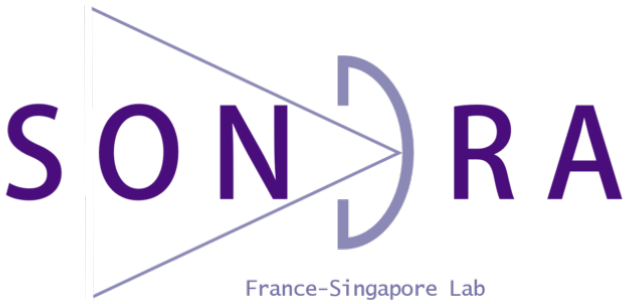
CentraleSupélec

Metz

PFE 2021-2022

Report

Deep Learning for SAR Image Super Resolution



Students : COLOMBO Daniel, ADARRAB Youssef, DALY Alexandre

Supervised by : FIX Jeremy (CentraleSupélec), REN Chengfang (Sondra), HINOSTROZA Israel (Sondra)

Contents

1 Introduction

With the advent of high computing power and high availablity of data, the possibility of extracting insighful information from low quality data is becoming more and more essential in many fields such as medical imaging, astronomical imaging or SAR remote imaging. Super-resolution is a technique that aims to reconstruct high resolution images from a low resolution input.

Our study focuses on applying super-resolution to SAR data, to enhance to acquired data quality, as high resolution SAR images are highly in demand for tasks such as land classification. We use the UAVSAR dataset provided by NASA and we focus on images of cities in the American Bay aera.

This work investigates two super-resolution different deep learning approaches, a more traditional network, the SRCNN model, and the SwinIR model, recently introduced and ranked at the top of most super-resolution benchmarks. We also investigate the effectiveness of multiple losses, such as the L1, L2 or SSIM losses, in order to find the one that produces the best outputs.

The code used in this project was mainly inspired by the model implementations available in these repositories:
• https://github.com/cszn/KAIR
• https://github.com/yjn870/SRCNN-pytorch

This report contains the following sections: Section 2 reviews previous work in the field of super-resolution. Section 3 outlines the data and preprocessing used in this study. Models tested are presented in Section 4, and the methodology followed to train them is presented in Section 5. Finally Section 6 concludes the report with a summary of our quantitative results and findings.

2 Related Work

2.1 Super-resolution overview

Super-resolution is a process that aims to reconstruct high-resolution images from a low-resolution input. The loss of resolution usually occurs because of multiple factors, such as blurring, compression or signal processing. Most of the time, it is due to a low-resolution camera or remote system capturing the data. Super-resolution is an important field of computer vision and image processing, it has numerous applications in various fields, such as satellite imaging, medical imaging or astronomical imaging systems.

We can model super-resolution by a degradation mapping function D, such as blurring, applied to high-resolution I_y images and that produces a degraded output of low-resolution I_x.

$$I_x = D(I_y; \delta) \tag{1}$$

We can easily obtain a degraded image I_x from a high-resolution output I_y, and the parameters of the degradation model δ, such as the noise, but the inverse problem of reconstructing a high-resolution image from a low-resolution one can be tricky, and is an ill-posed problem, when we do not exactly know the degradation mapping used. Indeed, we would not be able to simply take the inverse of the degradation function D^{-1}.

With super-resolution, we try to reconstruct a high-resolution image, $\hat{I}_y$, with a super-resolution model F, and its parameters θ.

$$\hat{I}_y = F(I_x; \theta) \tag{2}$$

In the ideal case, we would have $\hat{I}_y = I_y$, and the reconstructed image would be a perfect copy of the original high-resolution image. In the following sections we will present multiple approaches used to perform super-resolution, and we will particularly emphasize on Deep Learning methods.

2.1.1 Interpolation based super-resolution

Before machine learning became the norm in computer vision and super-resolution, methods based on interpolation were commonly used, such as bicubic interpolation and Lanczos resampling.

Bicubic interpolation estimates the value of a pixel based on the surrounding pixels. It takes a weighted average value of the 16 surrounding pixels to calculate the interpolated value. The value of the interpolated image is given by :

$$B(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} A(x_i y_j) \tag{3}$$

where A is the original image point, B is the interpolated image point, a_{ij} represents the weight parameter, and i, j represent the horizontal and vertical coordinates, respectively.

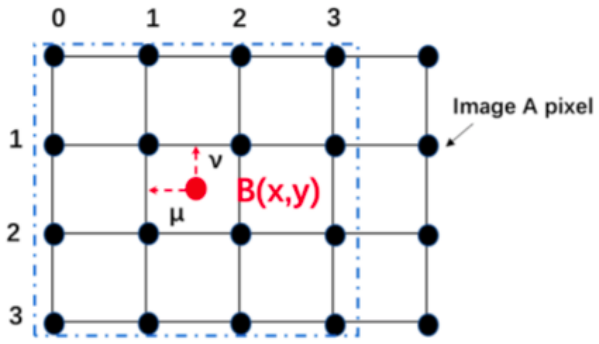


Figure 1: Illustration of the bicubic interpolation method

Lanczos resampling uses a low-pass filter applied to a digital signal to interpolate its values. Lanczos resampling is typically used to increase the sampling rate of a digital signal. The filter's reconstruction kernel, L(x) defines the effect of each input sample on the interpolated values, it is called the Lanczos kernel. It is composed of a sinc function sinc(x), multiplied by the Lanczos window, or sinc window, which is the central lobe of a horizontally stretched sinc function $sinc(x/a)$ for $a <= x <= a$.

$$L(x) = \begin{cases} \operatorname{sinc}(x)\operatorname{sinc}(x/a) & \text{if } -a < x < a, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

In an equivalent manner :

$$L(x) = \begin{cases} 1 & \text{if } x = 0, \\ \dfrac{a\sin(\pi x)\sin(\pi x/a)}{\pi^2 x^2} & \text{if } -a \leq x < a \text{ and } x \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The parameter a, a positive integer, determines the size of the kernel. The Lanczos kernel has 2a 1 lobes.

Given a signal with samples si, for integer values of i, the value S(x) interpolated is obtained by the discrete convolution of those samples with the Lanczos kernel:

$$S(x) = \sum_{i=\lfloor x \rfloor - a + 1}^{\lfloor x \rfloor + a} s_i L(x - i) \tag{6}$$

where a is the filter size parameter and $\lfloor x \rfloor$ is the floor function.

But these methods, even if they were fast and did not require data to use as example, were highly inaccurate because of the lack of information about how to determine a pixel's value. Furthermore, interpolation-based SR models do not take into consideration the image edge structure in their process, and this can cause distortions in the image edges.

2.1.2 Reconstruction based super-resolution

A another type of approaches used to generate high-resolution images from low-resolution ones is the reconstruction based methods or sparse coding methods. These methods usually incorporate a parametric prior distribution of the high-resolution image to put a constraint on the reconstructed image. Various models can be used for super-resolution tasks, such as the iteration-back-projection IBP, the maximum a posteriori (MAP), and the Total Variation (TV) models. However, reconstruction methods can lead to poor high-frequency details with presence of blur, degrading the quality of the image. Moreover, incorporating prior information about remote sensing images, such as SAR data, represents a challenge because of their complex nature, thereby limiting the efficiency of such methods.

2.2 State-Of-The-Art of Deep Learning Super-Resolution

2.2.1 Supervised Learning based super-resolution

In this section we will focus on learning approaches that aim to reconstruct a high-resolution image from a low-resolution one. We treat the problem as a supervised learning one, where the HR image is used a target as the LR image an input. The overview of the different methods is largely based on the following article [Wang et al.(2019)Wang, Chen, and Hoi], and the following blog articles [Khandelwal()] and [Matcha()].

The figure 2 summarizes the different techniques used in supervised learning and Deep Learning to realize super-resolution.

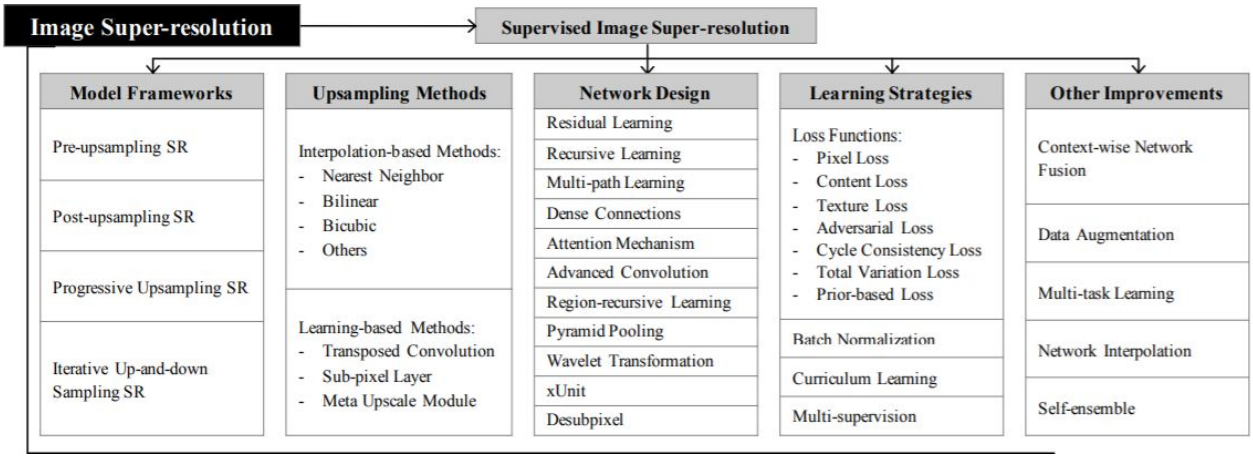


Figure 2: Overview of topics used for image super-resolution

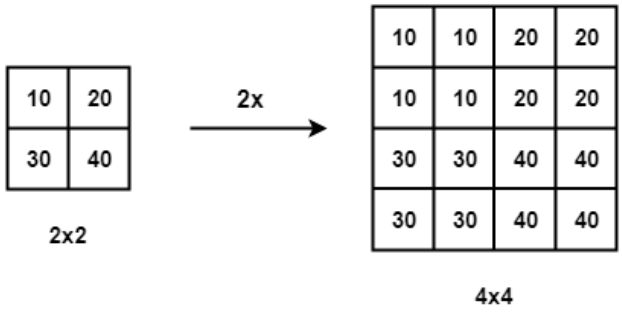


Figure 3: Illustration of the upsampling process

Before diving into the theory behind deep learning models that perform super-resolution, we will speak about upsampling, as illustrated in 25.

