cada época.

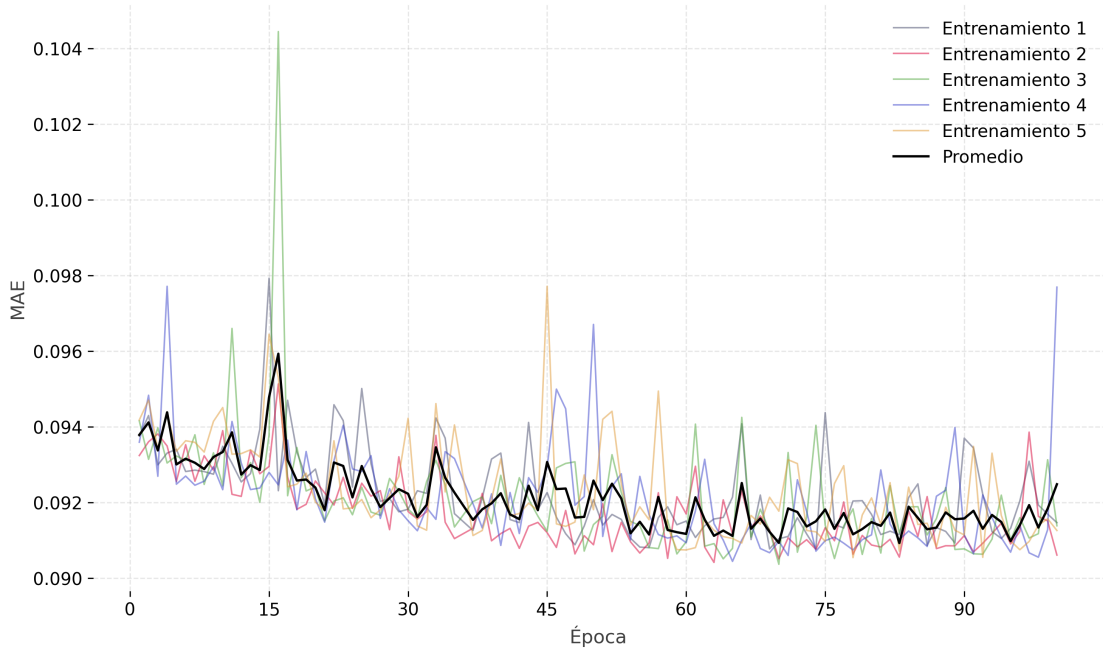


Figura 5.27: Evolución de la métrica MAE en el conjunto de validación para DSRGAN-B.

Para el conjunto de entrenamiento la disminución del MAE fue continua desde el inicio hasta la última época, mientras que para el conjunto de validación el MAE se estancó luego de la época 60. Esta diferencia se debió a que a partir de la época 60 la red comenzó a sobreajustarse a los datos de entrenamiento, lo que mejoró el rendimiento en este conjunto, pero no en el conjunto de validación. El generador de la red DSRGAN logró aprender durante el entrenamiento B, dado que el valor de la función de pérdida disminuyó de forma progresiva a lo largo del entrenamiento.

En DSRGAN-B se evidenció una mejora en comparación con DSRGAN-A en términos de MAE. El mejor MAE promedio en el conjunto de validación fue 0.0909 para el entrenamiento B, en comparación con 0.154 para el entrenamiento A, que representó 41.0 % de mejora.

5.3.4. Entrenamiento de SRGAN_{Z}

Como se mencionó en la Sección 4.5.3, para SRGAN_{Z} se realizaron únicamente 100 épocas de entrenamiento sobre el mejor modelo obtenido de la búsqueda de hiperparámetros. Además, para este entrenamiento no se distinguió entre las variantes de entrenamiento A y B, sino que se entrenó únicamente el mejor modelo de SRGAN_{Z}-B. La decisión de únicamente entrenar con 100 épocas el mejor modelo de SRGAN_{Z}-B se basó en el análisis de tiempos realizado en la Sección 5.2.2.

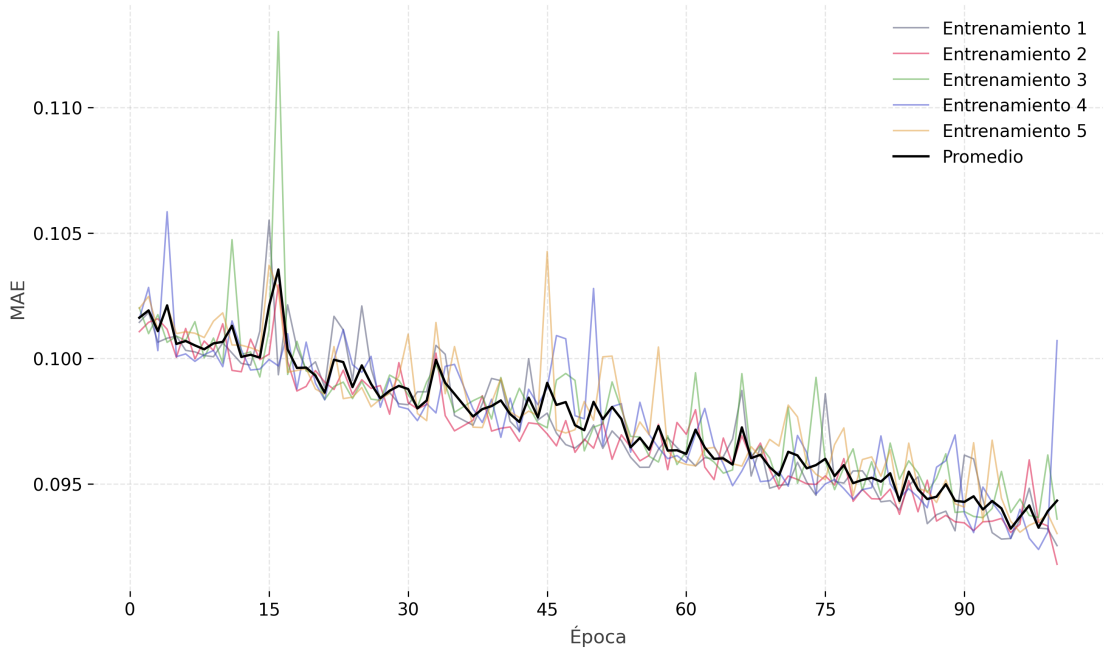


Figura 5.28: Evolución de la métrica MAE en el conjunto de entrenamiento para DSRGAN-B.

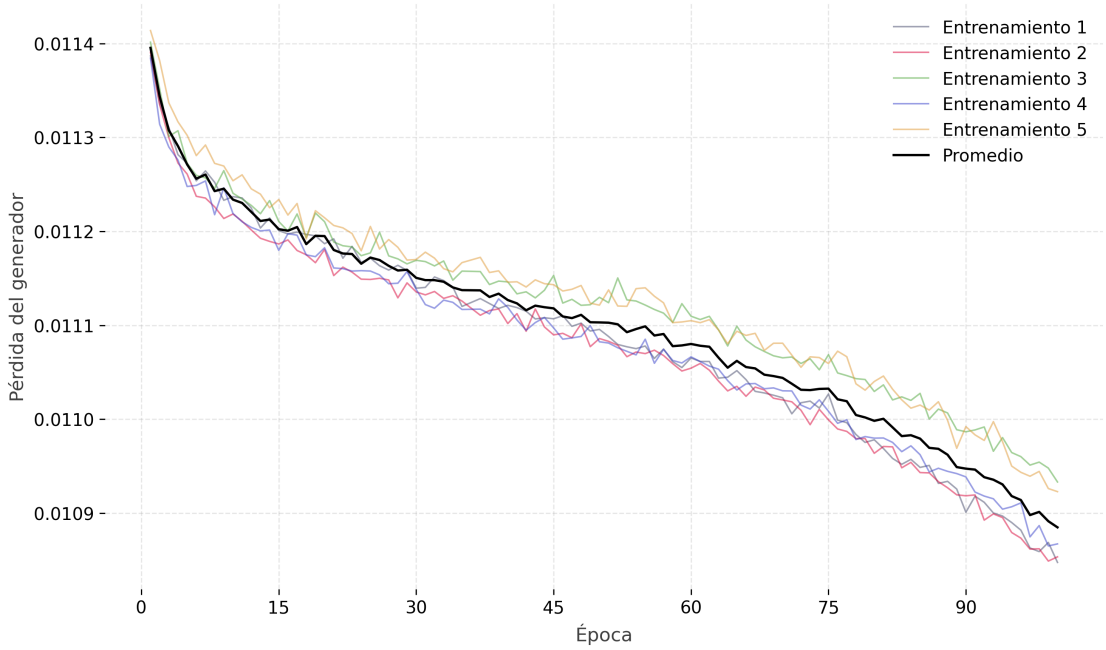


Figura 5.29: Evolución de la pérdida del generador en el conjunto de entrenamiento para DSRGAN-B.

La Figura 5.30 muestra la evolución de la métrica MAE en el conjunto de validación y la Figura 5.31 muestra la evolución de la métrica MAE en el conjunto de entrenamiento durante los cinco entrenamientos de SRGAN$_Z$-B. Además, la Figura 5.32 presenta el resultado de la función de pérdida del generador en cada época.

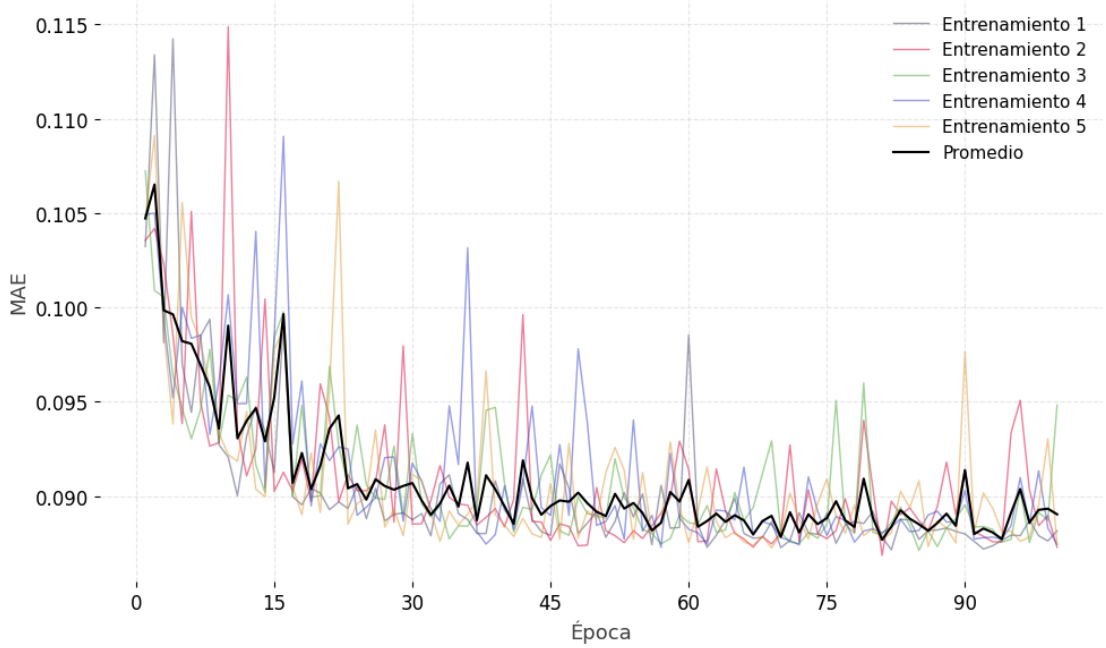


Figura 5.30: Evolución de la métrica MAE en el conjunto de validación para SRGAN$_Z$-B.

