GSatelite: Gaussian Splatting para la Superresolución de imágenes satelitales

Adrián García Tapia

25 de mayo de 2025

Resumen– Este artículo trata la superresolución (SR) de imágenes de satélite de Sentinel-2 mediante Gaussian Splatting (GS), una técnica novedosa en SR que aún no ha sido aplicada a imágenes multiespectrales. El objetivo principal de este trabajo es implementar el modelo GSASR, adaptarlo para tratar imágenes multiespectrales y comparar el rendimiento de la versión implementada en RGB y en multiespectral. Se describe cada bloque que conforma la arquitectura del modelo, que consiste en un *Encoder*, un *Decoder* y *Rasterizador 2D*. El trabajo busca evaluar la mejora que supone el uso de información multiespectral frente a la RGB para la SR con GS en datos Sentinel-2.

Palabras clave– Superresolución, Escala Arbitraria, Gaussian Splatting, Sentinel-2.

Abstract– This paper is about super-resolution (SR) of Sentinel-2 satellite images using Gaussian Splatting (GS), a new technique in the field of SR that is yet to be applied to multispectral images. The main goal of this work is to implement the GSASR model, adapt it to multispectral images and compare the performance of both versions. Each block that makes the architecture is described, which consists in an *Encoder*, a *Decoder* and a *2D Rasterizer*. This work evaluates the improvement of using multispectral information against just RGB for SR with GS in Sentinel-2 data.

Keywords– Superresolution, Arbitrary-Scale, Gaussian Splatting, Sentinel-2.

1 Introducción

La superresolución (SR) de imágenes es un problema ampliamente tratado en imágenes RGB, utilizando técnicas de Deep Learning (DL) para superar a las técnicas tradicionales como el reescalado bicúbico. Recientemente, han empezado a usarse arquitecturas que no solo obtienen mejores resultados que técnicas anteriores, sino que también permiten reescalar las imágenes por cualquier factor de escalado, con ejemplos que varían desde un $\times 2$ hasta un $\times 30$, incluyendo la posibilidad de escalados decimales como podría ser $\times 2,5$. El uso de Gaussian Splatting (GS) en este área es lo más novedoso con apenas tres artículos publicados a día de redactar este trabajo.

Sin embargo, este nuevo algoritmo aún no se ha aplicado a imágenes multiespectrtales como es el caso de ciertas imágenes satelitales. Por ello, este trabajo profundiza en dicho ámbito, adaptando un modelo de SR basado en GS a imágenes satelitales de Sentinel-2.

La estructura del documento es la siguiente. En la Sección 2 se revisa el estado del arte de la superresolución y las distintas técnicas utilizadas, tanto en imágenes RGB como en multiespectrales, enfocadas más a las imágenes multiespectrales de satélite. Las Secciones 3, 4, 5 explican los objetivos de este trabajo así como la metodología seguida y la planificación realizada para conseguir dichos objetivos. La Sección 6 describe cada componente del modelo utilizado con el propósito de cada uno para lograr la resolución objetivo. La Sección 7 detalla el progreso realizado hasta el momento, los problemas que han ocurrido durante el desarrollo y las expectativas para las próximas semanas. La Sección 8 muestra los resultados y compara con el modelo original, así como el rendimiento del modelo RGB frente al multiespectral. Por último, en la Sección 9 se presentan las conclusiones donde se debaten los resultados obtenidos, se proponen posibles vías de investigación para un futuro y concluyen el trabajo.

2 Estado del arte

2.1. Superresolución

La SR se divide en: superresolución fija y superresolución arbitraria (ASR), y en superresolución con una sola imagen (SISR) y con múltiples imágenes (MISR). Para este trabajo, es de interés la ASR y la SISR.

