GausSat: Gaussian Splatting for remote sensing Super-Resolution

Adrián García Tapia

June 30, 2025

Abstract– We explore Gaussian Splatting (GS) for arbitrary-scale super-resolution (ASSR) in multispectral (MS) Sentinel-2 satellite imagery, proposing **GausSat**, the first GS-based framework to handle MS inputs across all 12 available Sentinel-2 bands. By adapting the GSASR architecture, we enable scalable super-resolution (SR) while preserving spectral coherence. Experimental results show that MS-aware SR outperforms RGB-only baselines in structural fidelity (SSIM) and perceptual metrics (LPIPS/DISTS), despite lower PSNR. We address metric inconsistencies in satellite imagery and highlight the value of MS data in enhancing spatial detail. This work positions GS as a promising paradigm for remote sensing SR.

Keywords– Super-resolution, Arbitrary-Scale, Gaussian Splatting, Sentinel-2, Multispectral, Remote Sensing.

Resumen– Exploramos el uso de Gaussian Splatting (GS) para la superresolución a escala arbitraria (ASSR) en imágenes satelitales multiespectrales (MS) de Sentinel-2, proponiendo **GausSat**, la primera red neuronal basada en GS que procesa entradas MS en las 12 bandas de Sentinel-2 disponibles. Adaptando la arquitectura de GSASR, habilitamos una superresolución (SR) escalable preservando la coherencia espectral. Los resultados experimentales muestran que la SR con información MS supera las métricas obtenidas con solo información RGB en fidelidad estructural (SSIM) y métricas perceptuales (LPIPS/DISTS), a pesar de obtener un PSNR más bajo. Abordamos las inconsistencias métricas en las imágenes satelitales y destacamos el valor de los datos MS para mejorar el detalle espacial. Este trabajo posiciona GS como un paradigma prometedor para la SR en remote sensing (RS).

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

1 Introduction

Image super-resolution (SR) is a technique that enhances the original resolution of an image. From its inception, it has been used to improve the quality of images taken in our everyday lives with our phones or cameras. As time passed, more and more uses for this algorithm have been found. In entertainment, it is used to increase the performance of video games without losing visual quality. In can also be used to improve the quality of pictures or videos taken long ago with camera sensors that were not as good as today's. However, the most promising application is data generation. SR algorithms can be used to generate high-quality data from low-quality sensors to train other algorithms for purposes other than aesthetic, for example object detection and segmentation. The benefits of this data are twofold: first, it cheapens data generation by using cheaper sensors, and second, it allows us to use low-quality sensors that cannot be modified for any reason, such as satellites that cannot be modified while they are orbiting Earth.

In recent years, SR has evolved a lot and has received a lot of influence from the 3D reconstruction community. In particular, they have been influenced by Novel View Synthesis (NVS) techniques, which generate new images of an object or scene from a set of pictures taken from different points of view. These NVS techniques include NeRF [13], a revolutionary type of Implicit Neural Representation (INR) algorithm that can recreate 3D scenes with more detail than sparse points, and Gaussian Splatting (GS) [9], an efficient algorithm that produces faster and better results than NeRF

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

using Gaussians to encode the geometry of the scene. The latter technique was first extended to SR in 3D scenes to increase the quality of the reconstructed scenes, and recently it made its way through the SR of 2D images.

Both of these algorithms (INR and GS) applied to SR in 2D images allow for a very specific type of SR called Arbitrary-Scale Super-Resolution (ASSR). In the case of GS, it is due to its nature of representing the original discrete data in a continuous space. This allows us to sample the continuous space at will and effectively generate the upsampling factor we want, be it a $\times 2$ or a $\times 4.6$ scale factor.

At the time of writing this article, there were only three papers on the topic: Hu et al. 2024 [7], Chen et al. 2025 [3] and Peng et al. 2025 [14]. We focus on implementing the GSASR architecture proposed by Chen et al. 2025 [3] and extending its functionality for multispectral (MS) images.

After the next subsection, the document is structured as follows. Section 2 reviews the current State of the Art (SotA) of the different SR techniques applied to both RGB and MS images. Section 3 explains the development of the project as well as the tools we used. Section 4 focuses on the training configuration used to train every version of the neural network (NN). Section 5 presents the results obtained by each version of the NN in both quantitative and qualitative ways. Section 6 showcases the ablation study we conducted to validate our choice of Sentinel-2 preprocessing. And finally, Section 7 reviews the results presented and proposes ideas for future work.

1.1 Objectives

The main goal of this project is to explore a field that has not been explored yet: to use GS for SR on MS images. In particular, we focus on MS satellite images such as those from Sentinel-2 [1], a constellation of 2 different satellites from the Copernicus Project monitoring Earth through 13 different bands of the electromagnetic spectrum. To achieve this goal, we divided it into several tasks that iteratively helped us complete the project. Since this was such a novel work, we created a set of tasks that were needed to finish the project, and then some extra tasks in case there was more time to do them. The compulsory tasks included the implementation of each part of the GSASR architecture as described in the original paper, and the extra tasks were there to compare its performance against other SR models. In the end, we only had time to complete the necessary tasks due to some unexpected problems in the development, and we could not do the extra tasks. The development of these tasks is explained further in Section 3 and the tasks themselves can be found in Table 6 and on GitHub [1].

To achieve this goal, we did a Gantt diagram to plan each task and used Kanban as the organization methodology throughout the whole project. The planning can also be found on GitHub.

2 State of the Art

2.1 Super-Resolution

Super-resolution (SR) can be classified in different ways. For example, depending on how many images are used for the task, there is single image super-resolution (SISR) and multiple image super-resolution (MISR). The classification can also be based on the algorithm used for SR such as convolutional neural network (CNN), generative adversarial network (GAN), Transformers, diffusion models (DM), INR and, since the last year, GS. Depending on which part of the electromagnetic spectrum it works on, it can also be divided into SR for RGB images in the visual-light range, SR for MS images, and hyperspectral SR. And finally, if the SR algorithm is flexible enough to allow for different noninteger upscaling factors at inference, it falls under the category of ASSR, and if the upscaling factor is fixed, it pertains to the fixed scale super-resolution (FSSR).