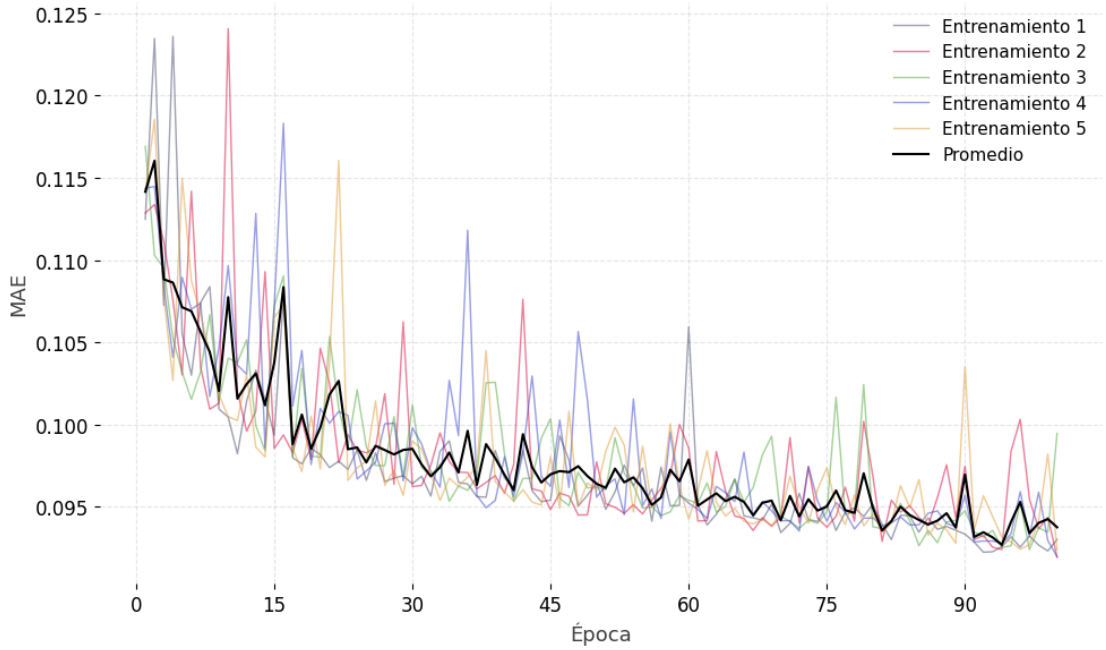


Figura 5.31: Evolución de la métrica MAE en el conjunto de entrenamiento para SRGAN$_Z$-B.

En SRGAN$_Z$-B, la pérdida del generador mejoró durante todo el entrenamiento, además a lo largo de los entrenamientos la métrica MAE disminuyó en el conjunto de entrenamiento y también en el conjunto de validación. En el conjunto de entrenamiento, la métrica MAE decreció sin estancarse hasta la época 100. En contraste, en el conjunto de validación la disminución de la métrica MAE se desaceleró aproximadamente en la época 30. Esta diferencia indicó que con la cantidad de datos e hiperparámetros utilizados, entrenar al modelo más allá de aproximadamente 30 épocas no se traduce en una mejora de los resultados.

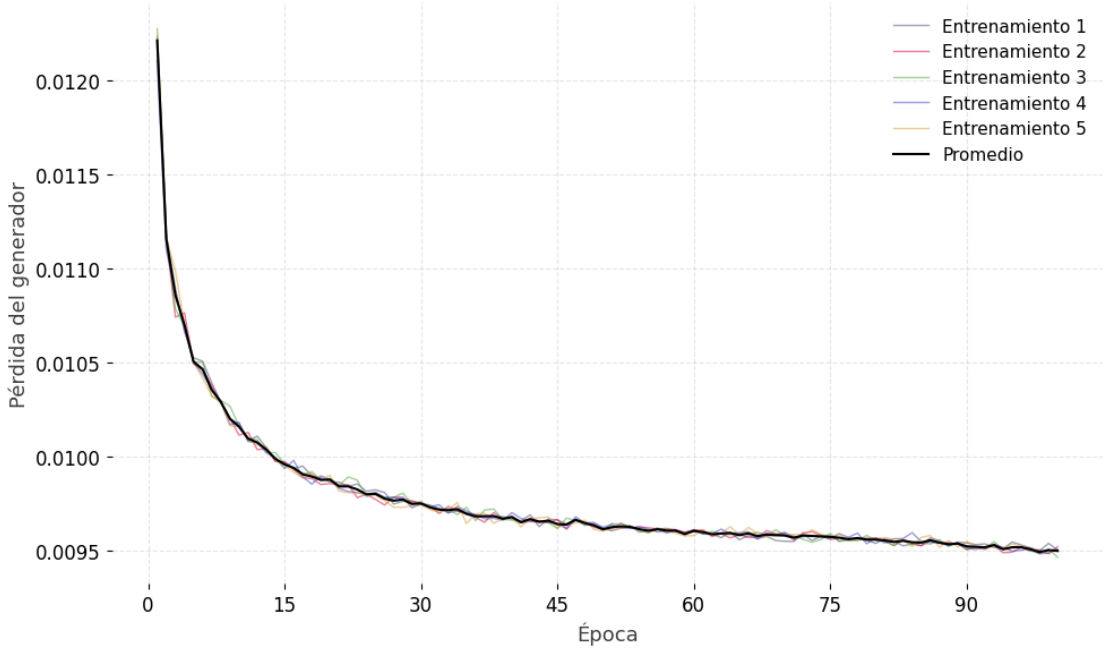


Figura 5.32: Evolución de la pérdida del generador en el conjunto de entrenamiento para SRGAN$_Z$-B

El mejor MAE promedio en el conjunto de validación fue 0.0877 para SRGAN$_Z$ en el entrenamiento B.

5.4. Comparación de resultados

Luego de los entrenamientos en profundidad se realizó una comparación entre los resultados de las métricas obtenidos en cada arquitectura. Para calcular las métricas en cada arquitectura, se seleccionó la época con menor MAE promedio entre los cinco entrenamientos ejecutados. Posteriormente se promediaron las métricas obtenidas en la época seleccionada para los cinco entrenamientos. En la Tabla 5.24 se presentan los resultados de comparar las métricas MAE, RMSE, max error, LE90, precision y recall calculadas en promedio.

Los valores de las métricas que se obtuvieron durante los entrenamientos corresponden a la evaluación de los modelos realizada en el conjunto de validación y utilizando su tamaño de crop correspondiente. Dado que todos los mejores modelos fueron entrenados en crops de 16×16, no se realizó una posterior evaluación en crops de 96×96 con ventana deslizante como si se realizó para la búsqueda de hiperparámetros. Como se utilizó

modelo	MAE	RMSE	max error	LE90	precision	recall
SRGAN-B	0.0933	0.254	**2.87**	0.180	0.992	**0.975**
ESRGAN-B	**0.0874**	**0.252**	**2.87**	0.164	**0.999**	0.912
DSRGAN-B	0.0909	0.254	2.89	0.172	0.994	0.887
SRGAN_{Z}-B	0.0877	0.264	2.96	**0.159**	**0.999**	0.941

Tabla 5.24: Comparación de las métricas MAE, RMSE, max error, LE90, precision y recall para los mejores modelos entrenados en profundidad.

un conjunto de evaluación distinto al utilizado en la búsqueda de hiperparámetros, los valores de las métricas obtenidos en los entrenamientos en profundidad y la búsqueda de hiperparámetros no son comparables.

No existieron grandes diferencias entre los resultados obtenidos por las arquitecturas evaluadas. Con respecto a la métrica MAE, el valor más bajo fue alcanzado por ESRGAN-B, con 6.3 % de mejora en comparación con el valor más alto, alcanzado por SRGAN-B. Con respecto a la métrica RMSE, el valor más bajo también fue alcanzado por ESRGAN-B, con 4.5 % de mejora en comparación con el valor más alto, alcanzado por SRGAN_{Z}-B. Por último, con respecto a la métrica LE90, el valor más bajo fue alcanzado por SRGAN_{Z}-B, con 11.7 % de mejora en comparación con el valor más alto, alcanzado por SRGAN-B.

La arquitectura ESRGAN-B superó a las demás arquitecturas en dos de las tres métricas principales (MAE y RMSE) y quedó en segundo lugar en la tercera métrica principal (LE90). Por esta razón se seleccionó a ESRGAN-B personalizado como el mejor modelo y se utilizó para aplicar las técnicas de aumento de datos propuestas.

5.5. Aumento de datos

En esta sección se presentan los resultados de aplicar en el mejor modelo de la arquitectura ESRGAN-B personalizada las dos técnicas de aumento de datos sobre el conjunto de entrenamiento descritas en la Sección 4.5.4. Se llevaron a cabo cinco entrenamientos para cada técnica de aumento de datos. Para la primera técnica de aumento de datos se ejecutaron 15 épocas de preentrenamiento seguidas de 100 épocas de entrenamiento. Para la segunda técnica aumento de datos se ejecutaron 5 épocas de preentrenamiento seguidas de 30 épocas de entrenamiento. La diferencia entre la cantidad de épocas entre ambas técnicas se debió al incremento en el número de instancias de entrenamiento generado por la aplicación de un stride menor al tamaño de los crops en la segunda técnica. Los hiperparámetros utilizados corresponden a los mismos que se usaron para los entrenamientos en profundidad con ESRGAN-B personalizado.

La Figura 5.33 muestra los resultados obtenidos en el conjunto de validación para la primera técnica de aumento de datos.

Con la primera técnica de aumento de datos el MAE promedio mejoró hasta la época 66, donde alcanzó su punto más bajo con un valor de 0.0909. El menor MAE se alcanzó en el segundo entrenamiento realizado en la época 86, con un valor de 0.0906. Sin embargo, este valor de MAE fue mayor que el obtenido en el entrenamiento sin aumento de datos, que fue 0.0873.

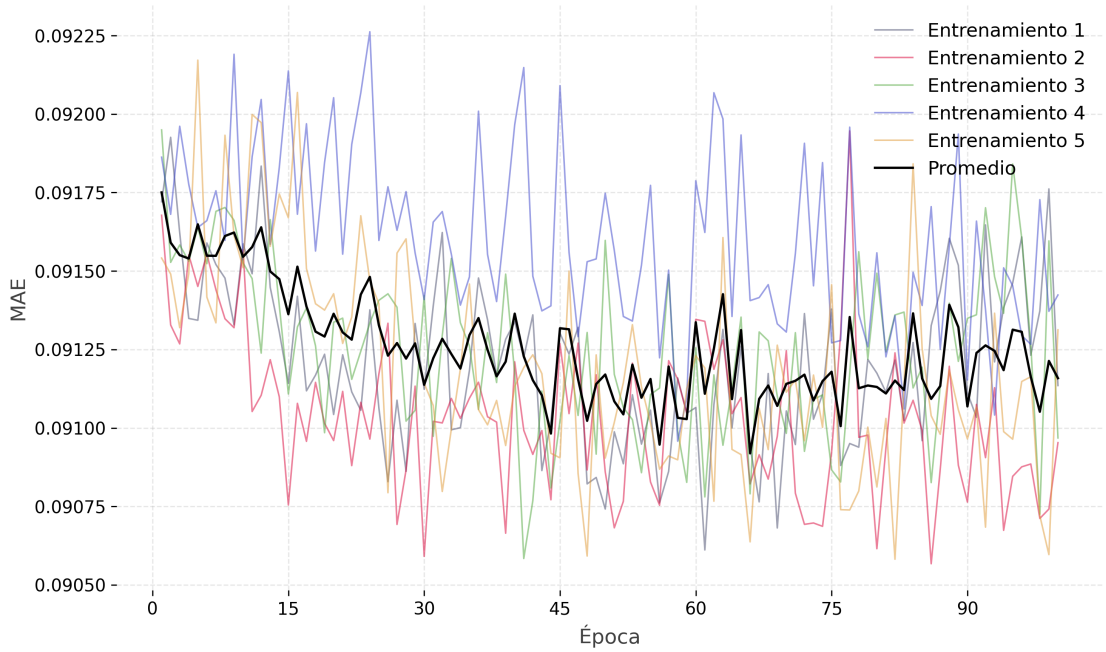


Figura 5.33: Evolución de la métrica MAE en el conjunto de validación para ESRGAN-B con la primera técnica de aumento de datos.