Upsampling refers to the process of increasing the number of pixels rows or columns or both in a image. It can be done with interpolation methods presented above, like bicubic interpolation or Lanczos upsampling.

Deconvolution Upsampling can also be realized with learning methods, in an end-to-end way, with transposed convolutions for example. A layer of transposed convolution or deconvolution is illustrated in the figure 4.

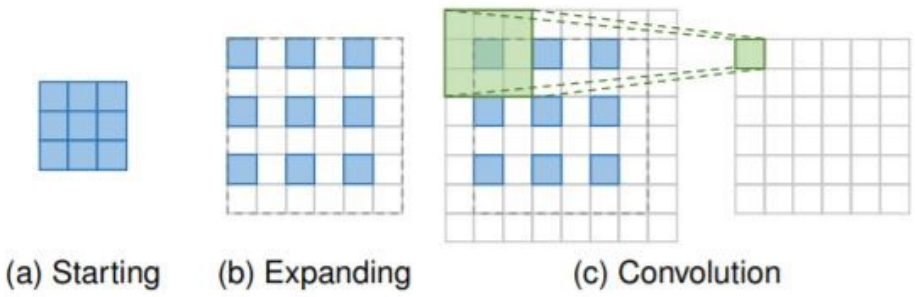


Figure 4: A transposed convolution layer

In the layer, we give as image as an input, represented in blue, with a convolution kernel in green that generates a picture with more pixels. The result is an upsampled image, whose resolution is higher.

Sub-pixel upsampling Another end-to-end upsampling method based on learning is the sub-pixel layer, illustrated in figure 5, it takes a low-resolution picture in input, represented in blue, and generates multiple

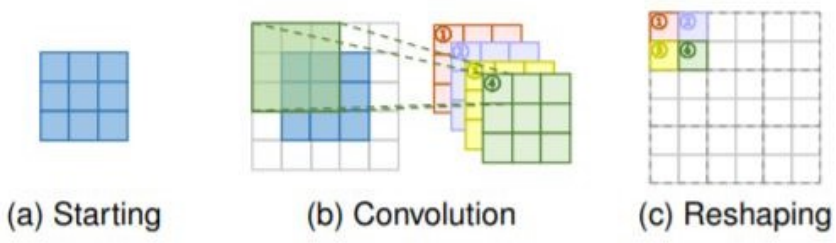
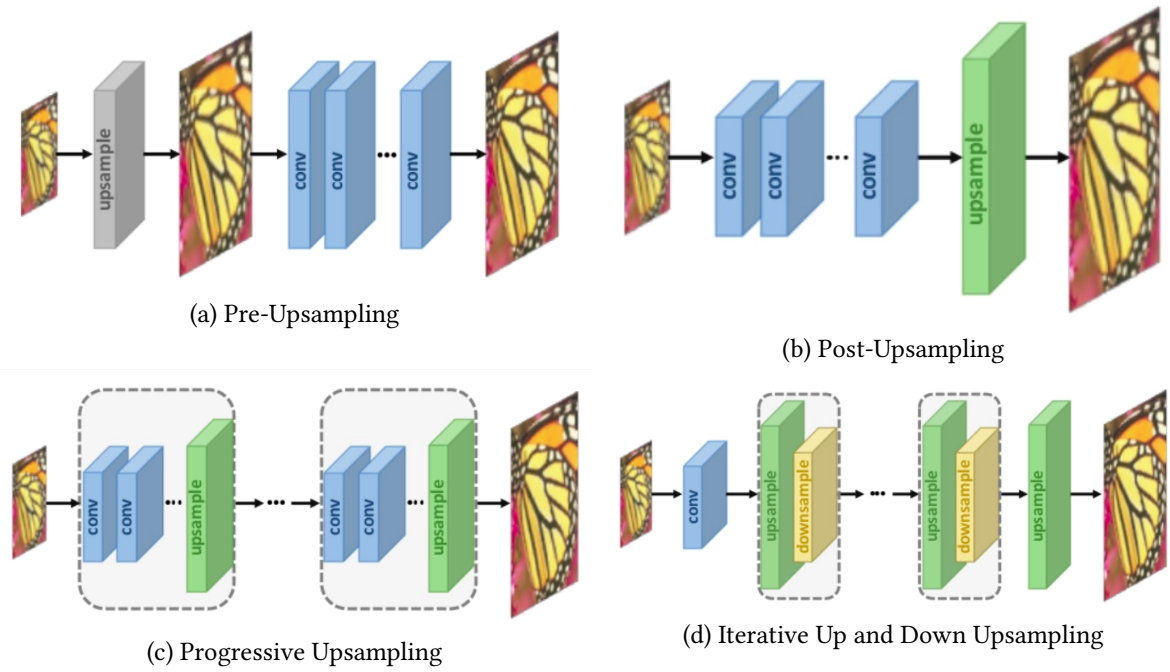


Figure 5: A sub-pixel layer

channels using different convolution kernels, represented in the figure in green, red and other colors. With a scaling factor of s, the sub-pixel layer generates s^2 channels. Therefore if we give an input of size h x w x c to the layer, we receive an output of size h x w x s^2c. After reshaping the pixels to produce the final image, we generate an output of size hs x ws x c. The sub-pixel layer will be used to define the SwinIR model used for super-resolution applied to our SAR dataset.

As presented in the table 2, super-resolution deep learning models can be classified into multiple frameworks. The group they belong to is defined by the nature of upsampling methods used and their position in the network.



(a) Pre-Upsampling

(b) Post-Upsampling

(c) Progressive Upsampling

(d) Iterative Up and Down Upsampling

Figure 6: Model Frameworks

Pre-Upsampling In this method, illustrated in figure 6a we don't do a direct mapping of low resolution images to high resolution images, since it is not an easy task. The low resolution images are first upsampled, giving a coarse high resolution imgae, using traditional methods (such as Bilinear interpolation) and then we use CNNs to refine the bigger image into a proper high resolution one. The advantage of this method is that we don't need to learn a direct mapping between the low resolution and high resolution image, we only need to learn how to refine a coarse image into a high-quality one.

Post-Upsampling In the post-upsampling method, the low resolution images are passed to the CNNs as they are. At the end of the network, we place learnable upsampling layers. This method is thereby an end-to-end deep learning trainable one. It is also more computationally efficient, because the feature extraction part

where most of the computations are done, is performed in a low-dimensional space, before the upsampling, as illustrated in 6b.

Progressive Upsampling In the post-upsampling method, a solution to reduce the complexity was proposed, but it was relying only on a single upsampling convolution layer. For large scaling factors, the learning process is still hard. Progressive upsampling was proposed to resolve this issue, in models such as Progressive SR (ProSR) [Yifan et al.(2018)Yifan, Perazzi, McWilliams, Sorkine-Hornung, Sorkine-Hornung, and Schroers] and Laplacian Pyramid SR Network (LapSRN) [Lai et al.(2017)Lai, Huang, Ahuja, and Yang]. The models cited above, progressively reconstruct the high resolution images with multiple CNNs and upsampling layers, as visible in figure 6c in order to simply the problem by dividing into small steps, with smaller scaling factors.

Iterative Up and Down Upsampling The final method is iterative up and down upsampling. It is based on the Hourglass architecture, and it is a type of convolutional encoder-decoder network that uses convolutional layers to break down and reconstruct images.
There are some variants such as the Stacked Hourglass network use several hourglass structures in series, effectively alternating between the process of upsampling and downsampling, an example an up-an-down network, the U-net architecture, is given in [Ronneberger et al.(2015)Ronneberger, Fischer, and Brox]. With this family of models, the relations between low resolution and high resolution images are captured in a better way, the quality of the reconstructed images is therefore higher.

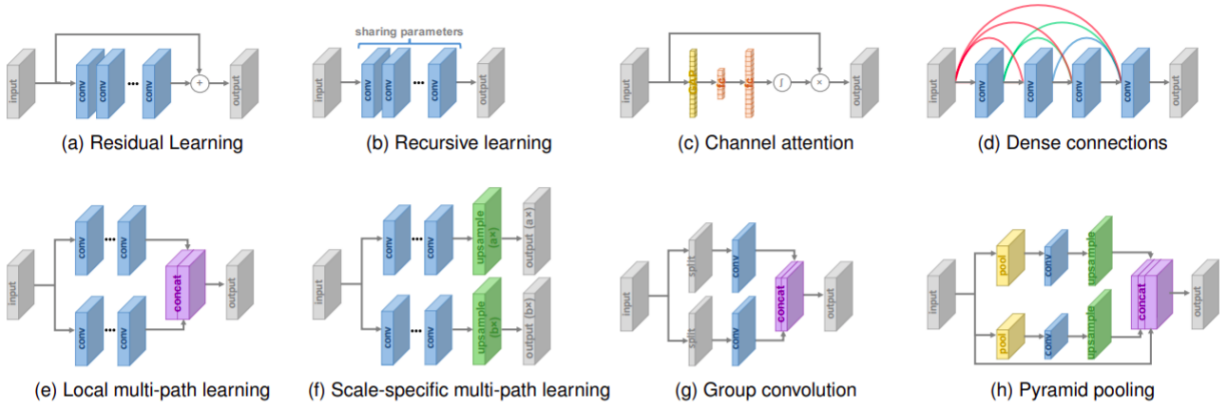


Figure 7: Network design strategies

Network design Apart from usual 2D convolutions, many variants can be used in network designs to create a super-resolution architecture. Many design strategies can be used, such as Skip connections, Spatial Pyramid Pooling and Dense Blocks, as shown in the figure 7 aiming to mix between low level and high level features, for an optimal performance.
In the following, we are going to present some state-of-the-art models for super-resolution.