SR is commonly done in 2D images that represent a single point of view. However, there are also SR methods in 3D scenarios to improve the rendered images of that 3D scene. An example of this is SRGS [5], an algorithm that takes advantage of the powerful representation of Gaussian primitives to provide high-quality rendering by optimizing on a high-resolution (HR) space with a sub-pixel constraint.

In 2D we have more variety in the methods used for SR. Using CNNs we are constrained to FSSR, some examples are ESPCN [17], a technique that focuses on the LR features and upscales the image by using a sub-pixel shuffling operator at the end, achieving a fast and memory efficient SR. EDSR [11] improves the quality of images by relying on deep residual networks. RCAN [24] introduces residual channel attention for SR.

Real-ESRGAN [19] uses ESRGAN [18] for restoration purposes and is trained on pure synthetic data. The main model adds Residual-in-Residual Dense Blocks (RRDB) without batch normalization to its pipeline to get high-quality results.

Moving to Transformers, there is SwinIR [10] based on the Swin Transformer [12] architecture, which relies on attention and uses shifted windows to make the self-attention calculation more efficient. Furthermore, with the rise of DM and their powerful generative capabilities, a lot of algorithms have been proposed for SR. Some examples are ResShift [23], which accelerates the diffusion process by shifting the residual between the high-resolution image and the low-resolution image. There is also OSEDiff [22], a one-step DM that uses the low-quality image as input to the denoising process to accelerate the diffusion process while achieving good results by eliminating the uncertainty introduced by random noise.

Finally, Meta-SR [8] is a method based on interpolation operators and the first to do ASSR instead of FSSR. However, it has been surpassed by INR based methods inspired by the success of NeRF on the 3D NVS task, which are also capable of ASSR. LIIF [4] was the first to adapt INR methods to the ASSR task, and others followed, such as CiaoSR [2], which is the current SotA of INR methods for ASSR.

2.2 Gaussian Splatting Super-Resolution

In the last year, GS [9] has made its way into the SR field first in 3D with the SRGS [5] and later in 2D, with the possibility for ASSR which appears directly due to its construction.

GaussianSR [7] was the first to take this technique to the 2D SR task. Despite being surpassed by CiaoSR, it out-

[1] https://github.com/adriangt2001/TFG-Satellite-GSSR

performs LIIF and ITSRN both in image quality and inference speed. GSASR [3] was the next advance in this area, proposing a different architecture based on attention and including a 2D rasterizer implemented in C++/CUDA for faster inference. It performs better than CiaoSR in final image quality and inference time. Finally, the most recent of all is ContinuousSR [14], uses some Gaussian priors to train and outperforms all previous ASSR methods in image quality and has a surprisingly fast inference time, beating other methods by orders of magnitude.

2.3 Sentinel-2 Super-Resolution

In remote sensing (RS), the SR methods vary depending on the target satellite. In this specific case, we focus on the Sentinel-2 satellite, a constellation of 2 satellites from the Copernicus Program of the European Space Agency (ESA) that captures images from 13 different bands of the electromagentic spectrum ranging from a deep blue color at 443 nm to the range of short-wave infrared (SWIR) at 2190 nm [1]. Each of these bands is captured in a different resolution because of the sensors used for each band. We include more information about each band in Table 1.

Some SR methods focus on taking all bands to 10 m/px as the target resolution. HFN [21] is a convolutional method that uses information from all channels to bring all to 10 m/px in a hierarchical way. The Transformer proposed by Sharifi et al. 2025 [16] is a recent method that also achieves the same goal.

Band	Wavelength	Resolution	Purpose
B01	443 nm	60 m/px	Aerosol detection
B02	490 nm	10 m/px	Blue
B03	560 nm	10 m/px	Green
B04	665 nm	10 m/px	Red
B05	705 nm	20 m/px	Vegetation classification
B06	740 nm	20 m/px	Vegetation classification
B07	783 nm	20 m/px	Vegetation classification
B08	842 nm	10 m/px	Near Infrared
B8A	865 nm	20 m/px	Vegetation classification
B09	945 nm	60 m/px	Water vapour
B10	1375 nm	60 m/px	Cirrus
B11	1610 nm	20 m/px	Snow/ice/cloud discrimination
B12	2190 nm	20 m/px	Snow/ice/cloud discrimination

Table 1: INFORMATION ON THE WAVELENGTH, RESOLUTION AND PURPOSE OF EACH BAND FROM SENTINEL-2.

On the other hand, there are algorithms to increase the resolution beyond those 10 m/px, though they usually focus more on the RGB bands rather than fusing information from all bands, leaving aside those channels that are not originally in 10 m/px. SENX4 [6] applies a $\times 4$ resolution increase to all 10 m/px bands using a convolutional neural network. RS-ESRGAN [15] applies a $\times 5$ upscaling to the 10 m/px bands, based on the previously mentioned ESRGAN [18].

In the hyperspectral domain we have DMGASR [20], a DM that allows the use of a large number of channels from an image to do the SR by grouping them into different encoders. Despite not being focused on Sentinel-2, it can be applied to it to increase the resolution of its bands.

3 DEVELOPMENT

The development consisted on three main tasks. The first was to implement from scratch the GSASR algorithm and extend its use case as explained in the next subsection. Secondly, we prepared and built the necessary datasets to train the model in its different variants. And finally, we compared the results with those of the original paper.

3.1 GSASR

As stated in Section 1, our main goal is to perform SR using GS, focusing on the GSASR [3] architecture. At the beginning of this project, the original paper had no code available, just the description of its architecture, and thus we implemented all of it from scratch. We went further on and extended it from just using RGB images to also support MS images just by changing the number of channels in its parameters. Furthermore, although the main dataset used for SR in RGB images is the DIV2K dataset, we were unable to find a standard dataset for this task using MS images from any source, so we also built that dataset which we will explain more in Section 4.