La ASR en su mayoría se hace con una sola imagen, por lo que también entra en la categoría de SISR. Esta consiste en entrenar

• E-mail de contacto: adrian.garciat@autonoma.cat
• Mención realitzada: Computació
• Treball tutoritzat per: Felipe Lumbreras Ruiz (Ciències de la Computació)
• Curs 2024/25

un único modelo para que sea capaz de mejorar la resolución de una imagen en cualquier factor, como por ejemplo $\times 2$, $\times 3$, $\times 4$, etc. incluidos factores decimales. Meta-SR [8] es un método basado en operadores de interpolación, y fue el primero en hacer ASR dentro de los distintos métodos de deep learning que hay actualmente. Sin embargo, el estado del arte hasta hace poco han sido los métodos basados en representaciones neuronales implícitas (INR), inspirados en el éxito que han tenido estos en tareas de reconstrucción 3D con los NeRF [6]. LIIF [4] fue pionero en adaptar los métodos INR a ASR y lo siguieron otros métodos como ITSRN [18], SRNO [16] y CiaoSR [2], siendo este último el actual estado del arte de esta familia de modelos.

2.2. Superresolución con Gaussian Splatting

Una vez más inspirados por avances en el mundo del 3D, últimamente se ha empezado a explorar el uso de GS [9] en ASR gracias al éxito que han tenido frente a los NeRF.

El primero en llevar esta técnica a ASR es GaussianSR [7], que a pesar de ser superado por CiaoSR en la calidad de las imágenes finales, rivaliza con LIIF e ITSRN superándolos tanto en calidad como en velocidad de inferencia. GSASR [3] es el siguiente avance en esta área, superando a CiaoSR tanto en calidad de la imagen final como en tiempo de inferencia. Una de las diferencias clave respecto a GaussianSR es el uso de diversos mecanismos de atención y un rasterizador 2D implementado en cuda, que tiene su origen en la primera implementación de GS en 3D.

2.3. Superresolución Sentinel-2

En el área de las imágenes satelitales (RS) los métodos de SR suelen variar dependiendo del satélite de origen. En este caso nos centramos es Sentinel-2, una constelación formada por 2 satélites que proporcionan imágenes de 13 canales distintos, que van desde los 443 nm hasta los 2190 nm del espectro electromagnético [1]. Estos canales se encuentran a distintas resoluciones: 10 m/px, 20 m/px y 60 m/px.

Algunos de los métodos actuales se centran en unir todos los canales a una misma resolución de 10 m/px. HFN [17] es un método convolucional que junta información de todos los canales para llevarlos a 10 m/px de forma jerárquica. El transformer propuesto en [13] es un método reciente que también logra el mismo objetivo.

Por otro lado, hay métodos que intentan mejorar la resolución más allá de los 10 m/px, aunque se suelen centrar más en el RGB en lugar de unir la información de todos los canales y suelen dejar fuera los que están originalmente a 60 m/px. SENX4 [5] aplica una mejora de $\times 4$ a los canales de 10 m/px utilizando una red convolucional. RS-ESRGAN [12] se basa en el ESRGAN original [14] para aplicar una mejora de $\times 5$ a los canales de 10 m/px. Por desgracia estos últimos no suelen utilizar la información de todas las bandas para esta mejora.

Si saltamos al campo hiperespectral, DMGASR [15] es una técnica de difusión que permite utilizar una gran cantidad de canales C para llevar a cabo la superresolución, cosa que podría aplicarse a las imágenes de Sentinel-2.

3 Objetivos

El objetivo principal de este trabajo es implementar el modelo GSASR, adaptarlo a las imágenes multiespectrales de Sentinel-2 y ver qué mejora hay respecto a la versión RGB. Para comprobar esto, usaré solo los tres canales RGB de Sentinel-2 para la versión RGB y todos los canales para la versión multiespectral.

Aunque el anterior sea el principal objetivo del trabajo, en caso de haber tiempo suficiente también pretendo comparar el modelo respecto a las adaptaciones multiespectrales de los distintos métodos del estado del arte actual. Como propuestas escogeré CiaoSR, el Transformer propuesto en [13] y DMGASR. Además, al final de todo trataría de combinar HFN, que implementé durante mi estancia en las prácticas con éxito, para aplicar GSASR a partir de una resolución común de 10 m/px en todos los canales.

En la Tabla 2 hay un desglose de los objetivos en subobjetivos más pequeños.