Residual networks:

EDSR The ESDR (Single-Scale model) architecture, introduced in [Lim et al.(2017)Lim, Son, Kim, Nah, and Lee], is based on the SRResNET design, composed of multiple residual blocks. We can notice on the figure 8, that the residual block of the EDSR network doesn't have in Batch-Normalization layer.

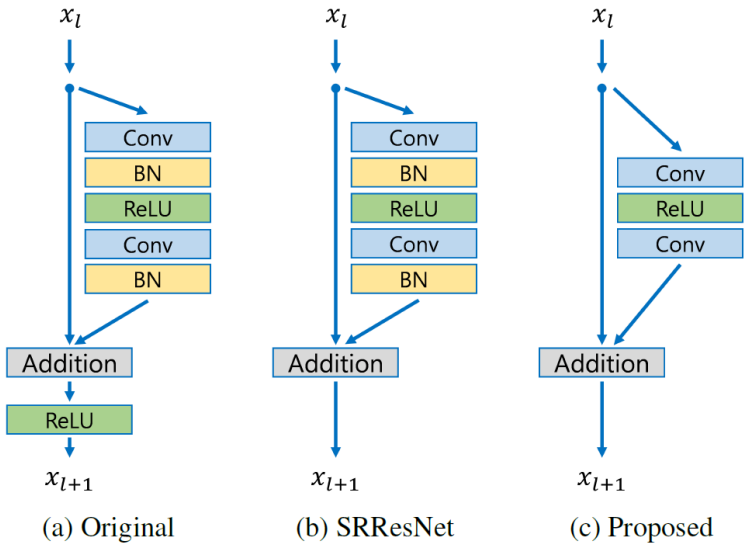


Figure 8: EDSR and SRResNet architectures

Indeed, deleting the normalization improves the performance, in terms of accurcy and memory, because it removes the limitation of the network's range and leads to a smaller memory consumption.

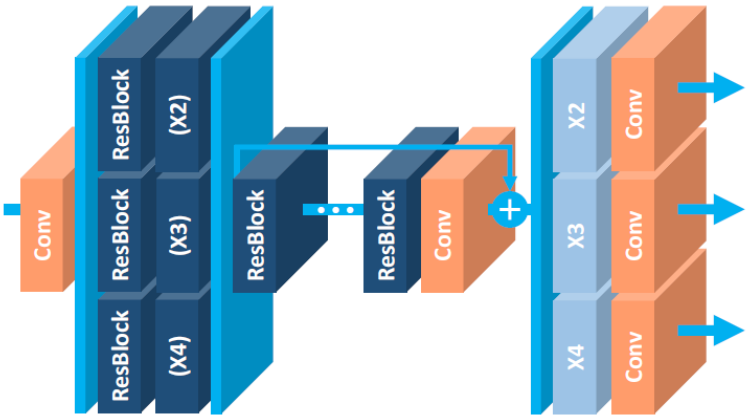


Figure 9: Network design strategies

MDSR The MDSR (Multi-Scale model) is an extension of EDSR, it uses a deeper and wider architecture to improve performance, with a total depth of 5x bigger than EDSR. Even with a 5x deeper architecture, the parameters are only multiplied by 2.5. The model has numerous input and output modules that give a 2x, 3x and 4x better resolution. The network contains shared residual blocks for all the modules as shown in figure 9.

Recursive networks :

Recursive networks use shared parameter weights in convolutional layers to be more memory-efficient. Some of the architectures involving recursive units.

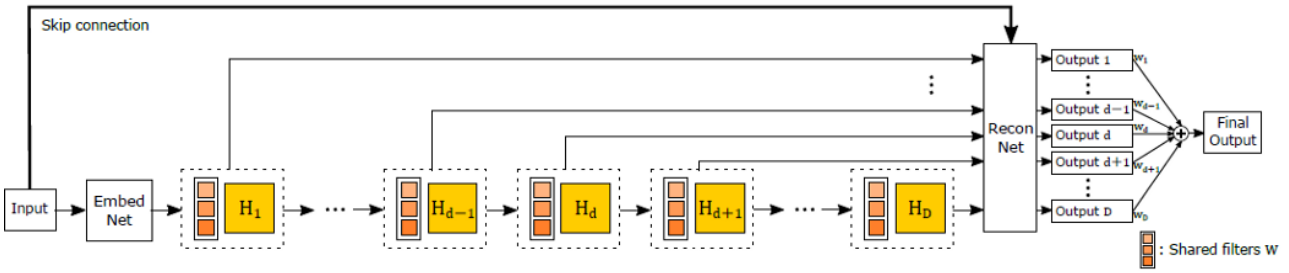


Figure 10: Network design strategies

DRCN Deep Recursive Convolutional Network (DRCN), [Ghifary et al.(2016)Ghifary, Kleijn, Zhang, Balduzzi, and Li] is an architecture where the same convolution layer is applied many times. As illustrated in the figure 10, the convolutional layers in the residual block are shared.

The model can be viewed as en ensemble of multiple networks. Indeed, all the outputs resulting from the intermediate shared blocks are sent to the reconstruction layer that generates the output.

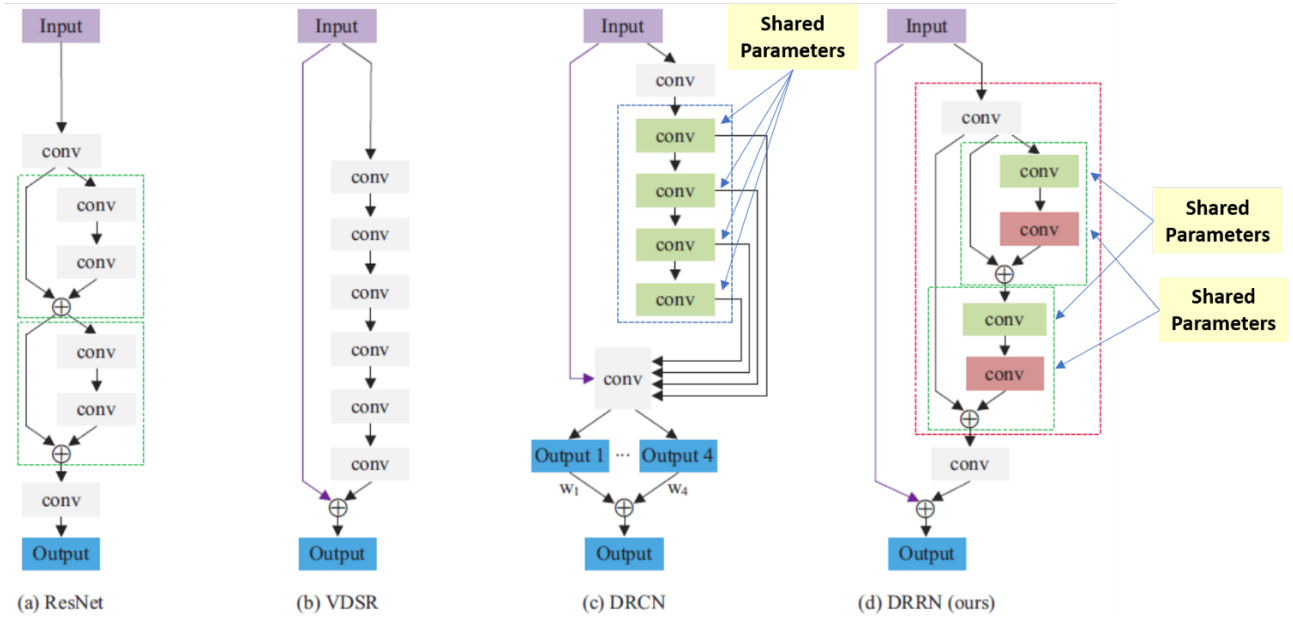


Figure 11: Comparison between ResNet, VDSR, DRCN and DRRN

DRRN Deep Recursive Residual Network (DRRN), introduced in [Tai et al.(2017)Tai, Yang, and Liu] is an architecture that improves DRCN by using residual blocks instead simple convolutional layers. Every residual block shares the parameters weights with the other residual blocks. The comparison between a classical model such as ResNet, DRCN and DRRN, can be seen in the figure 11.

Generative models :

Generative models such as GANs use the perceptual quality of the images produced to try to output the most optimal high resolution reconstructed image. The result is more natural and smooth to the human eye.