A diferencia de los entrenamientos sin aumento de datos de ESRGAN-B en los que el MAE promedio mejoró solo hasta la época 17, con la primera técnica de aumento de datos el descenso continuó durante un número significativamente mayor de épocas. Incluso luego de alcanzar su mínimo, el aumento posterior fue lento.

La Figura 5.34 presenta los resultados obtenidos en el conjunto de validación para la segunda técnica de aumento de datos. Se utilizaron crops de 16×16 para los entrenamientos con la segunda técnica y, por lo tanto, se consideró el factor de incremento de la Ecuación 4.7 para igualar el número de iteraciones con respecto a los entrenamientos anteriores. Se decidió definir en 5 la cantidad de épocas de preentrenamiento y en 30 la cantidad de épocas de entrenamiento.

Con la segunda técnica de aumento de datos los resultados fueron superiores a los obtenidos con la primera técnica de aumento de datos. El MAE promedio disminuyó hasta la época 16 en la cual alcanzó el valor mínimo de 0.0867. A partir de la época 16 se dio un crecimiento lento del MAE promedio hasta las últimas épocas, en las que este aumento se aceleró en todos los entrenamientos.

El quinto entrenamiento con la segunda técnica de aumento de datos alcanzó el MAE más bajo, con un valor de 0.0863 en la época 25. Este valor fue el mejor obtenido por un modelo en el conjunto de validación en este proyecto de grado.

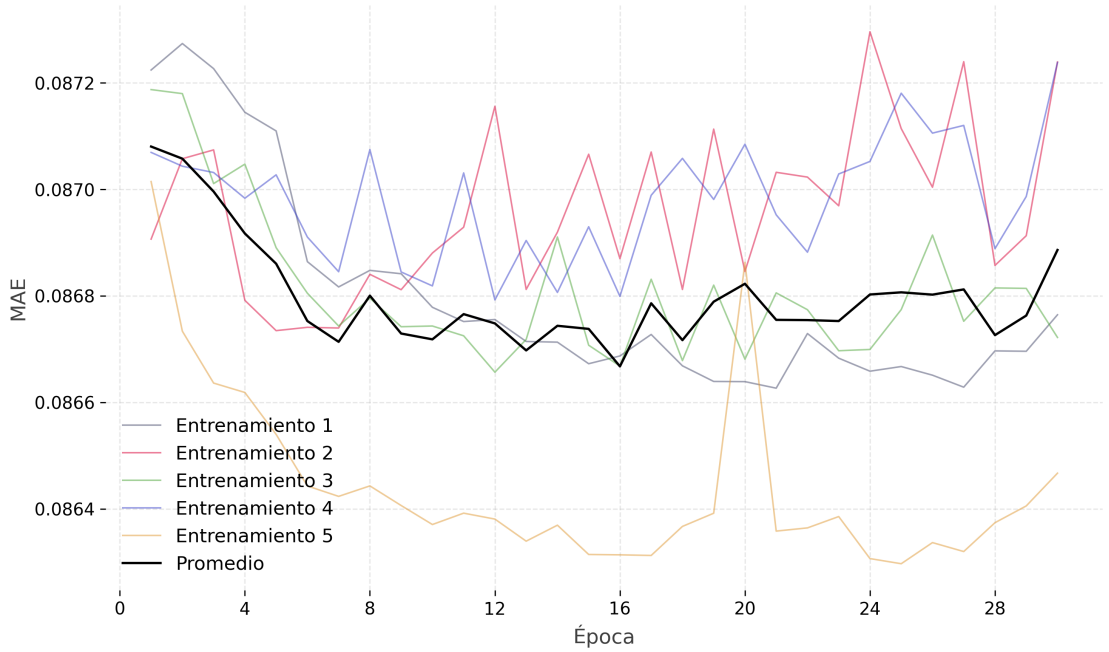


Figura 5.34: Evolución de la métrica MAE en el conjunto de validación para ESRGAN-B con la segunda técnica de aumento de datos.

5.6. Análisis del mejor modelo: ESRGAN-B personalizado

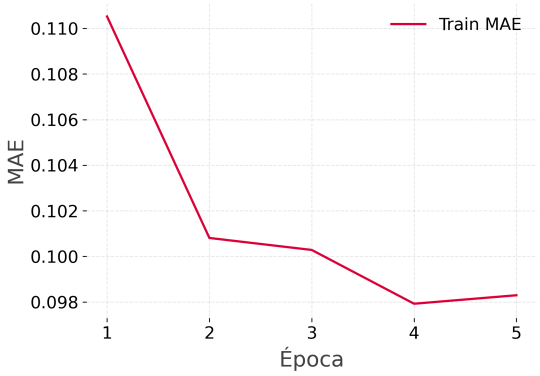
Esta sección presenta un análisis detallado de los resultados obtenidos con el modelo ESRGAN-B personalizado entrenado con la segunda técnica de aumento de datos. En particular, se analizó la evolución de las métricas MAE, RMSE, LE90, max error, precision y recall, así como la evolución de las funciones de pérdida del discriminador y del generador. Además, se reportaron las mismas métricas para el preentrenamiento del modelo.

5.6.1. Entrenamiento del modelo ESRGAN-B personalizado

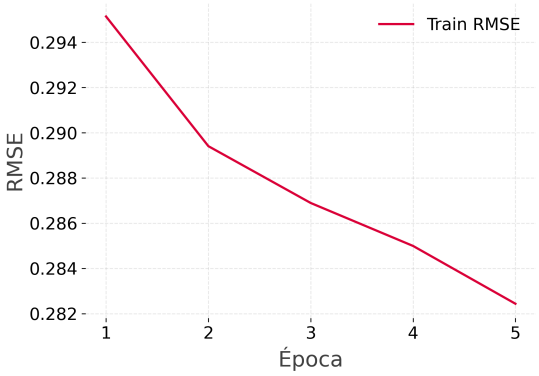
Con el objetivo de facilitar la interpretación de los resultados, las gráficas correspondientes al conjunto de entrenamiento se muestran en color rojo, mientras que las gráficas correspondientes al conjunto de validación se presentan en color azul.

La Figura 5.35 muestra la evolución de las métricas durante el preentrenamiento del modelo en el conjunto de entrenamiento.

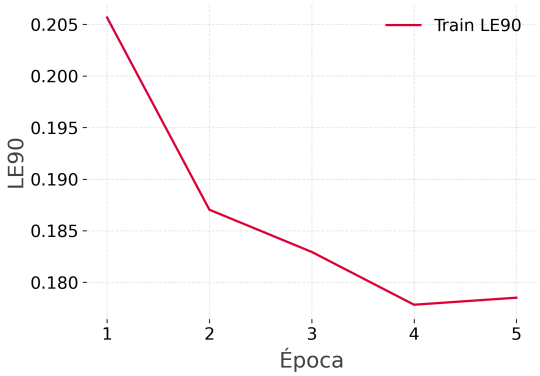
El modelo mostró mejoras en todas las métricas que cuantifican el error en las celdas con datos. Sin embargo, en las métricas de precision y recall, el rendimiento del modelo se redujo y ambas métricas alcanzaron el valor 0.0. Esta caída se debió a que durante el preentrenamiento se utilizó únicamente el MAE como función de pérdida del generador y por lo tanto no se consideraron a las celdas sin datos en su cálculo. Como consecuencia, las predicciones del modelo se limitaron al rango (0, 1), correspondiente a los valores presentes en las celdas con datos, que fueron los únicos valores que el modelo aprendió. Como el modelo predijo todas las celdas como celdas con datos, la precision y el recall se redujeron a 0.0.



(a) Evolución de la métrica MAE

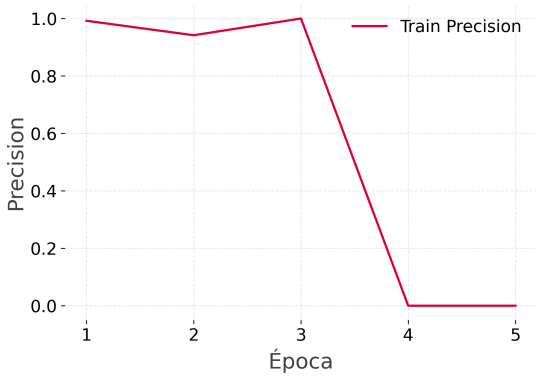


(b) Evolución de la métrica RMSE



(c) Evolución de la métrica LE90

(d) Evolución de la métrica max error

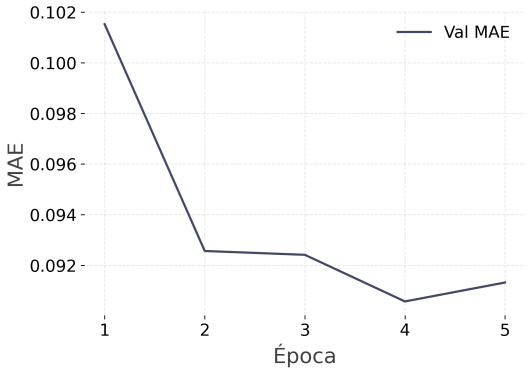


(e) Evolución de la métrica precision

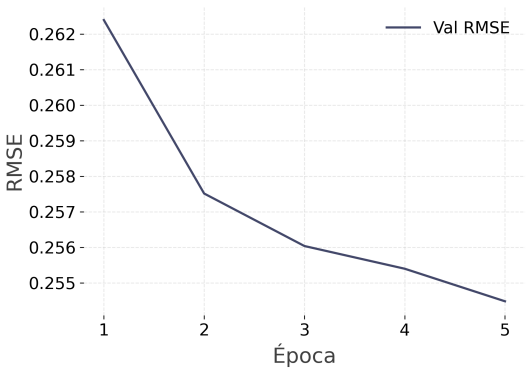
(f) Evolución de la métrica recall

Figura 5.35: Evolución de las métricas de evaluación en el conjunto de entrenamiento durante el preentrenamiento para ESRGAN-B personalizado.