The architecture is divided into three parts: the first is the Encoder which extracts a feature map from the image, then there is the Decoder that generates all Gaussian parameters, and finally the 2D Rasterizer that samples the Gaussians to generate the HR image. The Encoder relies on a backbone to extract the feature map, in our case we chose to use EDSR [11] as it is the backbone that gives better results in the original work. The reason for not testing more backbones is that we did not have enough time to test multiple backbones because, as we will see later in Section 4, each training experiment takes one week, and implementing the algorithm until it worked took a couple of months. The Condition Injection Block in the Decoder takes some trainable embeddings for the Gaussian and fuses its information with windows from the feature map extracted from the Encoder. Those new embeddings are then passed through L stacked Gaussian Interaction Blocks, which refine the embeddings with themselves by applying the same shifted windows attention mechanism used in Swin [12] while also adding information of the target scale factor s. The final embeddings go through five different Multilayer Perceptrons (MLP) to output the final parameters of each Gaussian. In the end, there is the 2D Rasterizer that samples the final HR image. In Figure 1 is a visual representation of the model architecture, and a more detailed explanation of each of its parts can be found in the original paper [3]. To differentiate the original model from ours, we decided to name our implementation GausSat, since its main purpose is to work on MS satellite images.

Equations (1) and (2) are the same Gaussian equations as the original. μ is the mean of the Gaussian and places it at a certain point in space. σ is the standard deviation of the Gaussian and ρ is the correlation between the x and

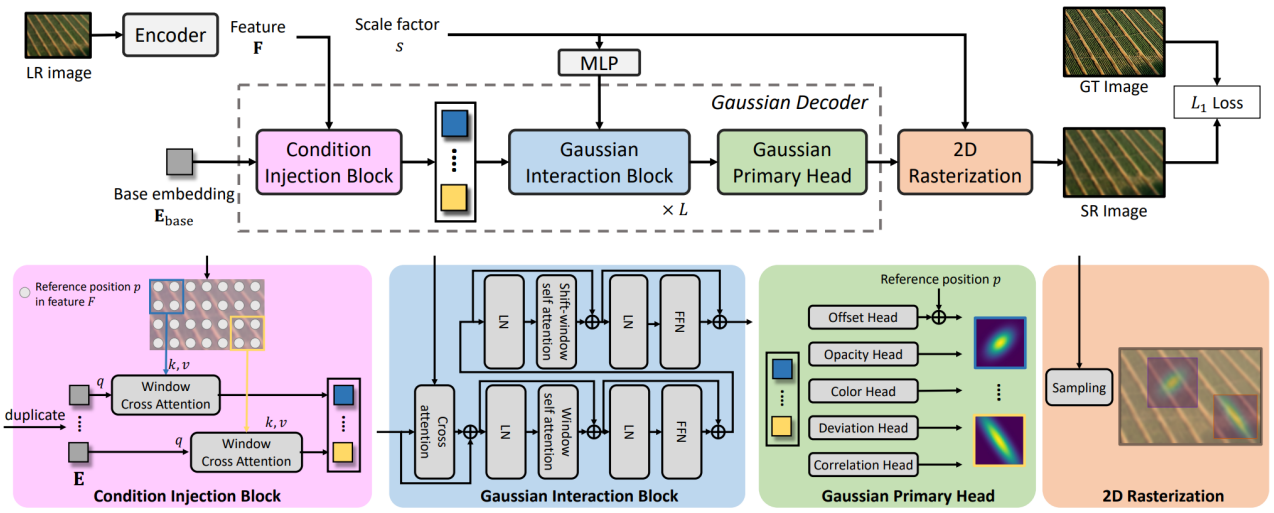


Fig. 1: GSASR architecture, extracted from the original paper. It follows an Encoder-Decoder architecture with a Rasterizer at the end. The Encoder is a backbone and extracts features from the input image. The Decoder generates Gaussian Embeddings and fuses the information from these embeddings and the input image features with various attention mechanisms, to generate the final parameters of each gaussian that represents the image. (*Source: GSASR, 2025*)

y axes, these two parameters give shape and orientation to the Gaussian. Δx and Δy are the distances in both axes from the evaluated point to the Gaussian center defined by μ. Finally, α is the opacity of the Gaussian and c is a vector describing the color of the Gaussian in each channel of the image.

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

$$G(x,y) = \alpha c f(x,y)\ . \quad (2)$$

To implement both the Encoder and Decoder, we used Python 3.10.14 and PyTorch 2.5.1. Just as in the original work, we chose to implement the 2D Rasterizer in C++/CUDA as a PyTorch extension, so we compiled it with CUDA version 12.4 and made sure to use PyTorch compiled with the same CUDA version. The Algorithm 1 is the pseudocode for the 2D rasterizer. Since it is built in C++/CUDA, we launch one thread for each Gaussian, and every thread adds its contribution to every pixel of the image that is close enough to the Gaussian center. Our modification of that algorithm is to add the number of channels C as a controllable parameter instead of assuming it is always going to be 3.

To build the Sentinel-2 dataset, we used Rasterio, a Python library to read and write different raster formats.

Algorithm 1 2D Rasterizer algorithm

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

```
Initialize I_SR as a zero array of dimensions (sH, sW, C).
for each G_i in {G_1, G_2, ..., G_N} do
    Initialize α, μ_x, μ_y, σ_x, σ_y, ρ, c from G_i.
    for each pixel (x, y) in I_SR do
        if |x/s − μ_x| < rH and |y/s − μ_y| < rW then
            Obtain f(x/s, y/s) using Eq. (1);
            Obtain G_i(x/s, y/s) using Eq. (2);
            I_SR(x, y; s) += G_i(x/s, y/s)
        end if
    end for
end for
```

3.2 HFN

GSASR, as many other algorithms, expects the input to be a single tensor with C channels, which implies that all channels need to have the same size. However, as mentioned earlier, each Sentinel-2 band captures images at a different resolution. To solve this issue, instead of downsampling all channels to the lowest resolution, we used HFN [21] to bring all bands to 10 m/px. There is also no code available for this architecture, so we also built it from the description in their paper. It is a CNN that first upsamples the input image by using bicubic interpolation, concatenates the image with the HR bands, and passes the it through Residual Dense Blocks to get all the features of the image and Residual Channel Attention Blocks to contribute with information from every channel. This architecture is based on pansharpening, a SR technique that upscales a LR image and then uses another HR image of the same scene to enhance the details on the predicted image. In particular, this algorithm uses a hierarchical approach for pansharpening. Since Sentinel-2 has 3 different resolutions at 10 m/px, 20 m/px, and 60 m/px, the NN first upscales the 20 m/px bands to 10 m/px using the information from the original 10 m/px bands, and then it uses the same strategy for the 60 m/px bands, but this time it uses both original and predicted 10 m/px bands to guide the process. Figure 2 has a visual representation of its pipeline, and a more detailed explanation of its architecture can be found in the original source [21]. We also show some examples of its predicted images in Figure 6 from the Appendix.