SRGAN Super-Resolution Generative Adversarial Network (SRGAN), introduced in
[Ledig et al.(2016)Ledig, Theis, Huszar, Caballero, Aitken, Tejani, Totz, Wang, and Shi], uses a GAN-based architecture to generate visually pleasing images. It relies on a SRResNet as a backbone and uses a three-term loss to optimize the results. SRGAN uses a traditional MSE loss, capturing pixel similarity, a perceptual loss

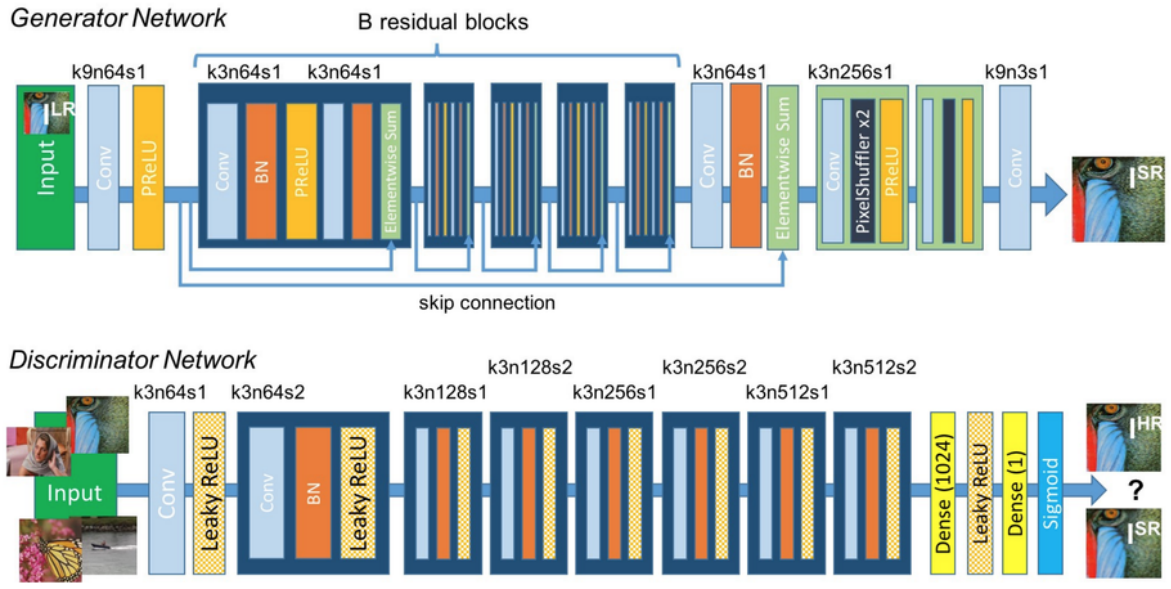


Figure 4: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

Figure 12: Network design strategies

and an adversarial loss for the discriminator. The process consists of producing images from noise that will be given to a discriminator network that will try to distinguish between a true high resolution image and a super-resolution reconstructed image. Finally, the GAN loss is backpropagated to train the generator and discriminator models.

3 Data

3.1 Fundamentals of Radar and SAR

3.1.1 Historical overview of Radar imaging

RAdio Detection And Ranging, or RADAR, was invented in the 20th century as a tool for detecting and knowing the localization of objects in a three-dimensional space.
The invention is usually either attributed to the German inventor Christian Huelsmeyer, who created the "Telemobiloskop", a microwave-based system that aims to detect far metallic objects, or to the British engineer Robert Watson-Watt who proposed a system able to detect and locate aerial objects up to 30 km.
The technology was developed to become small enough to be placed on airplanes, enabling new applications of radar systems, such as the discipline of Earth observation.
In the next sections, we will discuss the application of imaging radar sensors, mainly based on the work in the book [Flores et al.(2019)Flores, Herndon, Thapa, and Cherrington], Side Looking Airborne Radars (SLAR), their limitations and how Synthetic Aperture Radar (SAR) was proposed as solution.

3.1.2 Side Looking Airborne Radars (SLAR)

Radar systems were used for imaging purposes, given that these type of sensors are capable of providing data in all-weather and all-day conditions. It became possible to map areas that were covered by clouds, rain or darkness. Radars sensors could also provide insightful information about the surface they were interacting with. In the second half on the 20th century, the first airborne radar systems were available with the devel-

opment of Side-Looking Airborne Radar (SLAR). The SLAR system observation is done with a radar sensor placed on an airborne that is flying at a fixed altitude H in a straight line.

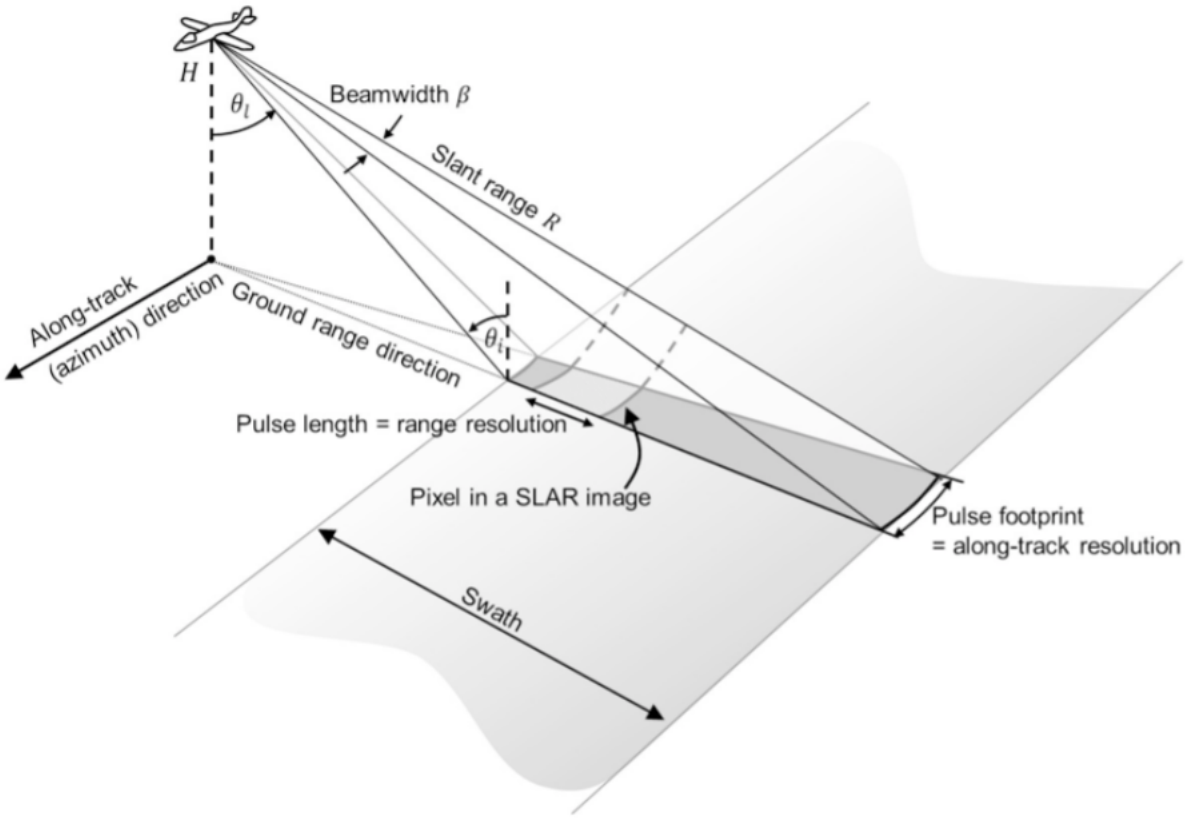


Figure 13: Geometry of a SLAR imaging system

The SLAR system is different from common optical systems, its antenna is not pointing towards the nadir, but it observes the earth with an angle of θ_i, called the look angle. The system sends a beam that illuminates a strip on the observed surface.
When the aircarft is flying, the SLAR system sends short microwave pulses of length τ_p, that illuminate the dark area on the fig 1, known as the antenna illumination footprint, of a size S.
The size S in the azimuth or range directions are defined by both the wavelength λ and the side length of the antenna L in the considered direction, i.e the antenna's beamwidth, $\beta = \lambda/L$.
The size S of the illumination footprint along the azimuth or range direction is given by the following formula:

$$S \approx (R.\lambda)/L = R.\beta \tag{7}$$

With R being the distance of the sensor from the surface observed.

The echoes collected from the ground are identified by their arrival time in both range and azimuth direction to generate a two-dimensional image. Echoes from the surface observed arrive increasingly later in range direction, from the near-range to the far-range edges of the strip observed. If the range separation between objects is greater than half the transmitted pulse length, they can be discriminated as objects from separate ranges.

As a result, a SLAR system's range resolution is defined by :

$$\rho_R = \frac{c.\tau_p}{2} \tag{8}$$

With c corresponding to the speed of light.

The variable ρ_R in eq (2) is called the slant range resolution of a SLAR system as it describes a SLAR's ability to distinguish objects at diferent (slant) distances from the radar.
Remote sensing is usually more interested in the ground resolution ρ_G, which is derived from ρ_R and the angle θ_i, defined previously. This variable quantifies the sensor's abilty to distinguish objects located on the surface of the earth.

$$\rho_G = \frac{\rho_R}{sin(\theta_i)} \tag{9}$$

The ground resolution is not constant, it becomes better when the airborne is far from the nadir, as the incidence angle θ_i increases. This behaviour is opposite to most optical imaging systems, that suffer from a resolution loss when the angle θ_i becomes higher.
In the azimuth direction, the surface is illuminated as the airborne moves along its track. The azimuth resolution for SLAR systems can be interpreted as the ability to discriminate objects in the azimuth direction.
It is defined as ρ_{A_Z}, the width of the antenna footprint S_{A_Z}, which is highly limited by the length of the antenna in the azimuth direction, L_{A_Z}.

The equation for the azimuth resolution is given by :

$$\rho_{A_Z} = S_{A_Z} \approx \frac{\lambda}{L_{A_Z}}.R = \beta_{A_Z} \tag{10}$$

The equation (4) indicates that the radar imaging system's azimuth resolution is degraded in a linear way with the increase of the distance between the sensor and the ground R.
This implies that for SLAR systems, the resolution is smaller when the airborne carrying the sensor is flying at high-altitude, i.e when the R becomes large. It is therefore not practical to obtain good images for high-altitude systems.
To obtain a good azimuth resolution, a solution consists of increasing the length of the antenna L_{A_Z} to decrease S_{A_Z} and reach the desired resolution ρ_{A_Z}. However, this approach can rapidly become unrealistic, as the required antenna length becomes too long to be actually used in an real-world system.
To solve this issue, a more practical solution was invented to address the problem of the azimuth resolution, the Synthetic Aperture principle, which will be introduced and developped in the following section.