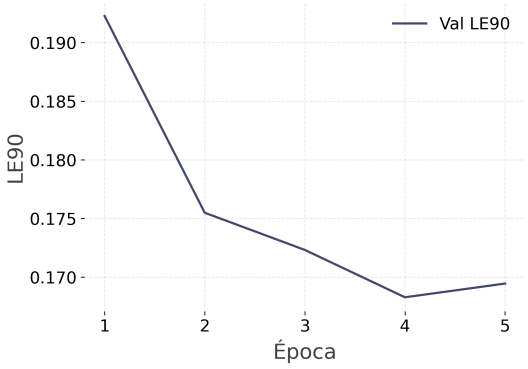
La Figura 5.36 muestra la evolución de las métricas MAE, RMSE, LE90, max error, precision y recall durante el preentrenamiento del modelo en el conjunto de validación.



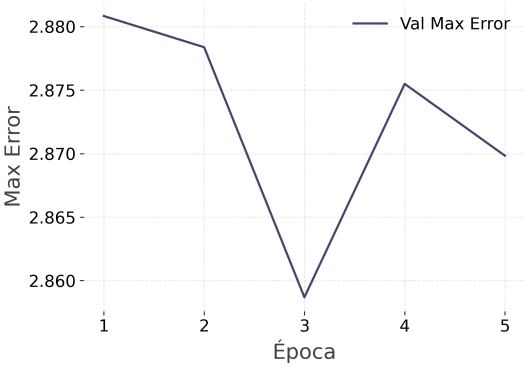
(a) Evolución de la métrica MAE



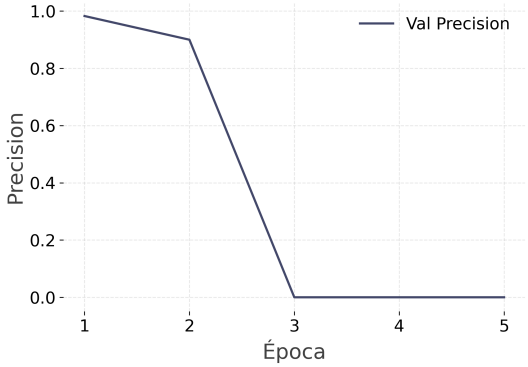
(b) Evolución de la métrica RMSE



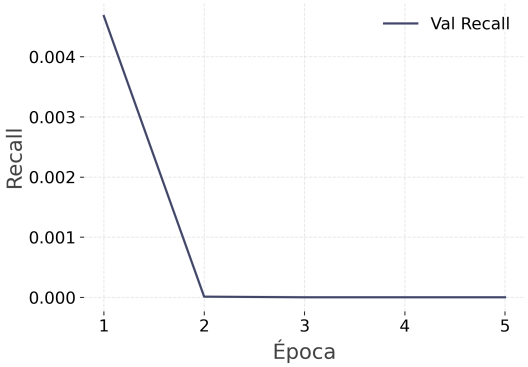
(c) Evolución de la métrica LE90



(d) Evolución de la métrica max error



(e) Evolución de la métrica precision



(f) Evolución de la métrica recall

Figura 5.36: Evolución de las métricas de evaluación en el conjunto de validación durante el preentrenamiento para ESRGAN-B personalizado.

Durante el preentrenamiento, las métricas MAE, RMSE, LE90, max error, precision y recall evaluadas en el conjunto de validación siguieron tendencias muy similares a las mostradas en el conjunto de entrenamiento. Las métricas MAE, RMSE, LE90 y max error mostraron mejoras mientras que los valores de las métricas precision y recall, al igual que en el conjunto de entrenamiento, alcanzaron un valor de 0.0.

La Figura 5.37 muestra la evolución de las métricas MAE, RMSE, LE90, max error, precision y recall durante el entrenamiento del modelo en el conjunto de entrenamiento.

Desde la primera época del entrenamiento, los resultados obtenidos superaron a los alcanzados al final del preentrenamiento en todas las métricas. La métrica MAE, que en la última época del preentrenamiento fue de 0.0983, se redujo a 0.0932 en la primera época del entrenamiento. Las métricas alcanzaron en la última época del entrenamiento mejores valores a los observados al inicio del entrenamiento.

(a) Evolución de la métrica MAE

(b) Evolución de la métrica RMSE

(c) Evolución de la métrica LE90

(d) Evolución de la métrica max error

(e) Evolución de la métrica precision

(f) Evolución de la métrica recall

Figura 5.37: Evolución de las métricas de evaluación en el conjunto de entrenamiento durante el entrenamiento para ESRGAN-B personalizado.

Desde la primera época del entrenamiento, los resultados obtenidos superaron a los alcanzados al final del preentrenamiento en todas las métricas. La métrica MAE, que en la última época del preentrenamiento fue de 0.0983, se redujo a 0.0932 en la primera época del entrenamiento. Las métricas alcanzaron en la última época del entrenamiento mejores valores a los observados al inicio del entrenamiento.

Las métricas de precision y recall, que finalizaron el preentrenamiento con un valor de 0,0, mostraron una mejora sustancial al alcanzar 0,9998 y 0,4256 al final de la primera época del entrenamiento, respectivamente. Estas mejoras se debieron a la incorporación del discriminador, que mediante la adversarial loss, corrigió la tendencia del generador a clasificar todas las celdas como celdas con datos. Si bien la métrica precision alcanzó un valor muy alto tras el preentrenamiento, presentó una leve disminución hacia el final del entrenamiento y descendió a 0,9995, un valor elevado de todas formas. En cambio, la métrica recall comenzó en un valor significativamente más bajo y luego aumentó de forma sostenida a lo largo de las épocas hasta alcanzar el valor 0,8940 al final del entrenamiento.

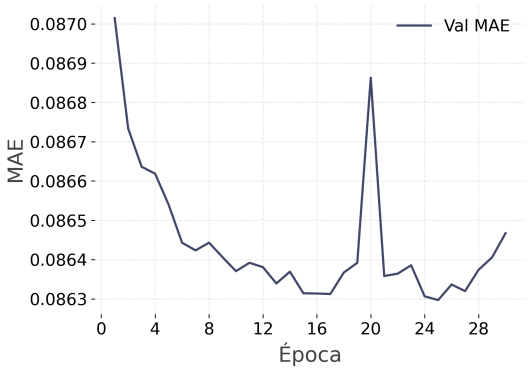
La Figura 5.38 muestra la evolución de las métricas MAE, RMSE, LE90, max error, precision y recall durante el entrenamiento del modelo en el conjunto de validación.

A diferencia de lo que sucedió en el conjunto de entrenamiento, en el conjunto de validación las métricas MAE y LE90 siguieron una tendencia descendente hasta aproximadamente la época 21, a partir de la cual incrementaron su valor. En contraste, las métricas RMSE y max error mostraron una tendencia ascendente durante todo el entrenamiento. Las métricas precision y recall mostraron resultados similares a los presentados en el conjunto de entrenamiento.

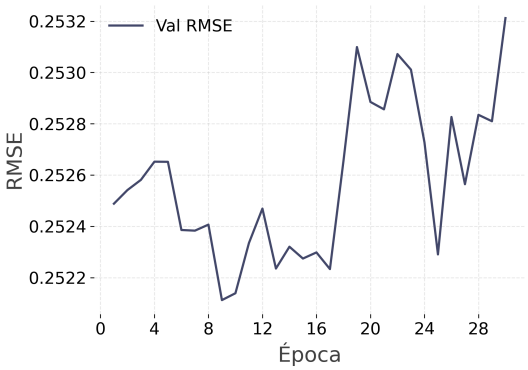
La mejora del modelo en el conjunto de validación durante el entrenamiento fue menos pronunciada que en el preentrenamiento. La reducción de la métrica MAE entre la primera época y el mejor valor alcanzado fue del 0,8 %, mientras que el incremento en RMSE fue del 0,28 %.

La Figura 5.39 muestra la evolución de las funciones de pérdida del discriminador y del generador durante el entrenamiento del modelo.

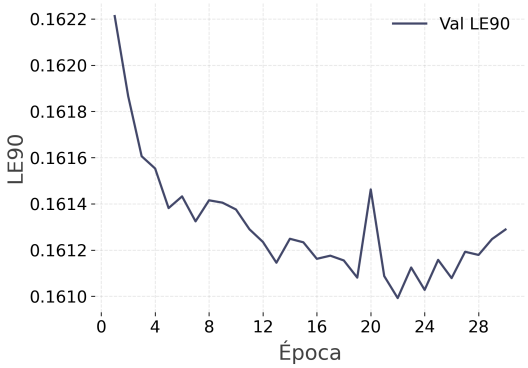
El generador y el discriminador redujeron sus respectivas funciones de pérdida durante el entrenamiento, lo que indicó que se produjo un aprendizaje en ambos componentes de la GAN. La función de pérdida del generador presentó un valor inicial de 8.5×10^{-2} al finalizar la primera época y un valor de 8.4×10^{-2} en la última época. En el caso del discriminador, la función de pérdida disminuyó desde 2.9×10^{-2} al finalizar la primera época a 2.1×10^{-4} en la última época. La diferencia en la magnitud de las reducciones se explica por el hecho de que el generador ya había sido preentrenado y el discriminador no.



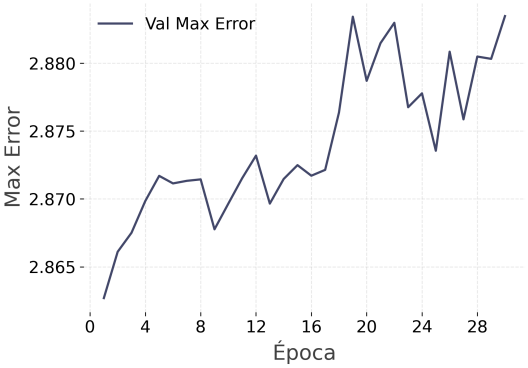
(a) Evolución de la métrica MAE



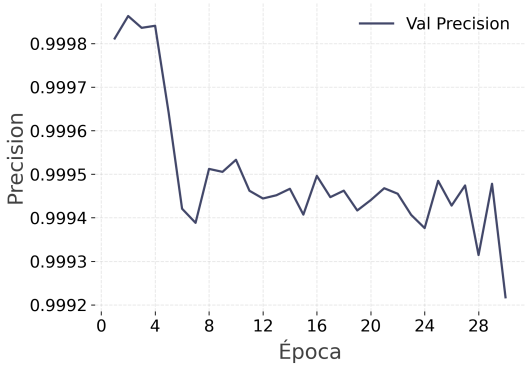
(b) Evolución de la métrica RMSE



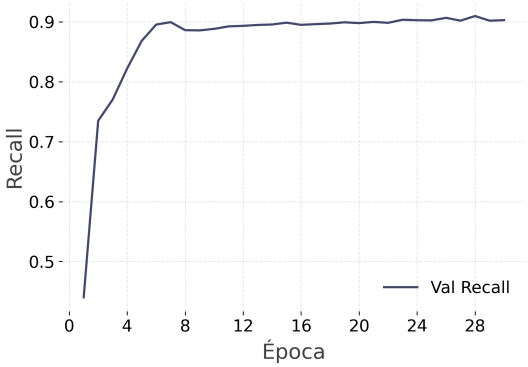
(c) Evolución de la métrica LE90



(d) Evolución de la métrica max error

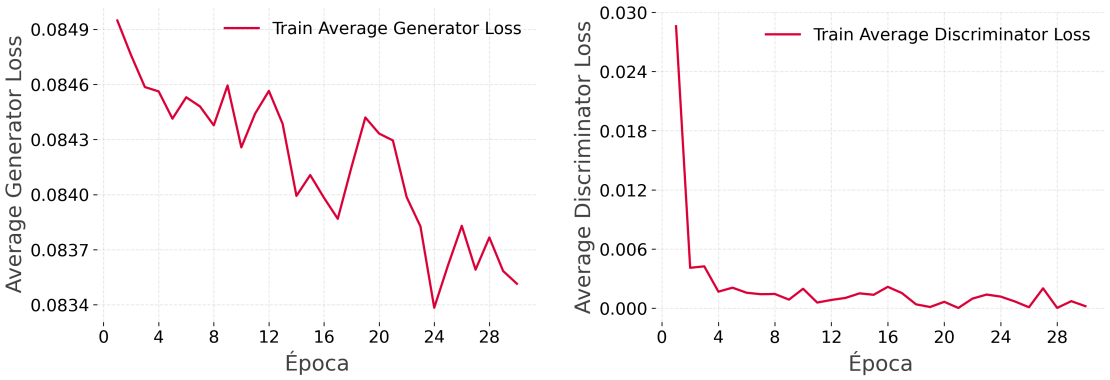


(e) Evolución de la métrica precision



(f) Evolución de la métrica recall

Figura 5.38: Evolución de las métricas MAE, RMSE, LE90, error máximo, precision y recall en el conjunto de validación durante el entrenamiento para ESRGAN-B personalizado.