In this architecture, we used the same Python and Pytorch versions as before and Rasterio to read Sentinel-2 data.

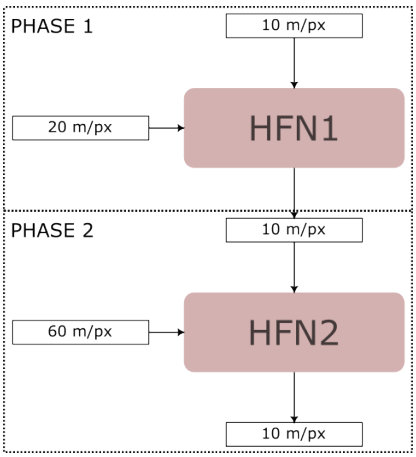


Fig. 2: HFN pipeline. In Phase 1 it uses HR 10 m/px bands to guide the SR process of 20 m/px bands to a resolution of 10 m/px, and concatenates them to form a single MS image at 10 m/px resolution. In Phase 2, it does the same for 60 m/px bands to bring them to a resolution of 10 m/px but using the new MS image at 10 m/px resolution. In the end, it outputs all Sentinel-2 bands at 10 m/px resolution.

4 Experiments

We performed three different experiments with this model: one with the DIV2K dataset, another only with RGB Sentinel-2 bands, and the third with all available Sentinel-2 bands. To evaluate these experiments, we used the Peak Signal-to-Noise Ratio (PSNR) as the main metric, and also the Structural Similarity Index Measure (SSIM), Learned Perceptual Image Patch Similarity (LPIPS) and Deep Image Structure and Texture Similarity (DISTS) as secondary metrics.

4.1 DIV2K Training

The first experiment was conducted to test the performance of our implementation of GSASR compared to the results reported in the original paper. For that we used the DIV2K dataset, a widely used dataset for the SR task since it was introduced in 2017, made of very HR natural images of many objects and scenes. This dataset is already divided into train, validation, and test, but since the test split is not available to the public, we only used the train and validation sets, named DIV2K800 and DIV2K100, respectively. To prepare the data for training, we follow the same procedure described in the original GSASR paper [3], which is the standard procedure for ASSR trainings with this dataset as seen also in [2] and [4].

We prepocess the data on demand, randomly cropping patches of $48s \times 48s$ pixels, where s is the scale factor randomly chosen between $\{1, 2, 3, 4\}$ at each iteration. The generated patch is then downsampled using bicubic interpolation to a size of 48×48 and then fed to the neural network along with s, the chosen scale factor. The standard procedure for downsampling is to use MATLAB's bicubic resizing, but we did not find that implementation for Python in time, so we tried PyTorch's interpolator operator with bicubic downsampling and antialiasing on a test image and found that the resulting images were quite similar.

The NN is configured in the same way as in the original work, with an embedding size of 180, a window size fixed to 12, 4 attention heads in every attention step, 6 stacked Gaussian Interaction Blocks, a rasterization ratio of 0.1 and a density of Gaussians m of 16. For the inner MLPs, we used their MLP decrease ratio of 4. As mentioned in Section 3, we keep EDSR as the backbone of the encoder, with an inner feature size of 64 and 16 residual blocks.

We tried to keep the same training hyperparameters, but our implementation is less memory efficient, and we do not have the same resources as they, so we tried to change them accordingly. Thus, we set a batch size of 9 as it is the maximum we can fit inside the GPU, the learning rate is lowered to $7.5e-5$, we keep the warm-up iterations to 2000 followed by a scheduler with a decay factor of 0.5 at iterations 500 000, 800 000, 900 000 and 950 000. In the last decay step, early stopping kicks in, using the PSNR to decide when to stop training. We also used mixed precision for a more efficient training and gradient clipping to ensure gradient stability in the whole process.

The training was carried out on an ADA 6000 GPU with 48 GB of VRAM for seven days.

4.2 Sentinel-2 Training

Our second experiment is the core of this work. Here, we trained two versions of this model on a Sentinel-2 dataset created by us manually downloading images from the Copernicus Browser[2]. We downloaded a total of 5 different images from different dates and locations, and since the HR bands of the images have a size of 10K, it is more than enough to build this dataset. These are the images that we used:

Around Pyhäjärvi, Finland 29/06/2022: S2B_MSIL2A_20220629T100029_N0510_R122 _T35VML_20240630T200504.

Catalonia, Spain 27/11/2024: S2B_MSIL2A_20241127T104309_N0511_R008 _T31TDG_20241127T131852.

Aragon, Spain 30/11/2024: S2B_MSIL2A_20241130T105329_N0511_R051 _T30TXK_20241130T132639.

Spanish-France border 30/11/2024: S2B_MSIL2A_20241130T105329_N0511_R051 _T31TCH_20241130T132639.

Northern Australia 16/05/2025: S2C_MSIL2A_20250516T014601_N0511_R074 _T52KBG_20250516T045613.

Three of these images were used to create the training dataset, another was used for the validation set, and the remaining served as the test set. We used Rasterio and HFN to preprocess the images and bring all bands to the same common resolution of 10 m/px. After that, the dataset is preprocessed in the same way as the DIV2K dataset.

The first version of the model uses only the RGB bands to upscale them to the target resolution, and the second model uses all available Sentinel-2 bands. Although Sentinel-2 has 13 bands, we only use 12 of them, since band B10 is not available in the images we downloaded.