3.1.3 Synthetic Aperture Radars (SAR)

In 1952, Carl Wiley, made an major discovery that resolved the issue related to the azimuth resolution that was encountred when using SLAR systems. He observed a similarity between the along-track position of an objected illuminated and the Doppler effect of the signal reflected back at the radar. A frequency analysis of the signals could help obtain a better resolution that that obtained from traditional SLAR systems. His discovery was first published as the Doppler beam-sharpening but is usually called aperture synthesis, and it is at the core of all current high-resolution imaging radar systems.

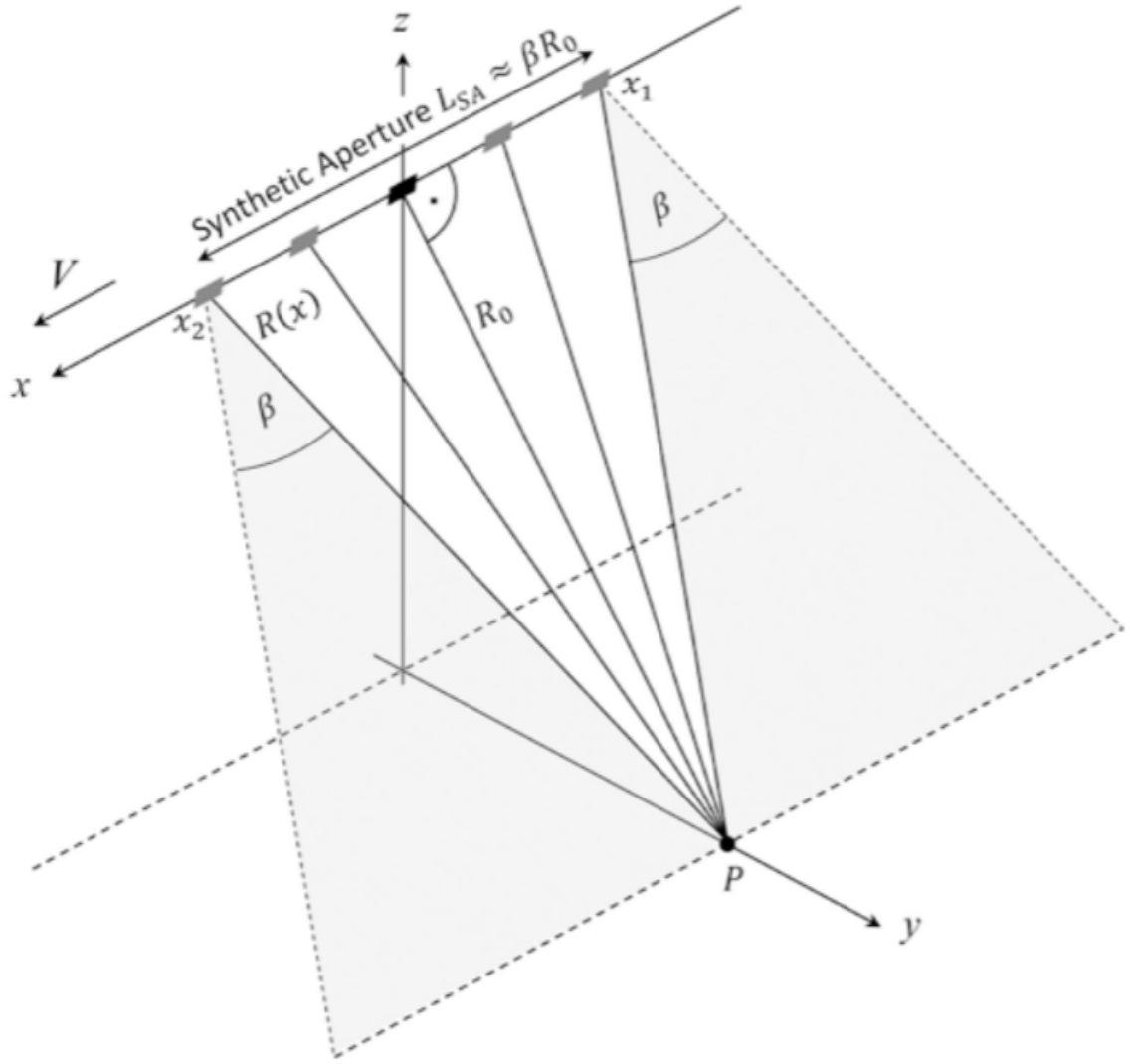


Figure 14: Geometry of a SAR imaging system tracking an object P

The principle of aperture synthesis enables a system to create a longer effective antenna, called synthetic aperture, from an ensemble of acquisitions made with a shorter antenna, as it moves alongs the flight track. As the resolution capability of a radar system is, by definition, related to the length of the acquisition antenna, the much longer antenna synthesized by the Wiley's effect enables to get a higher resolution from an airborne with a relatively small-sized antenna.

To illustrate this concept, the figure 2 show a radar antenna, represented by a gray rectangle, moving along the flight path from the right to the left with a velocity V. It continously transmits short radar pulses and receives the echoes returned from objects on the surface observed. Synthetic aperture radars are opposed to SLAR using real antennas, where azimuthal resolution is simply achieved by using a transmiting/receiving antenna with a narrower antenna lobe along the azimuthal direction. This allows the synthetic aperture radar to use a relatively small antenna to achieve a high resolution which is not dependent on the height of the radar holder. For SLAR systems, the limited length of the radar antenna results in an illuminated footprint size of multiple kilometers, degrading the resolution.

This is illustrated in the figure 2, as the antenna goes from the position x_1, where the object P is first observed to the position x_2 where the object is last observed. In a nutshell, SAR enables to create high-resolution images compared to small physical antennas. For a fixed antenna size and orientation, objects which are further away remain longer illuminated. Thus, SAR has the property of creating larger synthetic apertures for more distant

objects, which results in a consistent spatial resolution with the distance.

3.2 UAVSAR Data

There are many datasets available on the internet. We have chosen to use a dataset provided by the NASA (UAVSAR) which has been designed to study many physical phenomena on Earth. In particular, this data can be used to study earthquakes, volcanic phenomena (and other land deformations), ocean evolution, soil changes, glacier melting and vegetation evolution. UAVSAR stands for Uninhabited Aerial Vehicule Synthetic Aperture Radar and has collected thousands of flight lines. Hereafter you can find an example of a SAR image taken from the UAVSAR dataset above the Los Angeles Bay.