(a) Evolución de la pérdida del generador

(b) Evolución de la pérdida del discriminador

Figura 5.39: Evolución de las funciones de pérdida del generador y del discriminador durante el entrenamiento de ESRGAN-B personalizado.

5.6.2. Mecanismo de ventana deslizante

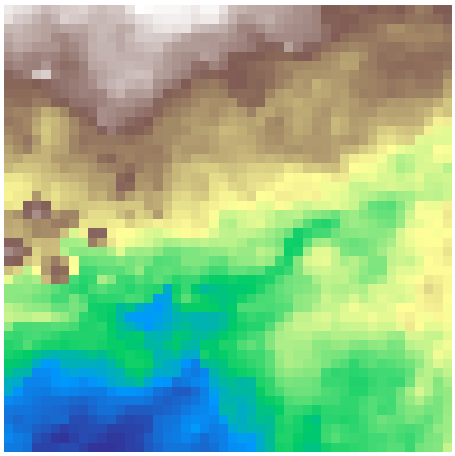
Para realizar la superresolución de un crop se utilizó el método de ventana deslizante descrito en la Sección 4.6.1. Este método permitió al modelo reducir el error en los crops superresueltos. La Figura 5.40 muestra una comparación visual y cuantitativa de los resultados obtenidos por el mejor modelo de la arquitectura ESRGAN-B personalizada al aplicar superresolución con y sin el método de ventana deslizante.

Se presentó un crop en baja resolución (LR) y su correspondiente en alta resolución (HR). Para este crop se mostró los resultados de aplicar superresolución al crop LR sin el método de ventana deslizante a la izquierda y con el método de ventana deslizante a la derecha. Además, se presentaron los errores absolutos para cada reconstrucción y una barra con la diferencia de elevación en metros de los errores absolutos.

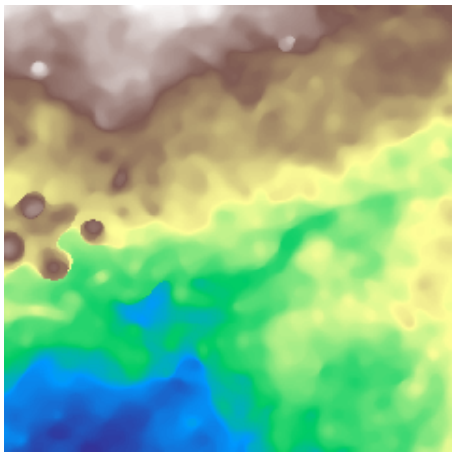
La comparación de la Figura 5.40 se realizó sobre un crop en baja resolución de tamaño 48×48. El tamaño de ventana elegido para el procesamiento fue de 16×16 dado que el modelo utilizado para la comparación fue entrenado con crops de tamaño 16×16. Para el método de ventana deslizante se utilizó un stride de 8 celdas, un 50 % del tamaño de la ventana.

Los resultados mostraron diferencias significativas entre ambos métodos, especialmente en los bordes de las ventanas cuando no se utilizó el mecanismo de ventana deslizante. Esto se debe a que en los bordes de las ventanas el modelo tiene menos información para hacer una buena predicción y, por lo tanto, se generó una discontinuidad en la elevación en el método sin ventana deslizante. Esta discontinuidad no fue apreciable cuando se aplicó el mecanismo de ventana deslizante.

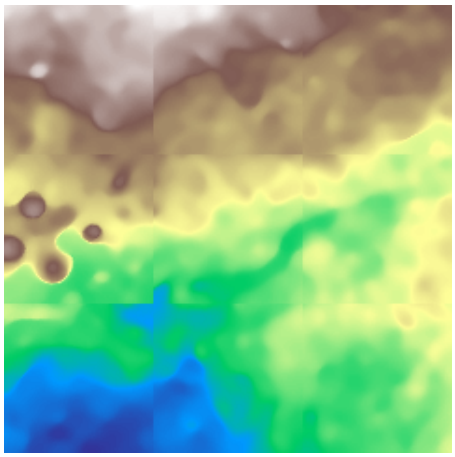
Desde una perspectiva cuantitativa, el uso del método de ventana deslizante redujo la métrica MAE en un 10.6 % y la métrica RMSE en un 7.7 % en el crop superresuelto. Esta mejora es significativa si se considera que se utilizó el mismo modelo para ambas reconstrucciones, lo que resalta la efectividad del enfoque propuesto.



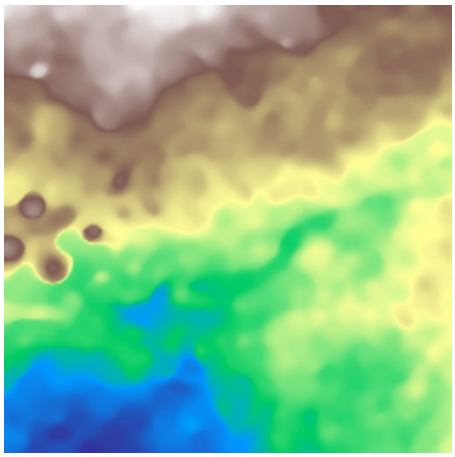
(a) crop en baja resolución



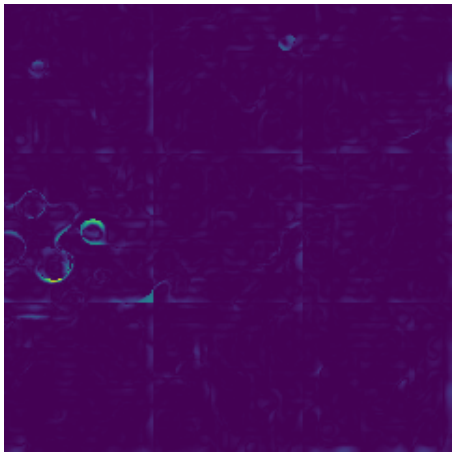
(b) crop en alta resolución



(c) crop superresuelto sin ventana deslizante

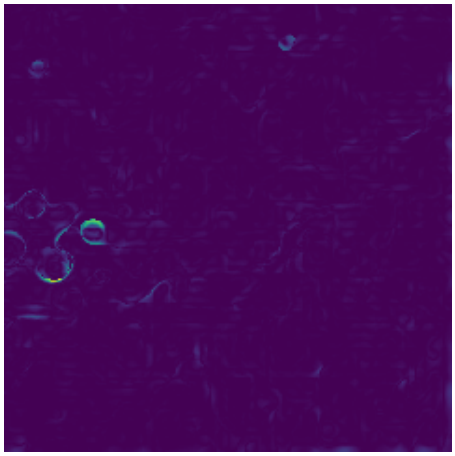


(d) crop superresuelto con ventana deslizante



MAE: 0.0879 RMSE: 0.2255

(e) error absoluto sin ventana deslizante



MAE: 0.0786 RMSE: 0.2082

(f) error absoluto con ventana deslizante

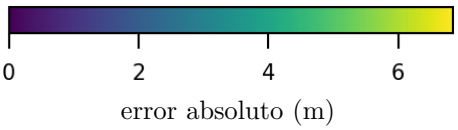


Figura 5.40: Comparación de un crop superresuelto con y sin el método de ventana deslizante.

5.6.3. Evaluación del modelo ESRGAN-B personalizado

Para evaluar los resultados del mejor modelo en el conjunto de evaluación se decidió comparar al modelo ESRGAN-B personalizado con la interpolación bilineal y la interpolación bicúbica. La Tabla 5.25 presenta los resultados de evaluar el modelo ESRGAN-B personalizado, la interpolación bilineal y la interpolación bicúbica en el conjunto de evaluación. Se reportan los promedios de evaluar las métricas MAE, RMSE, max error, LE90, precision y recall en cada crop del conjunto de evaluación. Además, se reportan los porcentajes de mejora de ESRGAN-B personalizado respecto a las interpolaciones para cada métrica.

modelo	MAE	RMSE	max error	LE90	precision	recall
ESRGAN-B	**0.0865**	**0.3375**	**9.3134**	**0.1340**	**0.9991**	0.9848
interpolación bilineal	0.2576	0.6482	11.3721	0.4980	0.9974	**0.9964**
interpolación bicúbica	0.2841	0.6890	12.2964	0.5327	0.9973	**0.9964**
mejora sobre bilineal (%)	66.4	48.0	18.1	73.1	0.17	-1.17
mejora sobre bicúbica (%)	69.6	51.0	24.2	74.9	0.18	-1.17

Tabla 5.25: Resultados de métricas en el conjunto de evaluación. Se incluye la mejora de ESRGAN-B personalizado respecto a las interpolaciones para cada métrica.

El modelo ESRGAN-B personalizado obtuvo mejores resultados en el conjunto de evaluación respecto a los métodos de interpolación evaluados. En la Tabla 5.25 se muestra cómo ESRGAN-B personalizado superó a las interpolaciones bilineal y bicúbica con mejoras del 66,4 % y 69,6 % en MAE, 47,9 % y 51,0 % en RMSE y 73,1 % y 74,8 % en LE90, respectivamente.

En la Tabla 5.26 se presenta el resultado del análisis de tiempo de inferencia en el conjunto de evaluación para el mejor modelo contra la interpolación bilineal y bicúbica. Se realizaron 10 ejecuciones independientes en una GPU NVIDIA A40.

modelo	tiempo mínimo (s)	tiempo máximo (s)	tiempo promedio (s)	desviación estándar
ESRGAN-B	34.44	38.03	36.54	1.04
interpolación bilineal	1.04	1.61	1.27	0.18
interpolación bicúbica	1.06	1.35	1.19	0.09

Tabla 5.26: Duración promedio en segundos de inferencia en el conjunto de evaluación para ESRGAN-B personalizado y las interpolaciones bilineal y bicúbica.