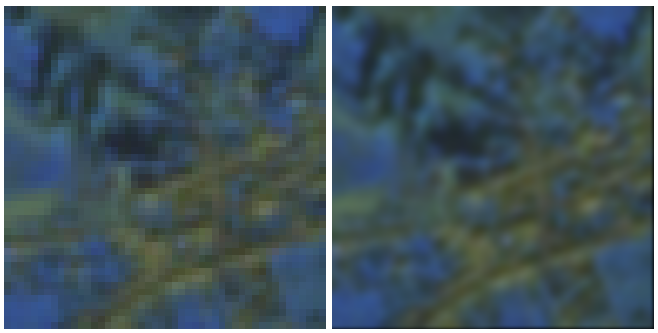
Both NNs are configured almost the same way as in the DIV2K experiment, and the training hyperparameters are the same. The only difference is that the MS version is configured to accept and output images of 12 channels instead of 3. The RGB-only experiment was performed on an L40S GPU with 48 GB of VRAM, and the MS experiment was trained on an ADA 6000 GPU with 48 GB of VRAM. Both trainings lasted seven days.

[2] https://browser.dataspace.copernicus.eu/

(a) LR (b) GSASR (c) GausSat

Fig. 3: Visual comparison of GSASR and GausSat on a $\times 8.6$ scale factor.

(a) LR (b) GausSat-RGB (c) GausSat-MS



(d) LR (e) GausSat-MS

Fig. 4: Visual comparison of GausSat-RGB and GausSat-MS on a $\times 12$ scale factor. Top row is the RGB image, available in both models; their brightness and contrast have been modified with the same factor in the three images for a better visualization. Bottom row is the image in 3 random MS channels. In both cases, the left image is the LR image and the other images are the output of each model.

4.3 Original training

A couple of weeks before submitting the final report of this project, the original paper code [3] was released, so we decided to train it as well to compare the results. However, there are a few things to clarify. First, we use the original training code as is; this means that bicubic dowsampling is done using a Python implementation of MATLAB's bicubic resize, which is different from our approach. Secondly, we keep the same hyperparameters as theirs except for the batch size. As we only have one GPU available per training, we use the maximum batch size we can fit inside, which is 14 for this case. This experiment was trained on an ADA 6000 GPU with 48 GB of VRAM for seven days.

5 Results

In this section we report the results obtained from the various experiments we conducted.

5.1 DIV2K

Table 2 shows the results of both GausSat and GSASR applying scaling factors from up to $\times 16$. As we can see, our model does not reach the expected results from the original GSASR. Furthermore, we tried to change the size of the input image for testing from 48×48 to the uncropped version of the image, but it failed to give good results and the output becomes random colored lines. That is why our test is done with cropped patches of 48×48, unlike the original test code for their implementation, which uses the entire images

donwscaled in the DIV2K100 dataset as input. We believe that this difference in the test code is what causes the mismatch between LPIPS and the other metrics as the scaling factor increases.

In Figure 3, we can see a qualitative comparison of the output of both models. As we can see, the results are quite similar despite the metrics that say that our implementation performs worse than the original. However, by taking a closer look at the GausSat image, we can see small dot artifacts around the edges of the image. The main purpose of this analysis is to establish a relation between our implementation and theirs, and to see which metric values and details we expect to see in the next subsection.

5.2 Sentinel-2

Here we show the quantitative results for our GausSat trained on the Sentinel-2 dataset. Table 3 shows the quantitative results for all available channels in each model for scaling factors of up to $\times 12$. We only use PSNR and SSIM because LPIPS and DISTS have no pre-trained weights on images with 12 channels. As we can see, the RGB version of the NN yields better results than those from the MS version. However, there is a strong discrepancy between the DIV2K experiment and this one, as both variations trained on this dataset perform better than the DIV2K version, the original GSASR and every other SotA model.

In Figure 4 we can see the qualitative results of this test with an example in RGB and another taken from 3 other different bands. In both cases, we can see that the same blurriness appears, which also matches the results shown in the previous subsection. We think that this mismatch between the reported metrics and the visual quality of the image is caused by the usual color range of the Sentinel-2 dataset. That is why we conducted another experiment, this time focusing only on the RGB bands, as the GausSat-RGB model has the best results of all. We can see the results of this test in Table 4.

The results reported in Table 4 are very different from those in Table 3. In this case, the MS version matches or even surpasses the performance of the RGB version in every metric except for PSNR, although they are more similar now. We concluded two things from this: the first one, backed by all metrics but LPIPS, is that using MS information for SR can be beneficial for the NN as it has more information to work with, and the second one is that we reinforced our suspicion that the issue is related to specific bands of the image, in this case the RGB bands. In the next subsection, we dive deeper into the metric inconsistency we found.

5.3 Metrics Analysis

Sentinel-2 images are encoded in an int16 format raster, which means they have values that can range from 0 to 65 536. However, its values rarely go beyond 20 000, and for RGB bands, they are usually in the range of 0 to 5000. As with any other computer vision task, we first need to normalize these images and work with them using floats in the usual range of 0 to 1 or -1 to 1. The official Sentinel-2 documentation states that the images have to be divided by 10 000 in order to normalize them, so in the end we are left with an image closer to the 0 to 1 range of float encoded images.

Our theory is that this low range is causing the metrics to be so high in the experiments using this dataset, especially in RGB bands, which after normalization are usually in the range of 0 to 0.5. That is why we use this subsection to analyze the PSNR and SSIM metrics to explain what is happening.

First, we focus on how the PSNR is calculated. According to Equation (4), this metric is based on two variables: MAX_I, which is the maximum value that the signal can take, and MSE. MAX_I is 1 for float encoded images, so that is fixed as a constant every time we calculate PSNR. On the other hand, MSE described in Equation (3) uses the difference between the predicted value and the real value of each pixel, averages them all, and outputs a number. So, if every pixel has a value between 0 and 0.5 most of the time and the NN learns to predict most values in that range, the possible error the algorithm will make is lower than the error it makes when working with the full range of 0 to 1. This explains why in both Sentinel-2 models, results are higher than in the DIV2K variants, and also why it has even higher values when using only the RGB bands of the image. Here are the equations for computing PSNR and MSE, where n is the number of pixels in the image, y_i is the real value of pixel i and p_i is the predicted value of pixel i:

$$\text{MSE} = \frac{1}{n}\sum (y_i - p_i)^2 \ , \tag{3}$$

$$\text{PSNR} = 10 \times \log_{10}\left(\frac{\text{MAX}_I^2}{\text{MSE}}\right) \ . \tag{4}$$