4 Metodología

Como metodología se ha decidido optar por Kanban debido a que no requiere de trabajo en equipo y este es un trabajo individual, por lo que es suficiente para organizar las distintas tareas del proyecto. Trello es el tablero Kanban usado para llevar un registro del estado de las tareas. Las tareas del proyecto están sacadas de la Tabla 2 y se han organizado en el tiempo en un diagrama de Gantt, que se encuentra en el dosier de este proyecto.

Durante todo el proceso, el objetivo ha sido conseguir el objetivo principal mencionado en la Sección 3, sin tener en cuenta tareas de los siguientes objetivos extra para evitar distracciones. Estas solo se habrían realizado en caso de cumplir con el objetivo principal a tiempo.

5 Planificación

Como se comenta en la Sección 4, para planificar las tareas en el tiempo se ha utilizado un diagrama de Gantt con MS Project. Las tareas han sido extraídas directamente de la Tabla 2. El diagrama final del dosier ha sido dividido siguiendo un orden temporal para facilitar su legibilidad.

6 Arquitectura

La mayor parte del trabajo ha consistido en reproducir e implementar el modelo descrito en el artículo original, el cual presenta una arquitectura híbrida que combina componentes desarrollados en PyTorch con módulos implementados en CUDA.

Tal como se ilustra en la Figura 1, el modelo está compuesto por tres bloques fundamentales: un *Encoder*, encargado de extraer las características relevantes de la imagen; un *Decoder*, que genera las representaciones gausianas; y un *Rasterizador 2D*, que produce la imagen final a partir de dichas representaciones.

6.1. Encoder

El *Encoder* es el módulo responsable de la extracción de características a partir de la imagen de entrada. Está compuesto por un *backbone* de un modelo de SR ya existente, al cual se le añade una capa de convolución adicional que adapta su salida para que sea compatible con la entrada del *Decoder*. En esta implementación se ha utilizado el modelo EDSR [10] como *backbone*, siguiendo una de las configuraciones propuestas en el trabajo original. No obstante, es posible sustituirlo por otros modelos de características.

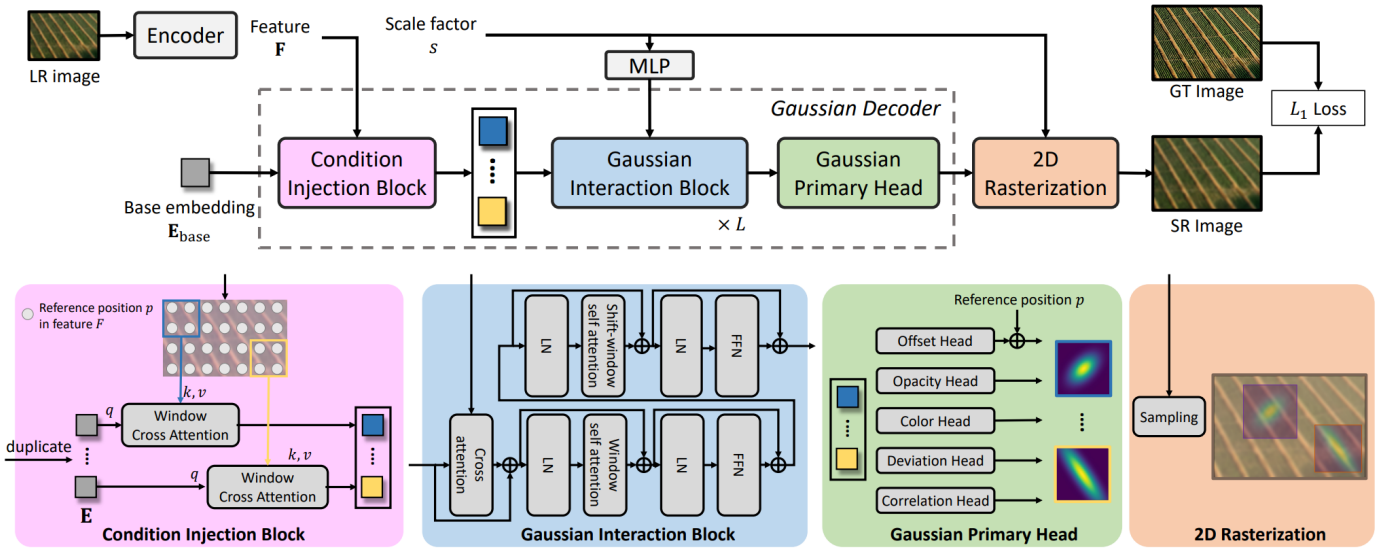

Fig. 1: Arquitectura GSASR.