Los métodos de interpolación presentaron un tiempo de inferencia significativamente menor en comparación con el modelo ESRGAN-B personalizado. En el experimento realizado, el tiempo promedio de inferencia de ESRGAN-B personalizado fue 28.8 veces mayor que el de la interpolación bilineal y 30.7 veces mayor que el de la interpolación bicúbica. Esta diferencia se debe a la baja complejidad computacional de las operaciones involucradas en los métodos de interpolación en comparación con la complejidad del

generador de ESRGAN-B personalizado. El conjunto de evaluación en donde se realizó la inferencia incluyó 157 crops de tamaño 96×96. Cada celda de un crop representa una porción de terreno de 2.5 m $\times$ 2.5 m, lo que equivale a un área total de evaluación de $157 \times 96^2 \times (2.5 \text{ m})^2 = 9.04 \text{ km}^2$. La superficie total de Uruguay es de 176215 km^2, por lo que el modelo tardaría $(176215 \text{ km}^2/9.04 \text{ km}^2) * 36.54 \text{ s} = 8.2$ días en generar DEMs en alta resolución para todo el territorio uruguayo utilizando una GPU NVIDIA A40.

La Tabla 5.27 muestra una comparación entre la interpolación bilineal y el modelo ESRGAN-B personalizado para cuatro crops seleccionados del conjunto de datos de evaluación. Se utilizó la interpolación bilineal para la comparación ya que es la interpolación que presentó mejores resultados. Para cada crop se realiza una comparación visual y también se presentan los valores correspondientes de las métricas MAE y RMSE.

El modelo ESRGAN-B personalizado demostró ser consistentemente superior al método de interpolación bilineal de manera cualitativa y cuantitativa. Las diferencias visuales fueron mayores en crops con bordes finos y detalles pequeños. Sin embargo, la diferencia en términos de métricas fue mayor (a nivel proporcional) en crops con menos detalles finos, como fue el caso del crop mostrado en la tercera fila de la Tabla 5.27.

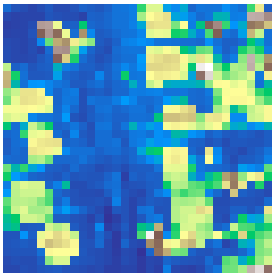
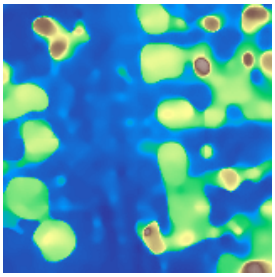
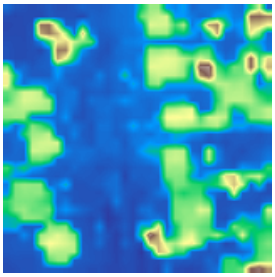
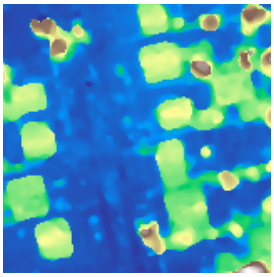
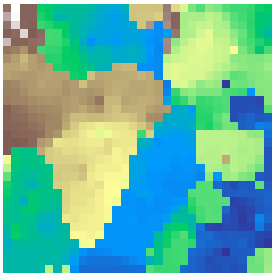
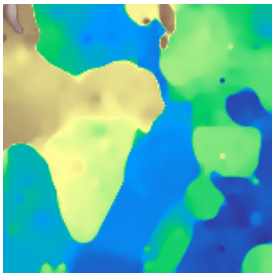
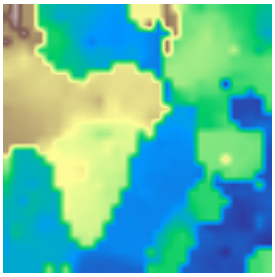
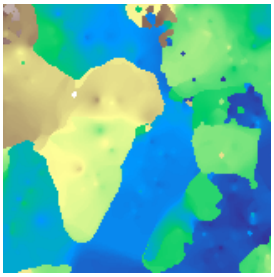
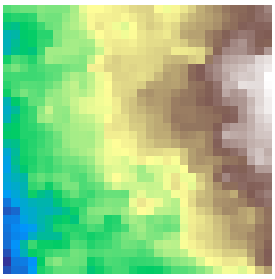
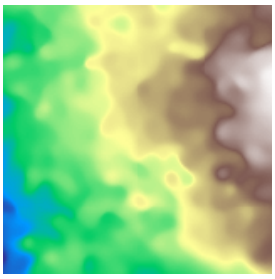
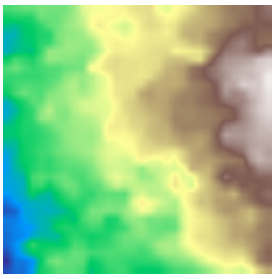
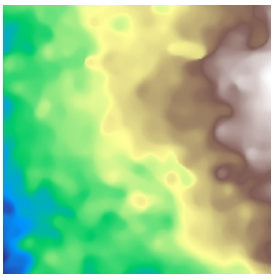
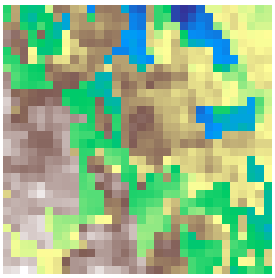
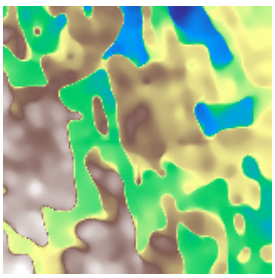
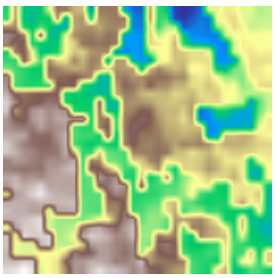
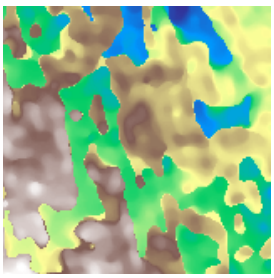
LR	ESRGAN-B	interpolación bilineal	HR
			
	MAE: 0.1887 RMSE: 0.4574	MAE: 0.4256 RMSE: 0.7585	
			
	MAE: 0.1315 RMSE: 0.4912	MAE: 0.2838 RMSE: 0.6443	
			
	MAE: 0.0476 RMSE: 0.0819	MAE: 0.1904 RMSE: 0.2659	
			
	MAE: 0.5044 RMSE: 1.2345	MAE: 1.4004 RMSE: 2.1638	

Tabla 5.27: Comparación visual y cuantitativa entre ESRGAN-B personalizado y el método de interpolación bilineal.

Capítulo 6

Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones y se incluyen posibles trabajos futuros a realizar sobre el tema abarcado en este proyecto de grado.

6.1. Conclusiones

En este proyecto de grado se abordó el problema de incrementar la resolución espacial de DEMs del territorio uruguayo de 2.5 a 0.5 metros mediante técnicas de superresolución basadas en GANs.

Se realizó un relevamiento de trabajos relacionados al problema de superresolución en DEMs con GANs y se definieron como arquitecturas seleccionadas a SRGAN (Ledig et al., 2017), ESRGAN (Wang et al., 2019), DSRGAN (Demiray et al., 2021) y una variante de la arquitectura SRGAN ya entrenada en otro conjunto de datos (Zhang y Yu, 2022). Cada arquitectura fue adaptada al problema de superresolución en DEMs. Los métodos de interpolación bicúbica y bilineal fueron tomados como base de comparación para el mejor modelo obtenido en este proyecto de grado.

Se realizó una búsqueda de hiperparámetros con el objetivo de hallar la mejor configuración para cada modelo de cada arquitectura evaluada. Con los mejores modelos obtenidos en cada arquitectura se realizó un entrenamiento en profundidad. Para medir la calidad de los resultados, se evaluó cada modelo con una selección de métricas consideradas relevantes para los DEMs. Las tres métricas que se consideraron de mayor relevancia fueron MAE, RMSE y LE90.

Los modelos basados en GAN superaron a los algoritmos de interpolación, tomando en cuenta las métricas principales, en todas las evaluaciones realizadas con los datos disponibles. El modelo desarrollado que alcanzó los mejores resultados fue ESRGAN-B personalizado. Este modelo, en comparación con la interpolación bilineal, redujo 66.4 % la métrica MAE, 48.0 % la métrica RMSE y 73.1 % la métrica LE90. En consecuencia, se concluyó que en los DEMs disponibles del territorio uruguayo, los modelos basados en GAN para superresolución son capaces de obtener resultados superiores en comparación con los algoritmos de interpolación.

Las modificaciones realizadas a los modelos seleccionados en este proyecto de grado fueron determinantes para alcanzar una mejora sustancial en la calidad de los resultados. Una de las modificaciones consistió en incorporar la métrica MAE en la función de pérdida de los modelos. Esta incorporación resultó en una mejora considerable en las métricas de evaluación. Los resultados también mejoraron sustancialmente con la incorporación del

entrenamiento B, que incluyó la desnormalización de los crops de DEMs en el cálculo de la función de pérdida durante el proceso de entrenamiento. Esta variante de entrenamiento demostró ser ampliamente superior al entrenamiento realizado inicialmente, denominado entrenamiento A, en todas las evaluaciones realizadas.

Se concluye que, si bien el uso de GANs para superresolución implica costos superiores en tiempos de ejecución y capacidad de cómputo en comparación con los métodos tradicionales de interpolación, estos costos no son excesivos y se justifican por la mejora significativa en los resultados obtenidos.

El trabajo realizado en este proyecto de grado se presentó en dos eventos en etapas preliminares de su desarrollo. El primer evento en el que se presentó el trabajo fue en las Jornadas del Instituto Panamericano De Geografía e Historia (IPGH) el 25 de septiembre de 2024. La presentación que se realizó en este evento se encuentra en `https://ipgh.org.uy/pages/eventos/2024-1.html`. En esta presentación se compararon los resultados obtenidos al realizar superresolución de crops de DEMs entre la SRGAN y las interpolaciones. El segundo evento en el que se presentó el trabajo fue en las Jornadas Uruguayas de Ciencias de la Computación (JUCC) 2024 "Jorge Vidart" el 13 de diciembre de 2024 (Nesmachnow y Moreno-Bernal, 2025), en donde se presentaron resultados más avanzados de las distintas arquitecturas con el entrenamiento A y su comparación con las interpolaciones.

6.2. Trabajo futuro

Con el objetivo de aprovechar al máximo el trabajo realizado en este proyecto de grado, se propone combinar las arquitecturas SRGAN, ESRGAN y DSRGAN. Para combinar las tres arquitecturas se propone entrenar una nueva red neuronal que reciba como entrada las salidas generadas por las tres arquitecturas y retorne un único DEM. La nueva red neuronal debería aprender a combinar las entradas para generar el DEM de salida. Este enfoque permitiría, en el mejor de los casos, aprovechar las fortalezas y mitigar las debilidades individuales de cada arquitectura.

Otra propuesta para trabajo futuro es incluir la información de la pendiente del terreno en la función de pérdida. Aunque la inclusión de la pendiente podría mejorar la capacidad de predicción de la elevación, este cambio requeriría que se genere un nuevo conjunto de datos con la información de la pendiente para el territorio uruguayo.

En los últimos años las arquitecturas basadas en transformers han demostrado un rendimiento sobresaliente en diversos dominios, por lo que experimentar con estas arquitecturas como método de superresolución de DEMs es prometedor. Entre los trabajos relacionados que fueron relevados en este proyecto de grado, la arquitectura DSRT (Li et al., 2023) se destacó por utilizar transformers para la superresolución de DEMs y según los autores reportó resultados superiores a los obtenidos con GANs. Se intentó contactar con los autores sin éxito, pero la arquitectura DSRT es una alternativa prometedora a investigar en trabajos futuros.