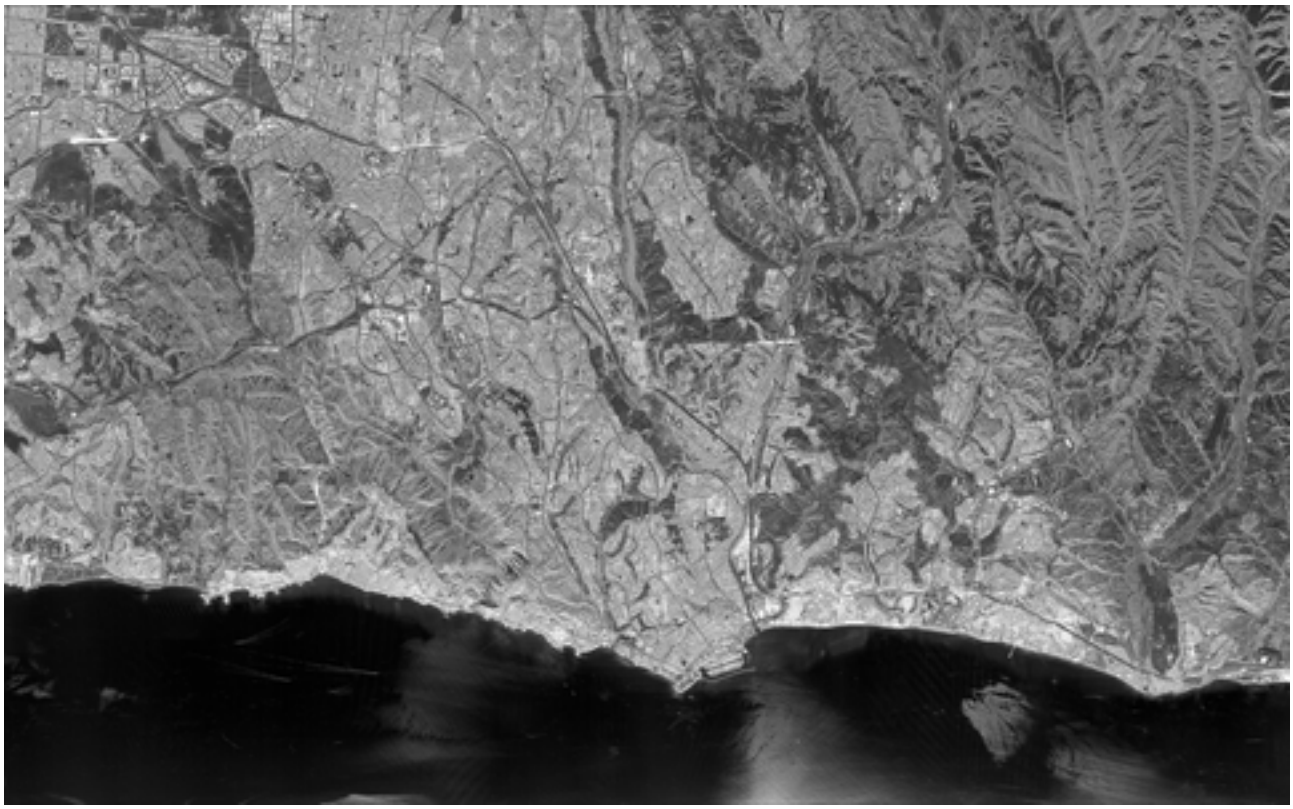


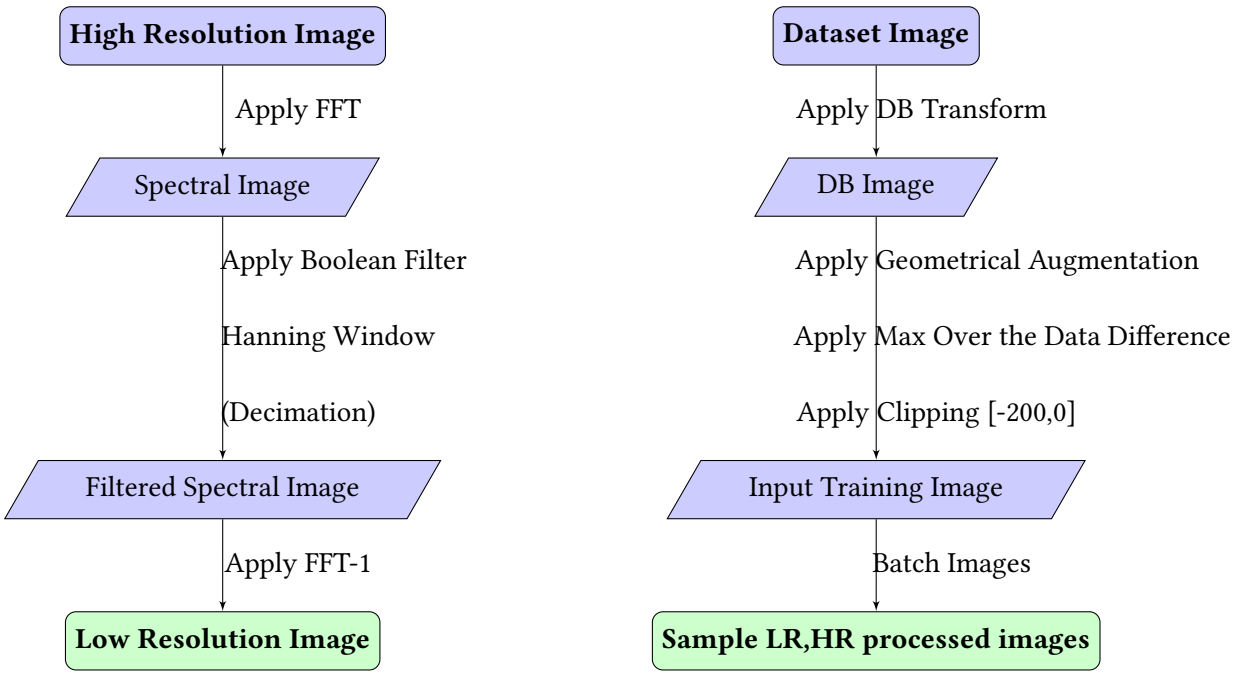
Figure 15: SLC Image Example

The images are often divided into segments. For a given region, you can have multiple segments taken at different times. There is also the possibility to choose different polarizations images (HH, HV, LL). We have chosen to take only the HH polarization for a first implementation. If we concatenate the 3 polarization, we can visualize a RGB like image. We have also chosen to work only with city images since we are interested in reconstructing details in low resolution images.

3.3 Dataset Generation

3.3.1 LR Image Generation Pipeline

Once we collected the dataset, we need to generate low resolution images from high resolution images that are the SAR SLC stacks. To do so, we apply some signal processing methods that enable to remove some features in the spectral space. The whole pipeline consists in first apply a Fast Fourier Transform (FFT) to our input SLC slack. Then we apply some filters (boolean filter + hanning window) in the spectral space that enable to remove some spectral features (typically the high frequencies, details of the image). Finally, we apply the inverse fourier transform (FFT^{-1}) to go back to the pixel space (we denote the points as pixels in the rest of the report). We get an image that has a smaller resolution. We can choose the downgrading factor when applying the low pass filter. The whole pipeline is summarized in figure 16.



(a) Dataset generation from UAVSAR

(b) Transformation Pipeline for NN Training

Figure 16: Pipeline for Generating Training Images

At the end, we get samples for our dataset as shown in figure 17a, 17b.

Hanning Window and Boolean Filter The boolean filter in the spectral space is a filter the value of which is 1 inside a rectangle and 0 outside. This a pure binary filter that thresholds the frequencies. It completely ignores all the frequencies that are outside the rectangle. One can have a better visualisation in figure 18a. The Hanning window consists in smoothing the boolean filter avoiding too much noise when reconstructing the image. The Hanning window (shown in figure 18b) is applied to the thresholded spectrum to smooth it even more. The hanning window is basically a gaussian 2D filter.

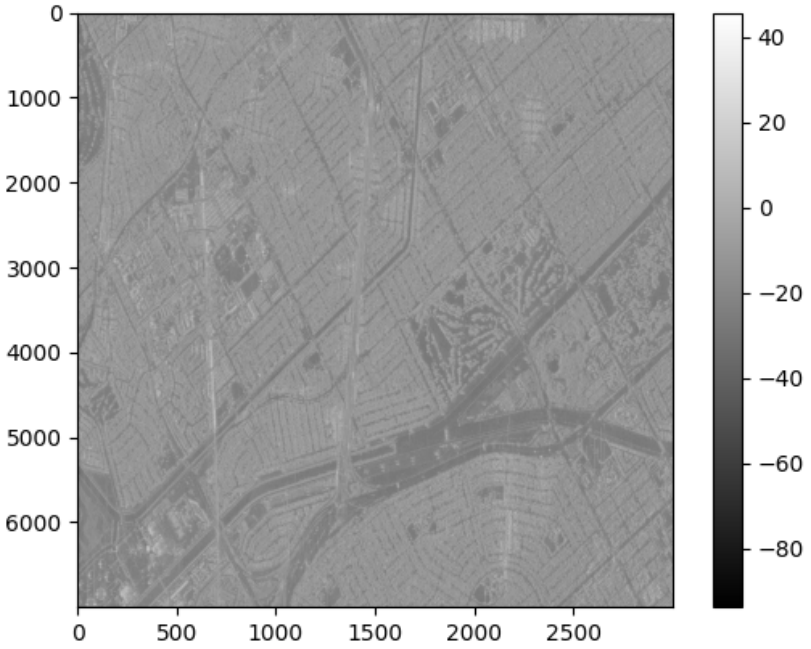
3.3.2 Training Validation and Test Sets

The SLC images are usually massive. Their resolution is often around 10 000px x 60 000px. Such images cannot be used directly for training a network. We need to generate smaller images to be able to train a decent network. To do so, we decided to cut small subimages from the big SLC image the size of which will be 256px x 256px. This parameter can be adjusted whenever we want, we just need to regenerate the dataset each time we decide to use a different size.

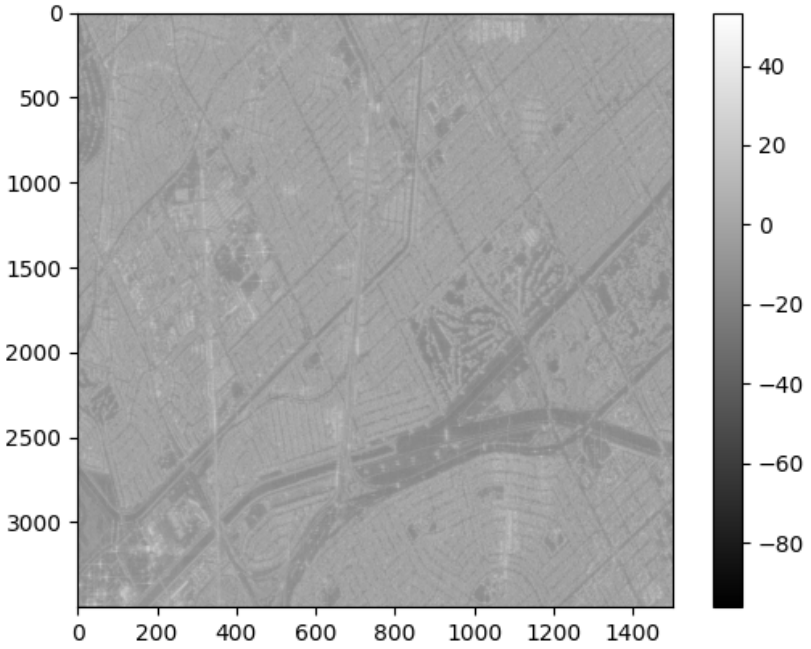
In the end, we get back a dataset including high resolution images as well as the corresponding low resolution images. For a single SLC slack, we can generate 10 000 subimages with a size of 256. That is the reason why we decided for the first implementation to work only with one or two SLC slacks. We then split the data into a training and validation sets with a training ratio of 80%.

3.4 Preprocessing

After generating the dataset, we apply some transformations on the images. First, we applied an intensity to decibel transformation since the raw values are very sparse. The logarithm enables the values to have a smaller range. Hereafter, we retrieve the maximum pixel value over the dataset to the whole dataset. This leads to have a range of value between - ∞ and 0. We observed now that the smallest values could be interpreted as noise. When plotting the histogram of the values (figure 20) we noticed that the pixels were distributed exponentially following a gaussian like distribution. To remove these values that potentially would not bring



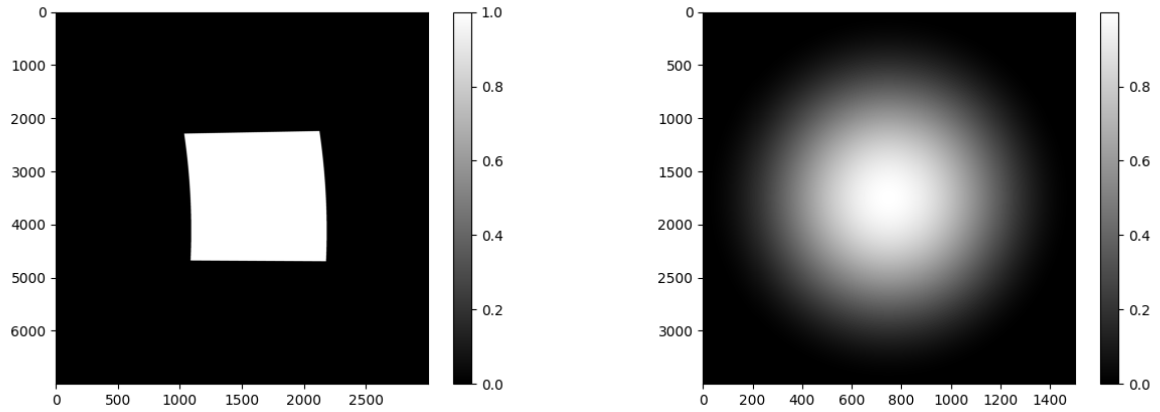
(a) HR Image : 3000x7000 px



(b) LR Image : 1500x3500 px

Figure 17: (a) High resolution image cropped from an SLC slack. The x axis is the azimuth axis and the y axis is the range axis. The colorbar represent the intensity in DB. (b) 1500x3500 px Low resolution image generated by the above pipeline.

any further information, we applied a clip transformation to threshold the smallest values. We clip all the images in the $[-200, 0]$ range to ensure noisy points are removed.