6.2. Decoder

El *Decoder* se encarga de transformar las características extraídas por el *Encoder* en una representación gausiana que sirva como base para la generación de la imagen final. Este módulo se divide en tres bloques principales: el *Condition Injection Block*, el *Gaussian Interaction Block* y el *Gaussian Primary Head*.

6.2.1. Condition Injection Block

Este bloque recibe como entrada tanto las características generadas por el *Encoder* como un *embedding* entrenable que contiene información inicial sobre las gausianas. Su objetivo principal es fusionar ambos tipos de información mediante un mecanismo de atención cruzada con ventanas, inspirado en la arquitectura Swin Transformer [11], como se detalla en la Ecuación (1). El resultado de este bloque son los *embeddings* gaussianos iniciales.

$$\text{Attention}\,(Q,K,V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{d}} + B\right)V \qquad (1)$$

6.2.2. Gaussian Interaction Block

Este bloque consta de dos fases diferenciadas. En la primera, se combinan los *embeddings* gaussianos obtenidos en el bloque anterior con un factor de escalado, utilizando un mecanismo de atención cruzada descrito en la Ecuación (2). En este esquema, las *queries* provienen de los *embeddings*, mientras que las *keys* y *values* derivan del factor de escala.

$$\text{Attention}\,(Q,K,V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{d}}\right)V \qquad (2)$$

En la segunda fase, se introduce interacción entre los diferentes *embeddings* gaussianos mediante atención por ventanas desplazadas, siguiendo el enfoque propuesto en Swin Transformer [11]. Este proceso se repite en serie a lo largo de L bloques de tipo *Gaussian Interaction*, permitiendo una integración progresiva de la información. Los *embeddings* resultantes de esta etapa contienen ya todos los parámetros necesarios para definir las gausianas finales.

6.2.3. Gaussian Primary Head

El bloque final del *Decoder* tiene como propósito extraer los parámetros individuales de cada gausiana. Para ello, toma los *embeddings* producidos por el bloque anterior y los procesa mediante cinco redes MLP independientes, las cuales predicen la opacidad, el color, la desviación estándar, el desplazamiento (*offset*) y la correlación de cada gausiana. Finalmente, el *offset* se suma a la posición relativa estimada para determinar la localización exacta de cada gausiana en el espacio de salida.

6.3. Rasterizador 2D

Este componente es responsable de generar la imagen final a partir de las gausianas obtenidas, aunque no incluye parámetros entrenables. Cada gausiana calcula su contribución a los píxeles cercanos empleando la función de densidad gausiana representada en la Ecuación (3), y su efecto acumulativo sobre la imagen final se determina según la Ecuación (4).

$$f\,(x,y) = \left(2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}\right)^{-1} exp\Bigg[-\frac{1}{2\,(1-\rho^2)} \times \left(\frac{\Delta x^2}{\sigma_x^2} - \frac{2\rho\Delta x\Delta y}{\sigma_x\sigma_y} + \frac{\Delta y^2}{\sigma_y^2}\right)\Bigg] \qquad (3)$$

$$G\,(x,y) = \alpha c f\,(x,y) \qquad (4)$$

En dichas ecuaciones, α representa la opacidad, c el color, y ρ la correlación, que junto con la desviación estándar σ define la forma y orientación de la gausiana. Los términos Δx y Δy corresponden a la distancia entre el centro de la gausiana y el píxel de destino.

El procedimiento general seguido para llevar a cabo esta etapa se resume en el Algoritmo 1.

Para garantizar un rendimiento computacional adecuado, esta parte ha sido implementada en CUDA, a diferencia del resto del modelo que se ha desarrollado íntegramente en Python utilizando la biblioteca PyTorch.