Bibliografía

Y. Bengio, P. Simard, y P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5, 1994.

P.A. Burrough, R.A. McDonnell, y C.D. Lloyd. *Principles of Geographical Information Systems*. OUP Oxford, 2015.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, y Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014.

Bekir Z. Demiray, Muhammed Sit, y Ibrahim Demir. D-srgan: Dem super-resolution with generative adversarial networks. *SN Computer Science*, 2, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, y Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Chao Dong, Chen Change Loy, Kaiming He, y Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *Computer Vision – ECCV 2014*. Springer International Publishing, 2014.

Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, y Hayit Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321, 2018.

GDAL/OGR contributors. *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation, 2025. URL `https://gdal.org`.

Sean Gillies et al. Rasterio: geospatial raster i/o for Python programmers, 2013. URL `https://github.com/rasterio/rasterio`.

Ian Goodfellow, Yoshua Bengio, y Aaron Courville. *Deep Learning*. MIT Press, 2016.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, y Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Hayit Greenspan. Super-resolution in medical imaging. *The Computer Journal*, 52, 2008.

Peter L. Guth, Adriaan Van Niekerk, Carlos H. Grohmann, Jan-Peter Muller, Laurence Hawker, Igor V. Florinsky, Dean Gesch, Hannes I. Reuter, Virginia Herrera-Cruz, Serge Riazanoff, Carlos López-Vázquez, Claudia C. Carabajal, Clément Albinet, y Peter Strobl. Digital elevation models: Terminology and definitions. *Remote Sensing*, 13, 2021.

Dianyuan Han. Comparison of commonly used image interpolation methods. *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*, 2013.

Sergey Ioffe y Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37. PMLR, 2015.

Donglai Jiao, Dajiang Wang, Haiyang Lv, y Yang Peng. Super-resolution reconstruction of a digital elevation model based on a deep residual network. *Open Geosciences*, 12, 2020.

Kinga Karwowska y Damian Wierzbicki. Mcwesrgan: Improving enhanced super-resolution generative adversarial network for satellite images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 16, 2023.

Diederik Kingma y Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

Jamy Lafenetre, Gabriele Facciolo, y Thomas Eboli. Implementing Handheld Burst Super-Resolution. *Image Processing On Line*, 13, 2023.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, y Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Zhuoxiao Li, Xiaohui Zhu, Shanliang Yao, Yong Yue, Ángel F. García-Fernández, Eng Gee Lim, y Andrew Levers. A large scale digital elevation model super-resolution transformer. *International Journal of Applied Earth Observation and Geoinformation*, 124, 2023.

Warren S. McCulloch y Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5, 1943.

Leonardo Moreira, Livia Poelking, Alan Graça, y Hideo Araki. Use of generative adversarial network algorithm in super-resolution images to increase the quality of digital elevation models based on alos palsar data. *Anuário do Instituto de Geociências*, 46, 2023.

Sergio Nesmachnow y Sebastián Iturriaga. Cluster-uy: Collaborative scientific high performance computing in uruguay. In Marco Torres y Jorge Klapp, editors, *Supercomputing*, volume 1151. Springer, Cham, 2019.

Sergio Nesmachnow y Pedro Moreno-Bernal. *Actas de las Jornadas Uruguayas de Ciencias de la Computación 2024 "Jorge Vidart"*. 02 2025.

Keiron O'Shea y Ryan Nash. An introduction to convolutional neural networks. *ArXiv*, abs/1511.08458, 2015.

Razvan Pascanu, Tomas Mikolov, y Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28. PMLR, 2013.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, y Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., 2019.

QGIS Development Team. *QGIS Geographic Information System*. QGIS Association, 2025. URL `https://www.qgis.org`.

Niles Ritter y Mike Ruth. *GeoTIFF Format Specification: GeoTIFF Revision 1.8.2*, 1995. URL `https://gis-lab.info/docs/geotiff-1.8.2.pdf`. Último acceso: 23 de diciembre de 2024.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6, 1958.

Juan Ruiz, Francisco Ariza, Juan Reinoso, Manuel Ureña, y Francisco Quesada. Deep learning methods applied to digital elevation models: state of the art. *Geocarto International*, 38, 2023.

Juan J. Ruiz-Lendínez, Francisco J. Ariza-López, Juan F. Reinoso-Gordo, Manuel A. Ureña-Cámara, y Francisco J. Quesada-Real and. Deep learning methods applied to digital elevation models: state of the art. *Geocarto International*, 38, 2023.

David E Rumelhart, Geoffrey E Hinton, y Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323, 1986.

Luis Salgueiro Romero, J. Marcello, y Verónica Vilaplana. Super-resolution of sentinel-2 imagery using generative adversarial networks. *Remote Sensing*, 12, 2020.

Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404, 2020.

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, y Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Karen Simonyan y Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio y Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Jingzhi Tu, Gang Mei, Zhengjing Ma, y Francesco Piccialli. Swcgan: Generative adversarial network combining swin transformer and cnn for remote sensing image super-resolution. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15, 2022.

Guido Van Rossum y Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, 2009.

Hongen Wang, LiYang Xiong, Guanghui Hu, Haoyu Cao, Sijin Li, Guoan Tang, y Lei Zhou. Dem super-resolution framework based on deep learning: decomposing terrain trends and residuals. *International Journal of Digital Earth*, 17, 2024.

Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, y Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Computer Vision – ECCV 2018 Workshops*. Springer International Publishing, 2019.

Bartlomiej Wronski, Ignacio Garcia-Dorado, Manfred Ernst, Damien Kelly, Michael Krainin, Chia-Kai Liang, Marc Levoy, y Peyman Milanfar. Handheld multi-frame super-resolution. *ACM Trans. Graph.*, 38, 2019.

Jin Yamanaka, Shigesumi Kuwashima, y Takio Kurita. Fast and accurate image super resolution by deep cnn with skip connection and network in network. In *Neural Information Processing*. Springer International Publishing, 2017.

Yifan Zhang y Wenhao Yu. Comparison of dem super-resolution methods based on interpolation and neural networks. *Sensors*, 22, 2022.

Apéndice A

Crops superresueltos

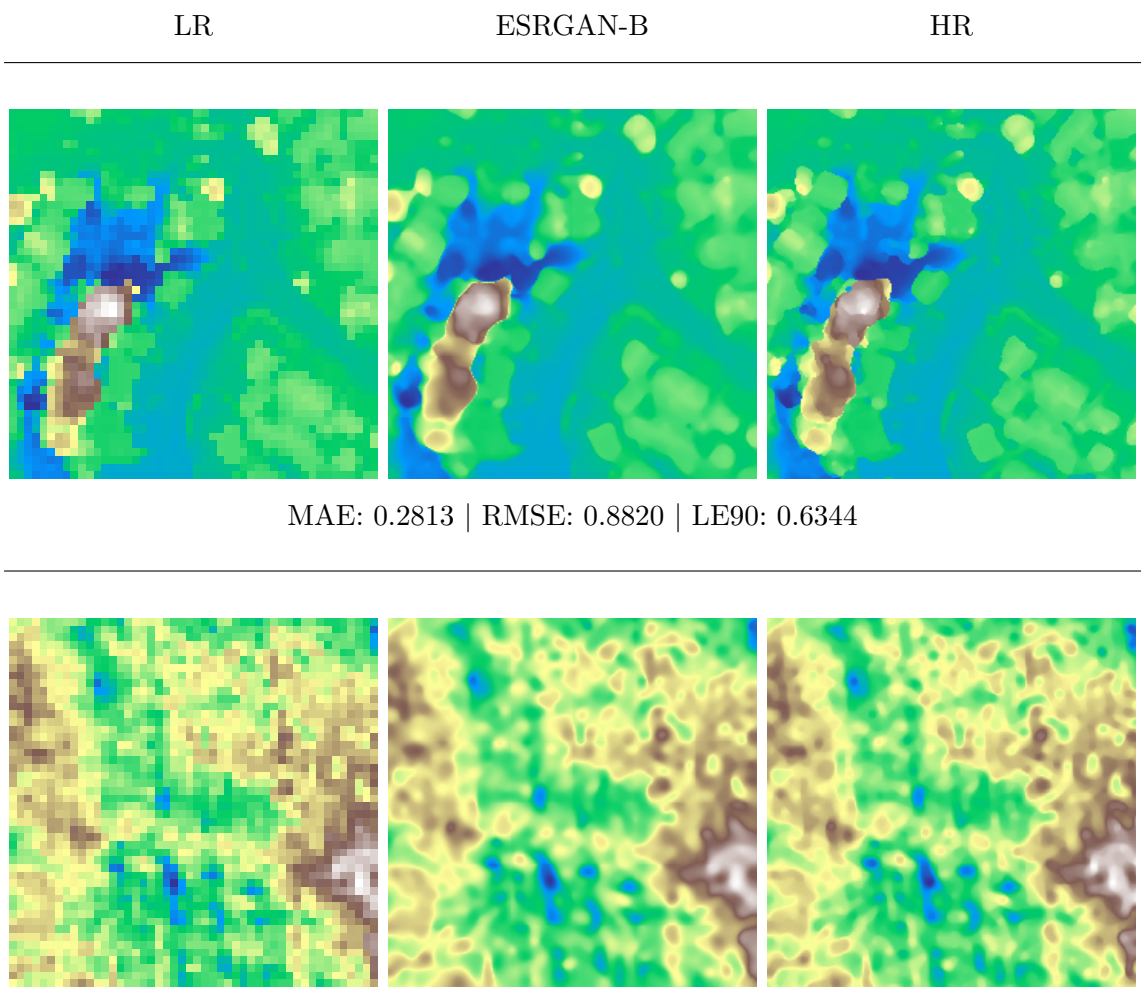
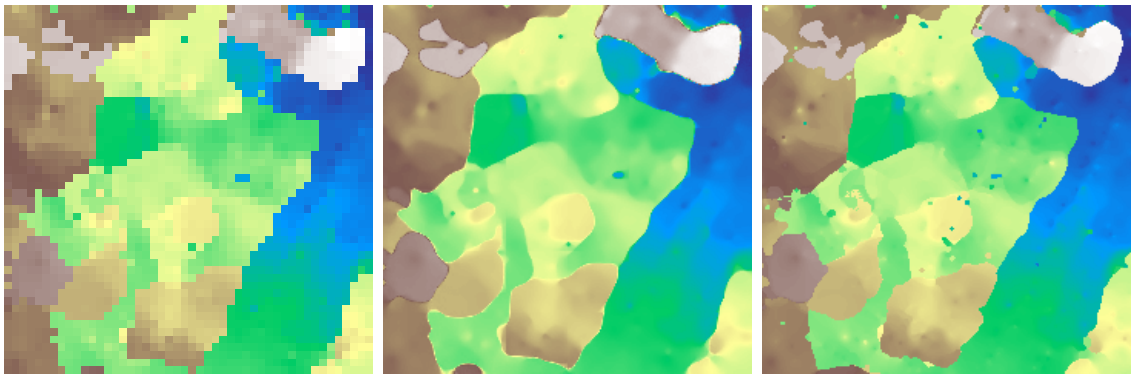
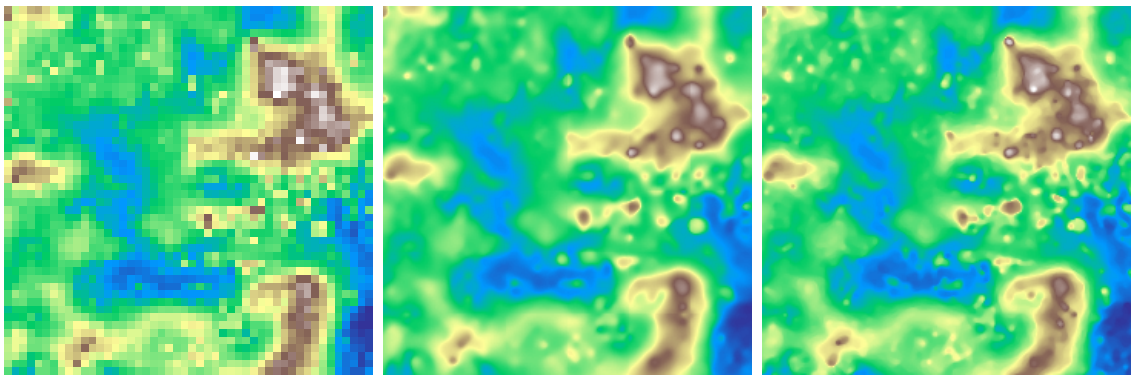


Figura A.1: Ejemplos de crops en baja resolución, superresueltos con ESRGAN-B y en alta resolución.