(a) Boolean Filter

(b) Hanning Window

Figure 18: Filters

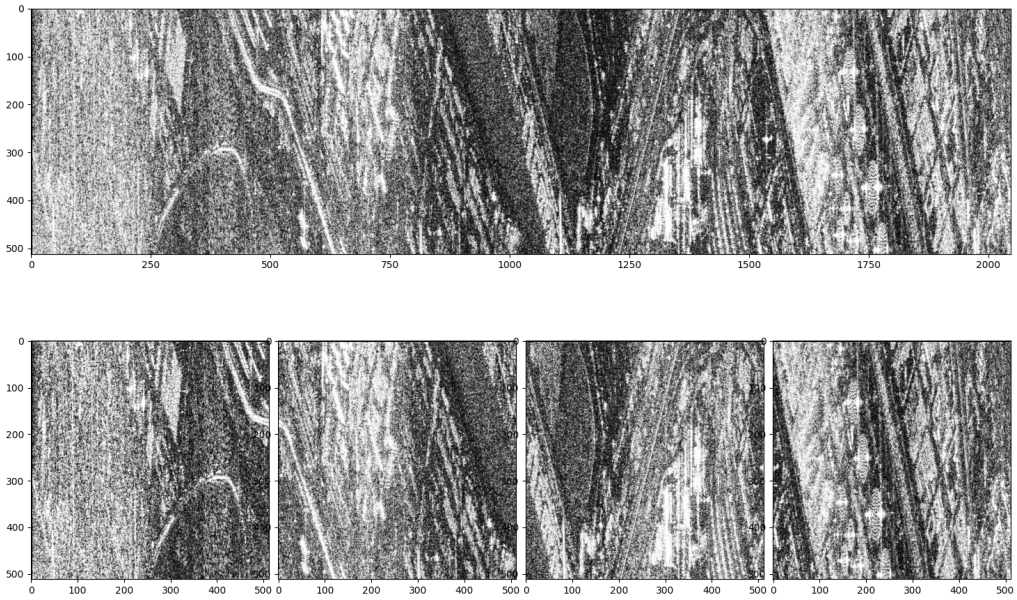


Figure 19: Subimages Generation from a Bigger SLC slack

We applied some augmentations on the training set as well. These include horizontal flip, vertical flip, rotation or a combination of both. We randomly apply a transformation given a certain probability. We have chosen to perform augmentation to diversify the input data. Figure 16b shows a summary of the transformation Pipeline. We provide figure 20 to better visualize what happens to the images during the pipeline. We also plot the histograms of the pixel values to have an representation of the pixel distribution over the image.

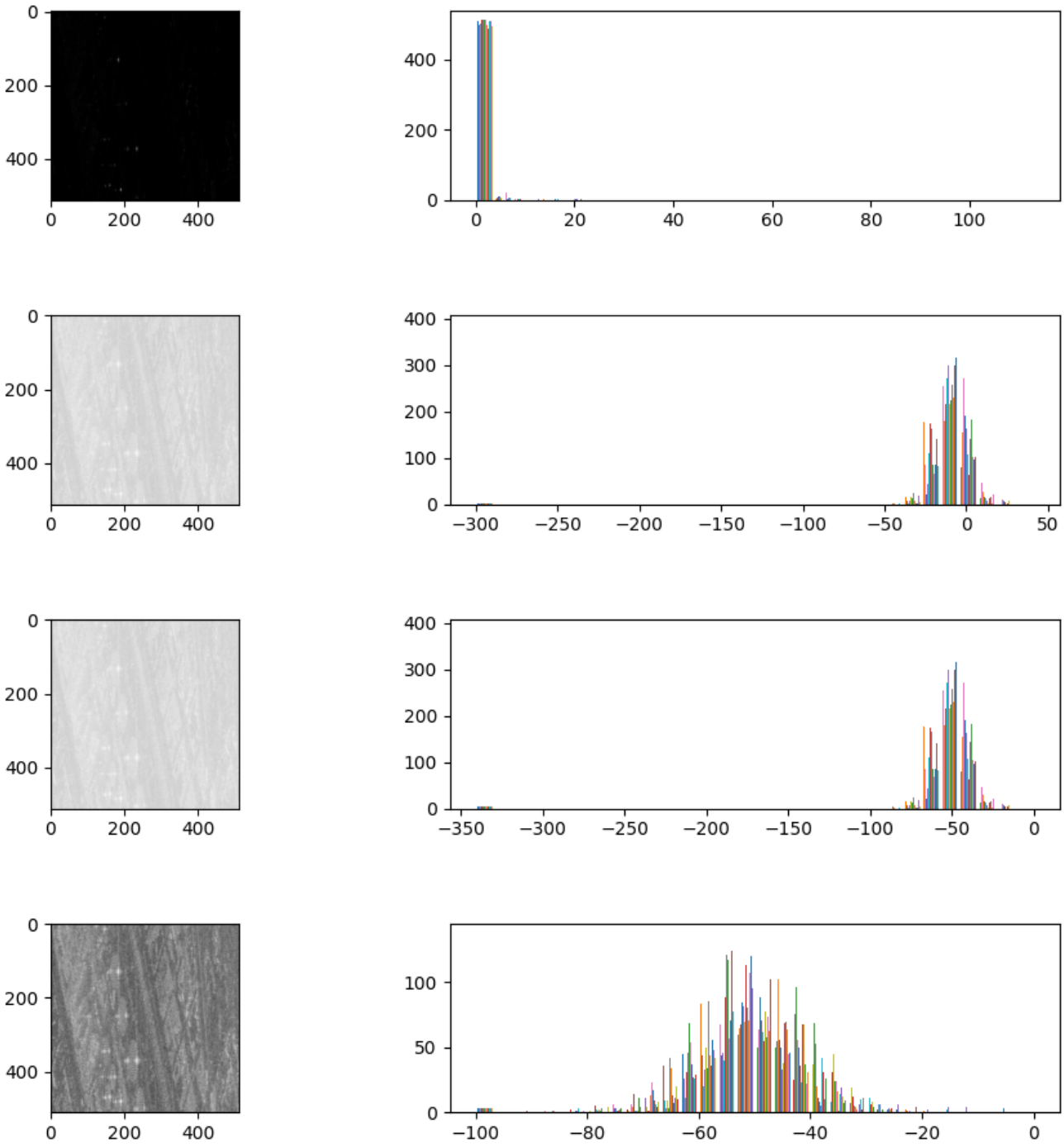


Figure 20: Transformation pipeline. For each step in the transformation pipeline, the associated histogram has been plot. Step 1 : apply db transformation. Step 2 : retrieve max. Step 3 : clip the values in the [-100,0] range

4 Models

4.1 SwinTransformer

Transformer Transformers have revolutionized deep learning with the self-attention mechanism. It allows to capture global interaction between parts of the input : at first it was used for Natural Language Processing in order to capture interaction between words and/or parts of sentences, but it is now also used in computer vision where tokens are substituted by windows of images. The Swin Transformer is an algorithm introduced by Microsoft researchers in [Liu et al.(2021)Liu, Lin, Cao, Hu, Wei, Zhang, Lin, and Guo] that uses this mechanism with the additional particularity of "shifted window". The SwinIR algorithm presented in [Liang et al.(2021a)Liang, Cao, Sun, Zhang, Van Gool, and Timofte] is the adaptation of the Swin Transformer to Super-resolution tasks. It has the advantage of having few weights and great PSNR performances compared to other existing image SR methods [Liang et al.(2021a)Liang, Cao, Sun, Zhang, Van Gool, and Timofte].

Structure The overall architecture of the SwinIR is given in Figure 21.

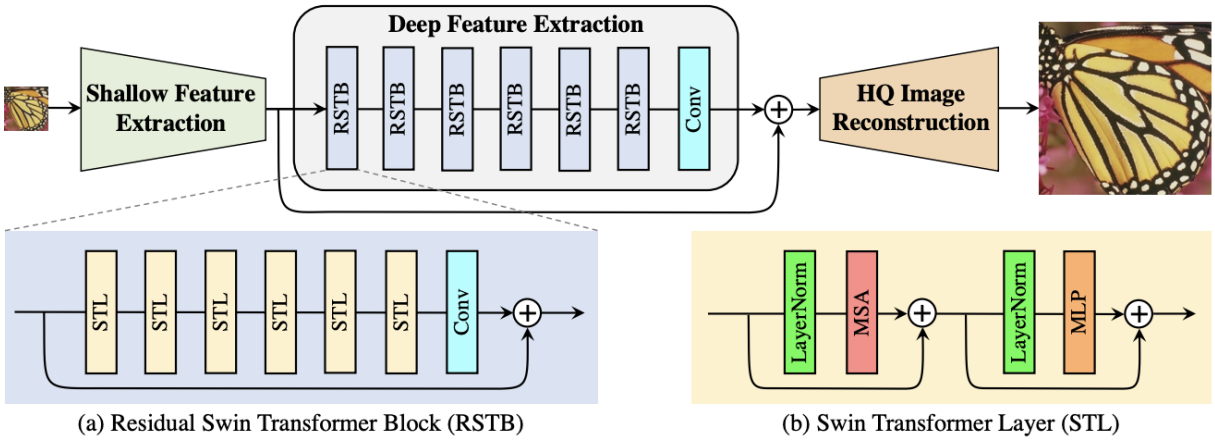


Figure 21: General structure of SwinIR

Shallow Feature Extraction This first part is simply composed of a 3×3 convolutional layer with C kernels, C being a hyperparameter of the algorithm. The input is of size $H \times W \times C_{in}$ (respectively height of image, width of image and number of input channels) and the output is of size $H \times W \times C$. As mentioned in [Xiao et al.(2021)Xiao, Dollar, Singh, Mintun, Darrell, and Girshick], the convolution layer is good at early visual processing, leading to more stable optimization and better results. It also provides a simple way to map the input image space to a higher dimensional feature space.