7 Progreso

El modelo ya ha sido implementado y entrenado utilizando el procedimiento estándar de los modelos de ASR. Para evaluar el modelo se ha definido como métrica principal el Peak Signal-to-Noise Ratio (PSNR), mientras que como métricas secundarias se emplean el Structural Similarity Index Measure (SSIM), Learned Perceptual Image Patch Similarity (LPIPS) y Deep Image Structure and Texture Similarity (DISTS).

Algorithm 1 Algoritmo de rasterización 2D

Input: N (2D) Gaussians $\{G_1, G_2, ..., G_N\}$; LR image of size (H, W); scale factor s; raster ratio r.
Output: Renderized image I_{SR}.

Inicializar I_{SR} como una array de ceros de dimensión $(sH, sW, 3)$.
for each G_i in $\{G_1, G_2, ..., G_N\}$ **do**
    Inicializar $\alpha, \mu_x, \mu_y, \sigma_x, \sigma_y, \rho, c$ desde G_i.
    for each pixel (x, y) en I_{SR} **do**
        if $|x/s - \mu_x| < rH$ and $|y/s - \mu_y| < rW$ **then**
            Obtain $f(x/s, y/s)$ using Eq. (3);
            Obtain $G_i(x/s, y/s)$ using Eq. (4);
            $I_{SR}(x, y; s) += G_i(x/s, y/s)$
        end if
    end for
end for

El modelo en RGB se ha entrenado una vez y se está haciendo otra prueba de entrenamiento en estos momentos. En ambos entrenamientos se han utilizado los parámetros propuestos en el artículo original, excepto por el *batch size* por restricciones en la memoria de la GPU. El primer entrenamiento se hizo en una NVIDIA RTX 3090 con 24GB de VRAM con un *batch size* de 2, mientras que para el segundo se ha utilizado una L40s de 48GB de VRAM con un *batch size* de 8. El dataset en ambos es DIV2K, creando parches aleatorios de $48s \times 48s$, donde s es el factor de escala, escogido de forma aleatoria entre $\{1, 2, 3, 4\}$. Luego, se reescalan las imágenes a 48×48 y se pasan al modelo en el entrenamiento junto al factor de escala s.

El siguiente paso es adaptar el modelo a imágenes multiespectrales. En este caso, el dataset se extraerá de Sentinel-2 y se utilizará el mismo procedimiento que al entrenar con DIV2K. Para comparar el rendimiento de ambos modelos, se reentrenará el RGB con los canales RGB del satélite y el multiespectral con los 12 canales disponibles.

Como las bandas de Sentinel-2 están en distinas resoluciones se hará una primera prueba pasando todas las bandas a una resolución común de 10m/px utilizando HFN [17] y más adelante (si da tiempo) se adaptará el modelo para que pueda trabajar con todos los canales a la vez aunque tengan distinta resolución.

El trabajo llegará hasta ahí por falta de tiempo, en lugar de entrenar varios modelos para comparar los resultados.

8 Resultados

8.1. Resultados cuantitativos

Como se puede ver en la Tabla 1, nuestro modelo en RGB no llega a los mismos resultados que el modelo original, seguramente debido a diferencias en el código e incluso con la forma de entrenarlo. También cabe destacar que no hemos podido igualar todos los parámetros a los originales por problemas de memoria en la GPU. Por falta de tiempo no ha sido posible realizar una búsqueda de hiperparámetros, pero esto sirve como base para comparar con los resultados del modelo multiespectral.

8.2. Resultados cualitativos

En la Figura **??** se puede observar el resultado de nuestro modelo en RGB. Tal y como se veía en la Tabla 1, la calidad de la reconstrucción va disminuyendo conforme se aumenta el factor de escala. Además, se pueden apreciar las gausianas que forman la imagen en aquellas con un factor de escala mayor.

9 Conclusiones

Hasta ahora, hemos presentado nuestra implementación de GSASR, un modelo que adapta GS a ASR. Nuestra implementación sigue la misma arquitectura a pesar de no disponer del código original, sin embargo no se han obtenido resultados similares a los presentados en el artículo de origen. Como tareas a futuro todavía falta adaptar y entrenar el modelo a imágenes multiespectrales para poder comparar los resultados con los obtenidos en nuestra versión RGB.