Then we analyze SSIM, which is a similar case. In Equation (5), to obtain an SSIM of 1 we need the same mean μ and sample variance σ in both images x and y. Thus, our images with a common value between 0 and 0.5 will have a mean value in that interval and a variance that will rarely exceed a value greater than 0.125. When our model starts predicting values in that range, the mean and variance from the predicted images will be much closer to those of the groundtruth than in the DIV2K case with its more dispersed color range. That is why we also get better results with SSIM as well. Here is the equation for computing the SSIM, where μ_x and μ_y are the mean values of pixels in images x and y, respectively, σ_x and σ_y are the sample variances of pixels from both images, σ_{xy} is the covariance of images x and y, and c_1 and c_2 are two variables used to stabilize the division with a weak denominator:

$$\text{SSIM} = \frac{(2\mu_x\mu_y + c_1)\,(2\sigma_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \ . \tag{5}$$

6 Ablation

We conducted an ablation study to determine whether the use of HFN to prepare every Sentinel-2 band is justified instead of just using bicubic to obtain the same results. Since preparing the dataset using two different methods implies that the groundtruth generated will be different for each case, the use of metrics will not add trustworthy information. In this case, bicubic upsampling performs worse than the HFN generated data, especially in bands such as 60

	PSNR ↑		SSIM ↑		LPIPS ↓		DISTS ↓	
	GSASR	GausSat	GSASR	GausSat	GSASR	GausSat	GSASR	GausSat
×2	**36.25**	33.10	**0.9466**	0.9148	**0.0828**	0.1260	**0.0545**	0.0628
×4	**30.55**	28.31	**0.8408**	0.7787	0.2653	**0.2128**	**0.1364**	0.2547
×6	**28.30**	25.25	**0.7691**	0.7033	0.3646	**0.2584**	**0.1930**	0.3548
×8	**26.95**	23.77	**0.7227**	0.6467	0.4293	**0.2900**	**0.2328**	0.4305
×12	**25.26**	21.96	**0.6698**	0.5860	0.5256	**0.3427**	**0.2924**	0.5360
×16	**24.17**	21.11	**0.6411**	0.5707	0.5877	**0.3785**	**0.3398**	0.5997
Best	∞		1		0		0	

Table 2: PERFORMANCE COMPARISON OF GSASR AND GAUSSAT AT DIFFERENT SCALING FACTORS. THE BEST RESULT IN EACH METRIC IS HIGHLIGHTED IN **BOLD**.

	PSNR ↑		SSIM ↑	
	GausSat RGB	GausSat MS	GausSat RGB	GausSat MS
×2	**44.06**	40.33	**0.9721**	0.9697
×3	**41.21**	36.20	**0.9503**	0.9277
×4	**39.72**	34.10	**0.9305**	0.8850
×6	**37.34**	31.38	**0.9027**	0.8152
×8	**36.54**	30.14	**0.8880**	0.7749
×12	**35.11**	28.63	**0.8788**	0.7443
Best	∞		1	

Table 3: PERFORMANCE COMPARISON BETWEEN GAUSSAT-RGB AND GAUSSAT-MS AT DIFFERENT SCALING FACTORS. THESE RESULTS ARE CALCULATED OVER THE RGB BANDS IN THE GAUSSAT-RGB MODEL AND OVER THE ALL BANDS FOR THE GAUSSAT-MS. THE BEST RESULT IN EACH METRIC IS HIGHLIGHTED IN **BOLD**.

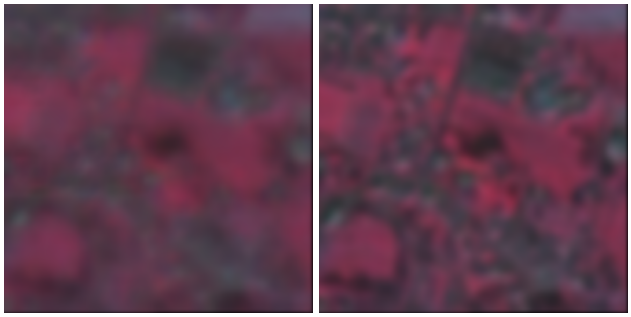
m/px where the scaling factor is so big. Thus, metrics will penalize the more detailed prediction as in the bicubic experiment the groundtruth is more blurry and a blurry output is encouraged, while the HFN high detail is more demanding, and it is harder to get a high value in metrics. To address this issue, we only use the original 10 m/px bands as groundtruth to compare both outcomes.

We show qualitative results in Table 5 and we can see that using HFN to prepare the data gives better results than using bicubic, even if that difference decreases as the scale factor increases. In Figure 5, we show some qualitative proof of the same result on bands different from the original 10 m/px.

We did not do the same experiment with the GausSat version trained only on RGB bands because those bands were originally in 10 m/px resolution and are not affected by HFN or bicubic, so the result from those two experiments would just be the same.

7 CONCLUSION

In conclusion, in this report we presented our implementation of GSASR adapted to satellite MS images, named GausSat to differentiate them. This NN uses GS to perform SR with any continuous upscaling factor. As seen in Section 5, our implementation does not match the results of the original NN using the DIV2K dataset. We consider two possible reasons.



(a) Bicubic (b) HFN

Fig. 5: GausSat-MS comparison at ×12 upscaling. (a) bicubic interpolation to prepare the data by increasing the resolution of the Sentinel-2 bands from 20 m/px and 60 m/px to a common 10 m/px; (b) HFN preprocess to achieve the same results. The image shown uses three random channels from Sentinel-2 different from the RGB channels.

The first reason is that the hyperparameters chosen to train our NN may not have been optimal for this task. Unfortunately, we ran into a lot of problems implementing every part of the architecture and it was not until a few weeks from the end that it started working as expected. At first, it behaved strangely, blurrying all the images in excess until nothing was recognizable. Once that was fixed, we found that not all Gaussians were being rasterized, so gradients were not flowing correctly, causing predicted images to have a decent level of detail in some parts of the image, while having undistinguishable blurs in other parts. Thus, with the remaining time, our limited resources and knowing that there were three versions of the model to train (until the original code was uploaded, which made four versions), we set the hyperparameters and did one experiment for each version.