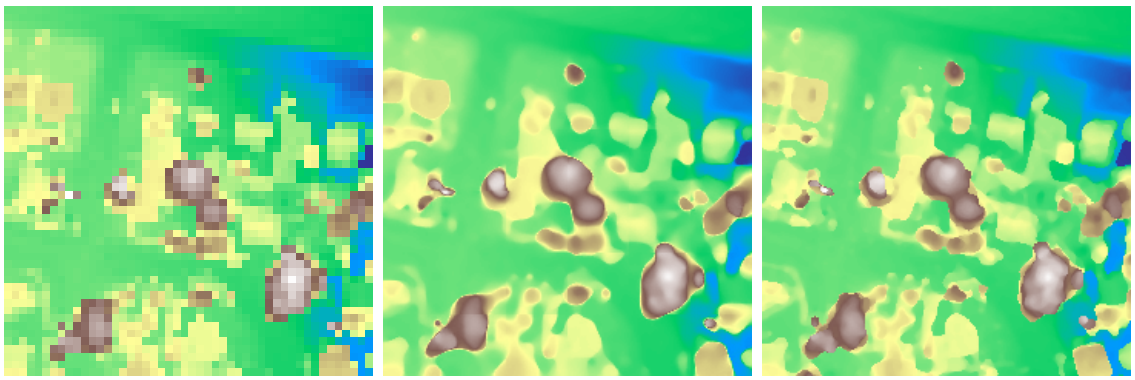
LR ESRGAN-B HR



MAE: 0.1637 | RMSE: 0.7222 | LE90: 0.1581



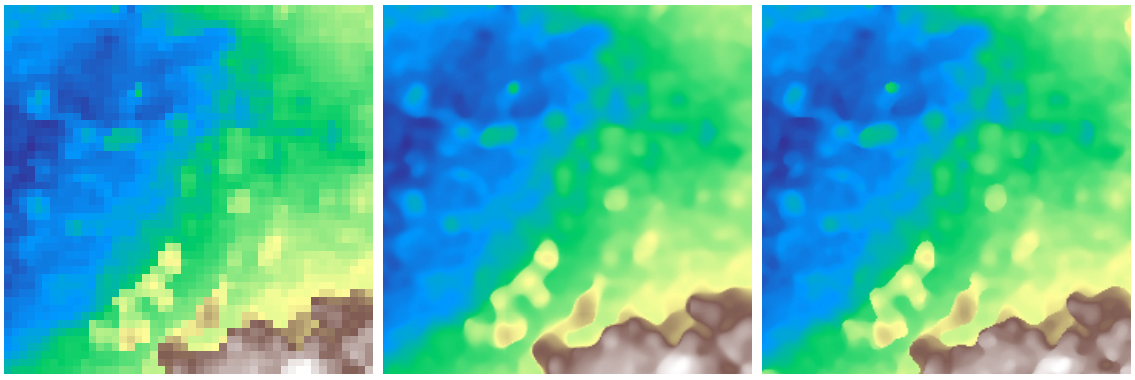
MAE: 0.0373 | RMSE: 0.0672 | LE90: 0.0885



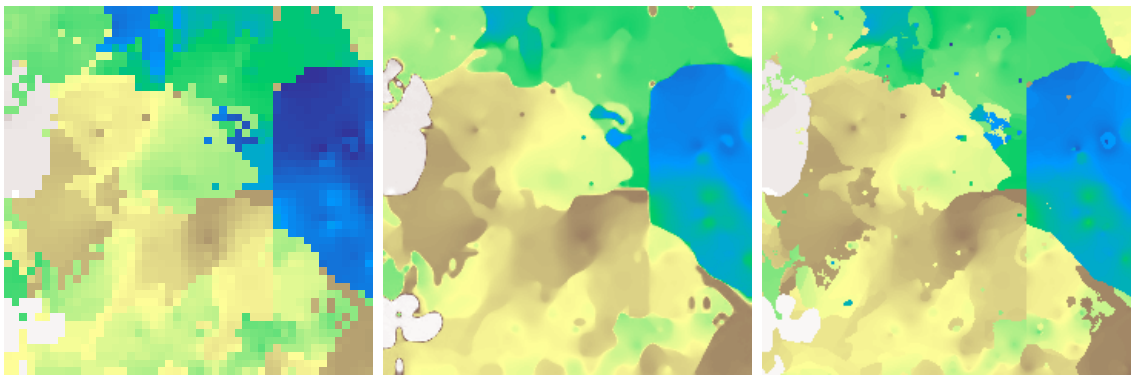
MAE: 0.2870 | RMSE: 0.8552 | LE90: 0.6223

Figura A.2: Ejemplos de crops en baja resolución, superresueltos con ESRGAN-B y en alta resolución.

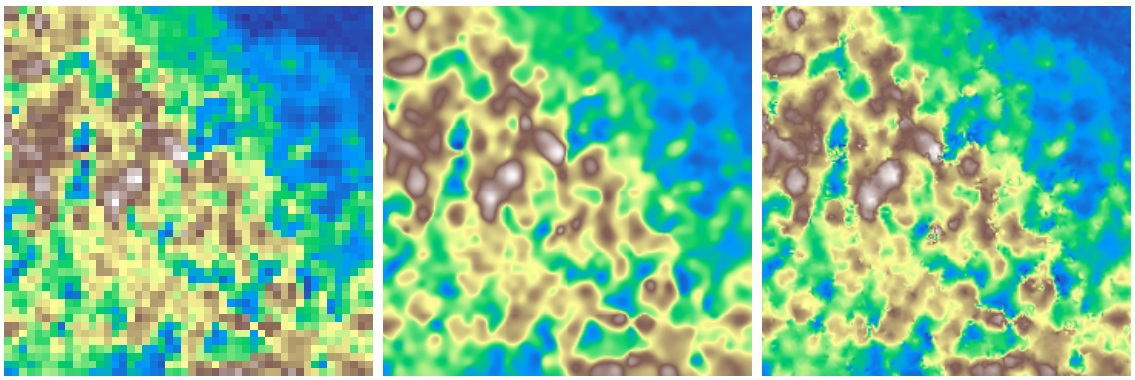
LR ESRGAN-B HR



MAE: 0.1424 | RMSE: 0.4419 | LE90: 0.2592



MAE: 0.1857 | RMSE: 0.7587 | LE90: 0.2069



MAE: 0.1709 | RMSE: 0.2902 | LE90: 0.4074

Figura A.3: Ejemplos de crops en baja resolución, superresueltos con ESRGAN-B y en alta resolución.

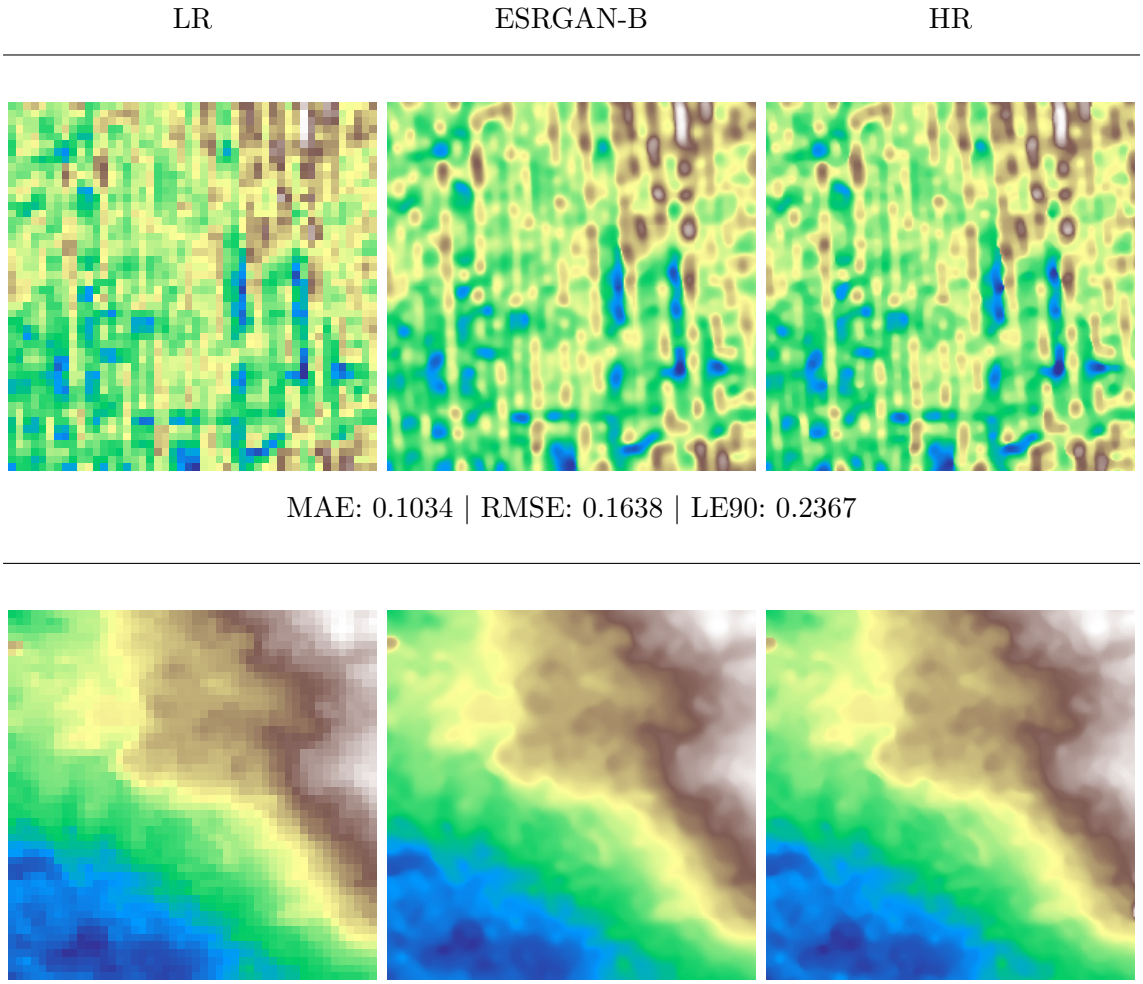


Figura A.4: Ejemplos de crops en baja resolución, superresueltos con ESRGAN-B y en alta resolución.