El trabajo se ha visto limitado por falta de tiempo y recursos, y muchas de las tareas propuestas en la planificación original han quedado pendientes por hacer. Además, los resultados del modelo podrían haber sido mejores, pero a falta de tiempo no se ha podido hacer una búsqueda de hiperparámetros con la que mejorar las métricas finales obtenidas.

Para un futuro se podría optimizar el uso de memoria del modelo y la velocidad de procesamiento de las imágenes. Además, se podría adaptar para aceptar imágenes con canales en distintas resoluciones, de forma que el canal con mayor resolución siente una base de distribución de gausianas y los otros ayuden a refinar su posicionamiento y añadirles el color de los canales. También se podrían hacer pruebas para ver hasta qué punto ciertos componentes del modelo son necesarios e incluso permitir cambiar la resolución final de una imagen una vez se tienen las gausianas de la imagen.

Agradecimientos

Gracias al Centre de Visió per Computador (CVC) por proporcionarme equipo para poder entrenar los modelos, así como un espacio de trabajo en el que poder trabajar a gusto y con un entorno agradable.

Gracias también a Arnau Marcos por ayudarme a optimizar el código de CUDA del modelo, así como al resto de compañeros en el CVC por apoyarme moralmente y ayudarme a identificar posibles problemas en el código que se me pasaron por alto.

Por último agradecer a mi tutor, Felipe Lumbreras, por darme ideas y sugerencias a lo largo de estos meses, así como proponerme esta locura de proyecto sobre un tema tan novedoso con el que he sufrido, disfrutado y sobretodo, aprendido.

Referencias

[1] European Space Agency. *Sentinel-2 User Handbook*, 2015. Accessed: 25/02/2025.

[2] Jiezhang Cao, Qin Wang, Yongqin Xian, Yawei Li, Bingbing Ni, Zhiming Pi, Kai Zhang, Yulun Zhang, Radu Timofte, and Luc Van Gool. Ciaosr: Continuous implicit attention-in-attention network for arbitrary-scale image super-resolution, 2023.

[3] Du Chen, Liyi Chen, Zhengqiang Zhang, and Lei Zhang. Generalized and efficient 2d gaussian splatting for arbitrary-scale super-resolution, 2025.

[4] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function, 2021.

[5] Mikel Galar, Rubén Sesma, Christian Ayala, Lourdes Albizua, and Carlos Aranda. Super-resolution of sentinel-2 images using convolutional neural networks and real ground truth data. *Remote Sensing*, 12(18), 2020.

	PSNR ↑		SSIM ↑		LPIPS ↓		DISTS ↓	
	GSASR-O	GSASR-I	GSASR-O	GSASR-I	GSASR-O	GSASR-I	GSASR-O	GSASR-I
x2	36,65	28,09	0,9495	0,7363	0,0767	0,2735	0,0514	0,3416
x3	32,89	14,34	0,8960	0,6073	0,1782	0,3642	0,0963	0,5272
x4	30,89	13,53	0,8486	0,5957	0,2518	0,3778	0,1301	0,5796
x6	28,60		0,7784		0,3435		0,1835	
x8	27,22	11,54	0,7321	0,5034	0,4077	0,4416	0,2214	0,6465
x12	25,50	5,48	0,6777	0,4397	0,4975	0,5071	0,2787	0,6369
x16	24,38	4,93	0,6473	0,3573	0,5563	0,5468	0,3242	0,6996
x18	23,93	2,90	0,6367	0,3793	0,5790	0,5697	0,3419	0,6405
x24	22,90	4,94	0,6150	0,4823	0,6299	0,5600	0,3877	0,5392
x30	22,19	0,62	0,6025	0,3758	0,6648	0,5930	0,4232	0,6899
Mejor	∞		1		0		0	

TABLA 1: RESULTADOS CUALITATIVOS SOBRE EL SET DE VALIDACIÓN DE DIV2K. GSASR-O ES EL MODELO DEL ARTÍCULO ORIGINAL Y GSASR-I ES EL MODELO QUE HEMOS IMPLEMENTADO. PARA CALCULAR LAS MÉTRICAS DE NUESTRO MODELO SE HAN DIVIDIDO LAS IMÁGENES EN PARCHES DE $48s \times 48s$ Y SE HAN BAJADO LA RESOLUCIÓN CON LA INTERPOLACIÓN DE PYTORCH CON ANTIALIASING ACTIVADO. LAS MÉTRICAS DEL MODELO ORIGINAL SON LAS QUE HAY EN SU ARTÍCULO [3].