The second reason is that there are some details in our code that differ from the original implementation. We did not have time to check how the original code works, but there can be things that the original paper omitted or that we interpreted differently. A more thorough inspection could shed light on the matter, but as the original code is available and works as expected, it would be better to conduct some ablation studies and directly adapt their code to the MS SR task rather than reimplementing it from scratch again.

	PSNR ↑		SSIM ↑		LPIPS ↓		DISTS ↓	
	GausSat-RGB	GausSat-MS	GausSat-RGB	GausSat-MS	GausSat-RGB	GausSat-MS	GausSat-RGB	GausSat-MS
×2	**44.06**	43.60	0.9721	**0.9726**	0.1081	**0.0977**	0.0479	**0.0406**
×3	**41.21**	40.86	**0.9503**	**0.9503**	0.1798	**0.1611**	0.1398	**0.1281**
×4	**39.72**	39.53	0.9305	**0.9314**	0.2449	**0.2155**	0.2291	**0.2016**
×6	**37.34**	37.28	0.9027	**0.9040**	0.3288	**0.2969**	0.3890	**0.3276**
×8	**36.54**	36.51	0.8880	**0.8887**	0.3864	**0.3571**	0.5220	**0.4573**
×12	**35.11**	34.93	**0.8788**	**0.8788**	0.4642	**0.4527**	0.6708	**0.6426**
Best	∞		1		0		0	

Table 4: PERFORMANCE COMPARISON BETWEEN BOTH VERSIONS OF GAUSSAT AT DIFFERENT SCALING FACTORS. GAUSSAT-RGB HAS TRAINED ONLY ON THE RGB BANDS AND GAUSSAT-MS HAS TRAINED WITH ALL AVAILABLE BANDS. THESE RESULTS ARE CALCULATED OVER THE RGB BANDS ONLY, AS LPIPS AND DISTS ARE NOT TRAINED ON MULTIESPECTRAL DATA. THE BEST RESULT IN EACH METRIC IS HIGHLIGHTED IN **BOLD**.

	PSNR ↑		SSIM ↑	
	HFN	Bicubic	HFN	Bicubic
×2	**39.99**	35.39	**0.9657**	0.9229
×3	**36.44**	35.29	**0.9296**	0.9152
×4	**34.66**	34.14	**0.8965**	0.8882
×6	**32.26**	32.06	**0.8471**	0.8437
×8	**31.28**	31.14	**0.8198**	0.8181
×12	**29.91**	29.82	**0.8000**	0.7993
Best	∞		1	

Table 5: RESULTS OF THE ABLATION STUDY TO TEST THE CONTRIBUTION OF HFN VS. BICUBIC TO PREPARE THE SENTINEL-2 IMAGES AND USING GAUSSAT-MS. THE BEST RESULTS ARE HIGHLIGHTED IN **BOLD**.

Leaving aside the results of the GausSat trained on DIV2K, the MS variation using Sentinel-2 gave some promising results. Despite the fact that the PSNR and SSIM have unrealistic values, our GausSat trained on all bands from Sentinel-2 slightly surpassed the version trained only with the RGB bands in every metric except for the PSNR. We conclude that it is a sign that the same NN architecture with the same parameters benefits from the extra information given by all MS bands.

Furthermore, we also noticed a strange behavior in the usual SR metrics when working with Sentinel-2 images and perhaps with satellite images in general. We showed how the unusual color range of these types of images affects the PSNR and SSIM metrics, making them unsuitable to compare with natural images from datasets like DIV2K. PNSR could be corrected by adjusting the MAX variable per image channel, but possible outliers caused, for example, by clouds could cause problems. The same applies to SSIM, which could adjust the c_1 and c_2 variables to adapt to this type of images.

We also draw some conclusions from the ablation study performed in this report. As we suspected, the use of HFN instead of bicubic to preprocess the dataset is justified, and the quality of the final results can be observed in plain sight. In addition, Table 5 showed that the metrics calculated only in the four original 10 m/px bands were better in the HFN prepared dataset than in the bicubic, strengthening our theory that the quality and quantity of MS data in an image can help further refinement when doing SR in the image, improving the final result we would obtain from processing each channel independently.

In general, we have tested and verified that the representative potential of GS is not limited to 3D scenes, but it also works for 2D images. This raises questions about whether that potential can extend to other tasks, such as segmentation, to mention one.

7.1 Future Work

We want to expand on some ideas presented in the conclusions as options for future lines of research.

First of all, in this project we had to rely on another NN to make sure that all channels from an image were the same, but considering that GS takes images to a continuous scale-agnostic space, we think it is possible to add information iteratively. This could be done by using information of the highest quality available to distribute Gaussians across that 2D space and using other channels to add color information and further define the Gaussians distribution. The resolution of these images would not matter as long as the image represents the same scene. This could also be expanded to using different close images from the same scene to refine the correct placement and shape of Gaussians, similar to the process used in 3D GS but applied to 2D images slightly shifted. This can introduce GS into MISR and not only SISR. This idea can go even beyond that and propose Gaussians to represent a continuous reality instead of a discrete one, as we have been doing so far. Information could come from different sensors, not just images, and be represented in the same continuous Gaussian space.

Secondly, we are concerned about the lack of a standard dataset for SR of satellite images that uses MS information, apart from RGB channels. Current datasets only have RGB channels from satellites for SR, and those with MS channels are focused on using pansharpening to reach the same reference resolution instead of pushing beyond that and improving the overall quality of the image. Thus, for tasks involving that extra image definition with MS information, a dataset is required, as without a standarized dataset comparison between methods and algorithms are meaningless unless perfect results are reached, which is not expected. Sentinel-2 is a good candidate as a source for this task, but it is important to take more satellites into consideration to choose correctly or even to use multiple satellites as sources.

And finally, we would like to do ablation studies on the architecture of these new algorithms, like, for example, testing whether it is important to use the scale factor as information to extract Gaussian parameters or not. The recent use of Gaussian priors in the latest GS SR work of Peng et al. 2025 [14] is an attractive idea and can benefit satellite images, so it would be a good project to extract Gaussian priors from multiple satellite sources, as they may differ from those in natural images.

8 Acknowledgements

Thanks to the Computer Vision Center (CVC) for providing me the necessary resources to do this project, as well as for offering a comfortable working space.