[6] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review, 2023.

[7] Jintong Hu, Bin Xia, Bin Chen, Wenming Yang, and Lei Zhang. Gaussiansr: High fidelity 2d gaussian splatting for arbitrary-scale image super-resolution, 2024.

[8] Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. Meta-sr: A magnification-arbitrary network for super-resolution, 2019.

[9] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023.

[10] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution, 2017.

[11] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[12] Luis Salgueiro Romero, Javier Marcello, and Verónica Vilaplana. Super-resolution of sentinel-2 imagery using generative adversarial networks. *Remote Sensing*, 12(15), 2020.

[13] Alireza Sharifi and Mohammad Mahdi Safari. Enhancing the spatial resolution of sentinel-2 images through super-resolution using transformer-based deep-learning models. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 18:4805–4820, 2025.

[14] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018.

[15] Zhaoyang Wang, Dongyang Li, Mingyang Zhang, Hao Luo, and Maoguo Gong. Enhancing hyperspectral images via diffusion model and group-autoencoder super-resolution network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(6):5794–5804, March 2024.

[16] Min Wei and Xuesong Zhang. Super-resolution neural operator, 2023.

[17] Jingan Wu, Liupeng Lin, Chi Zhang, Tongwen Li, Xiao Cheng, and Fang Nan. Generating Sentinel-2 all-band 10-m data by sharpening 20/60-m bands: A hierarchical fusion network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 196:16–31, 2023.

[18] Jingyu Yang, Sheng Shen, Huanjing Yue, and Kun Li. Implicit transformer network for screen content image continuous super-resolution, 2021.

APÉNDICE

O1(T)	O2(F)	O3(P)
Implementar RGB y Multiespectral.	Implementar RGB.	Crear repositorio de GitHub.
		Entender Condition Injection Block.
		Entender Gaussian Interaction Block.
		Entender Gaussian Primary Head.
		Entender 2D Rasterization Block.
		Aprender a combinar PyTorch+cuda.
		Estudiar las implementaciones en cuda de GS.
		Implementar Condition Injection Block.
		Implementar Gaussian Interaction Block.
		Implementar Gaussian Primary Head.
		Implementar 2D Rasterization.
		Preparar EDSR feature extractor.
		Conectar todos los bloques para el modelo final.
		Crear bucle de entrenamiento.
		Preparar Dataset RGB.
		Entrenar modelo.
		Comparar.
	Implementar MS.	Adaptar Condition Injection Block a MS.
		Adaptar Gaussian Interaction Block a MS.
		Adaptar Gaussian Primary Head a MS.
		Adaptar 2D Rasterization a MS.
		Adaptar EDSR feature extractor a MS.
		Preparar Dataset MS para EDSR.
		Preparar Dataset MS para GSASR.
		Crear bucle de entrenamiento para EDSR.
		Entrenar EDSR.
		Entrenar GSASR.
		Comparar.
Comparar con SotA Multiespectral	Comparar CiaoSR.	Preparar CiaoSR para MS.
		Crear bucle de entrenamiento.
		Entrenar CiaoSR.
		Comparar.
	Comparar Transformer SotA.	Preparar Transformer para MS.
		Crear bucle de entrenamiento.
		Entrenar Transformer.
		Comparar.
	Comparar Difusión SotA.	Preparar modelo para MS.
		Crear bucle de entrenamiento.
		Entrenar modelo de Difusión.
		Comparar.
Combinar HFN con GSASR.	HFN + GSASR.	Preparar HFN.
		Crear bucle de entrenamiento.
		Entrenar HFN.
		Comparar.

TABLA 2: TABLA DE OBJETIVOS.