I want to thank Arnau Marcos for his help in optimizing the C++/CUDA code of the model and to the rest of my CVC colleagues such as Jordi Ventosa, Paula Font and Gerard Asbert for discussing ideas with me and motivating me through the process of developing this project. They have also helped me identify some issues with the code that I did not catch just by myself.

Finally, thanks to Felipe Lumbreras for being my tutor in this project through all these months, helping me with doubts I had at specific moments, and being through the whole process since the start. I also want to thank him for suggesting this novel idea as the topic of my final thesis, which I have enjoyed and suffered, but that has helped me learn a lot and grow my motivation in Computer Vision.

References

[1] European Space Agency. *Sentinel-2 User Handbook*, 2015.

[2] Jiezhang Cao et al. 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] Xiang Feng et al. Srgs: Super-resolution 3d gaussian splatting, 2024.

[6] 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.

[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 et al. 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] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer, 2021.

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

[12] Ze Liu et al. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[13] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[14] Long Peng et al. Pixel to gaussian: Ultra-fast continuous super-resolution with 2d gaussian modeling, 2025.

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

[16] 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 EO and RS*, 18:4805–4820, 2025.

[17] Wenzhe Shi et al. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, 2016.

[18] Xintao Wang et al. Esrgan: Enhanced super-resolution generative adversarial networks, 2018.

[19] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data, 2021.

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

[21] Jingan Wu et al. Generating Sentinel-2 all-band 10-m data by sharpening 20/60-m bands: A hierarchical fusion network. *ISPRS Journal of Photogrammetry and RS*, 196:16–31, 2023.

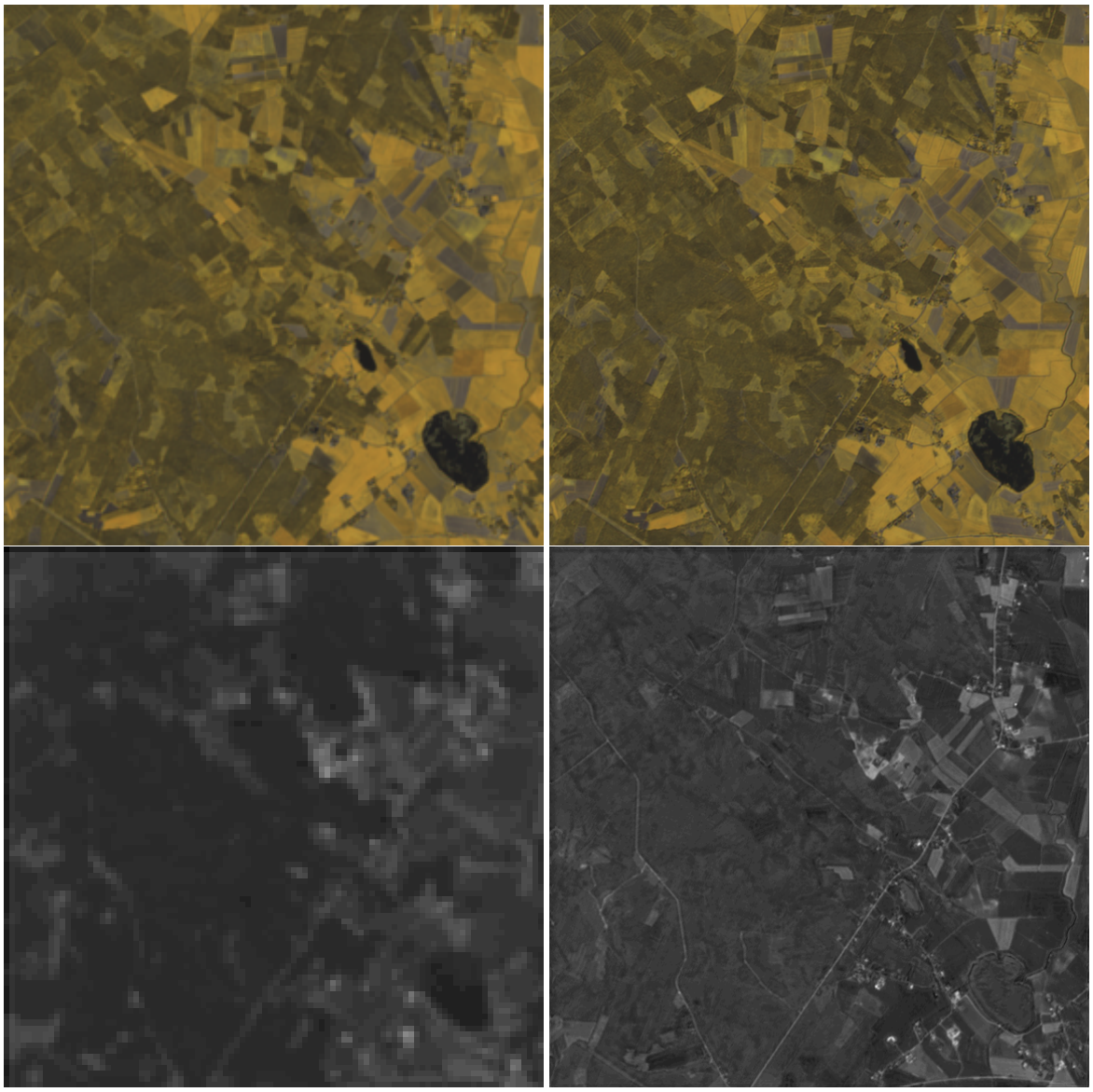
[22] Rongyuan Wu, Lingchen Sun, Zhiyuan Ma, and Lei Zhang. One-step effective diffusion network for real-world image super-resolution, 2024.

[23] Zongsheng Yue, Jianyi Wang, and Chen Change Loy. Resshift: Efficient diffusion model for image super-resolution by residual shifting, 2023.

[24] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks, 2018.

APPENDIX

A HFN EXAMPLE



(a) LR (b) HFN

Fig. 6: Example of LR bands upscaled to 10 m/px using HFN. In the (a) LR column, the top image is made from bands with a 20 m/px resolution, and the bottom image is one band in greyscale with a 60 m/px resolution. Both images in (b) HFN column are in the final 10 m/px resolution after being processed by HFN.

B TASKS

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.

Table 6: TABLE OF OBJECTIVES.