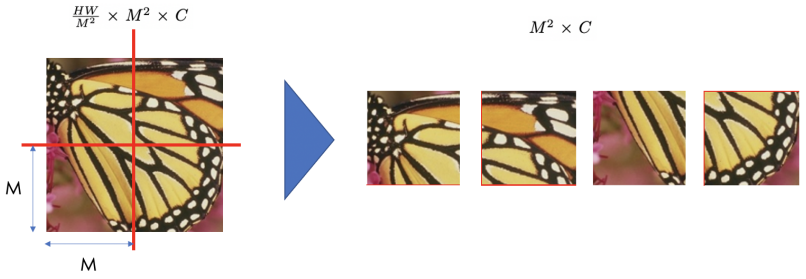
Deep Feature Extraction As we can observe on Figure 21, the deep feature extraction block is composed of multiple consecutive Residual Swin Transformer Blocks (RSTB) and ends with a 3x3 convolutional layer. This last convolutional layer brings the inductive bias of convolution into the Transformer-based network in order to enhance the future concatenation of shallow and deep features
[Liang et al.(2021b)Liang, Cao, Sun, Zhang, Van Gool, and Timofte]. Figure 21(a) shows that each RSTB is composed of multiple consecutive Swin Transformer Layer which structure is described on Figure 21(b). All the residual connections in the deep feature extraction block allow the aggregation of different levels of features from the different blocks to the reconstruction module
[Liang et al.(2021b)Liang, Cao, Sun, Zhang, Van Gool, and Timofte].

Swin Transformer Layer (STL) This layer is based on the Multi-head Self-Attention (MSA) of the Transformer Layer [Vaswani et al.(2017)Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin].

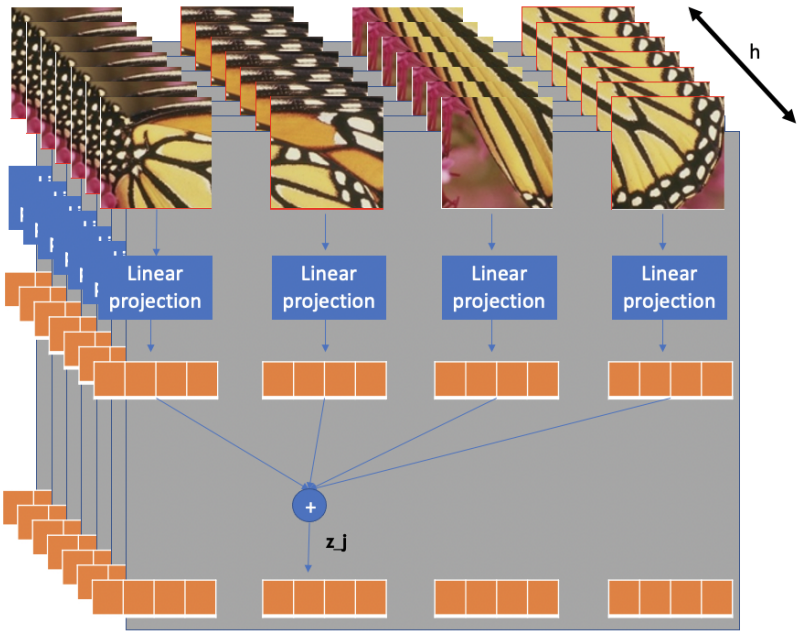
As shown in Fig11(a) Given an input of size $H \times W \times C$, Swin Transformer first reshapes the input to a $\frac{H \times W}{M^2} \times M^2 \times C$ feature by partitioning the input into distinct $M \times M$ local windows, where $\frac{H \times W}{M^2}$ is the total number of windows. Then for each of the h head that we see in Figure 22b, it computes the self-attention : for a local window feature $X \in \mathbb{R}^{M^2 \times C}$, the query, key and value matrices Q, K and V are computed as $Q = XP_Q, K = XP_K, V = XP_V$ where P_Q, P_K and P_V are projection matrices that are shared across the different windows for one head self-attention and we have $Q, K, V \in \mathbb{R}^{M^2 \times d}$
[Liang et al.(2021b)Liang, Cao, Sun, Zhang, Van Gool, and Timofte]. In Figure 22b we have

$$z_j = \sum_i softmax(\frac{Q_j K_i^T}{\sqrt{d}} + B)V_i \tag{11}$$

Each head has a relative position bias $B \in \mathbb{R}^{M^2 \times M^2}$. Results from all heads are then concatenated as mentionned in [Vaswani et al.(2017)Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin].



(a) Local window cutting



(b) Computations in Multi-head Self-Attention

Figure 22: Multi-head Self-Attention

Shifted window The input image being divided into fixed windows through the whole deep feature extraction block, the restored image may introduce border artifacts around each window as there is no connection between windows. While this problem can be alleviated by window overlapping, it would introduce extra computational burden. The shifted window principle is the specificity introduced in the SwinIR (Swin stands for Shifted-window) paper [Liang et al.(2021b)Liang, Cao, Sun, Zhang, Van Gool, and Timofte] that gives a solution to this problem. Regular and shifted window partitioning are used alternately in successive STLs to enable cross-window connections [Liu et al.(2021)Liu, Lin, Cao, Hu, Wei, Zhang, Lin, and Guo], where shifted window partitioning means shifting the feature by $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor)$ pixels before partitioning as it is illustrated in Figure 23.

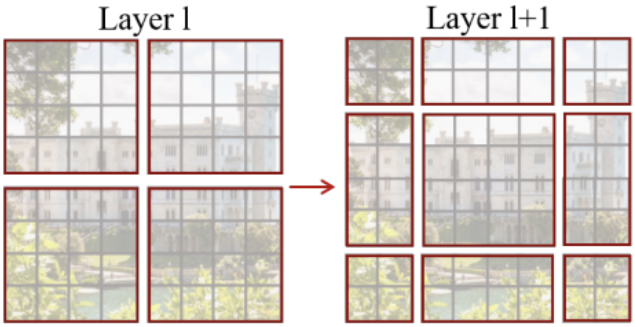


Figure 23: Shifted window principle

Cyclic shift allows to deal with different window sized induced by the shifted window mechanism. This transformation assumes that input images contain a certain periodicity. This technique is illustrated on Figure 24

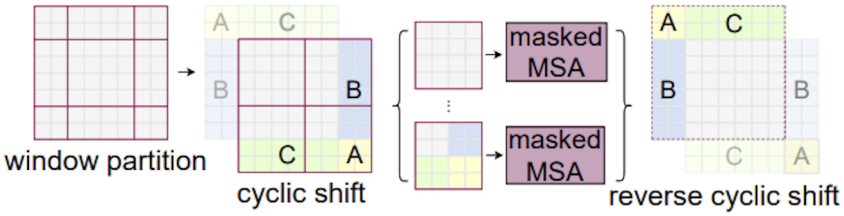


Figure 24: Cyclic shift padding

Next, following the MSA, the STL is composed of a multi-layer perceptron (MLP) that has two fully-connected layers with GELU non-linearity between. This is used for further feature transformations. The LayerNorm (LN) layer is added before both MSA and MLP, and the residual connection is employed for both modules.

HQ image reconstruction This block is composed of two layers : First, a 3×3 convolutional layer with the fixed number of output channel $C_{in} \times r \times r$. r is the upscaling factor. Then, the sub-pixel convolution layer[Shi et al.(2016)Shi, Caballero, Huszar, Totz, Aitken, Bishop, Rueckert, and Wang] is used to upsample the feature maps and get the HQ image. It is simply a rearrangement of pixels as illustrated in Figure 25.

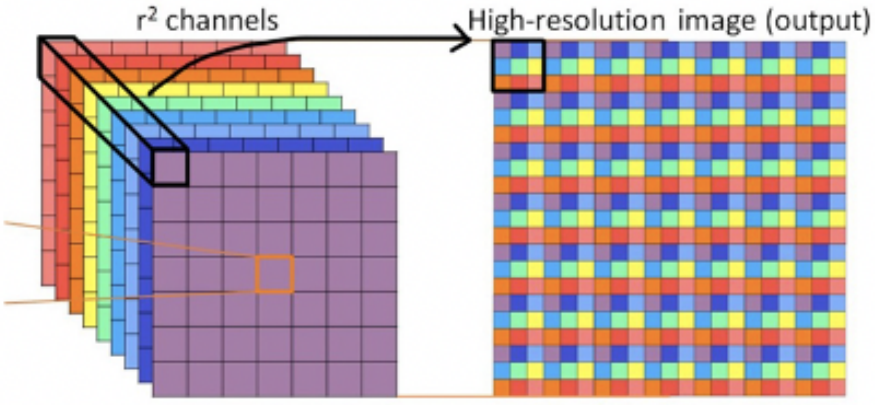


Figure 25: Sub-pixel convolution layer

4.2 SRCNN

In deep learning and especially in Computer Vision, CNN networks are usually used for classification tasks. However, the Super-Resolution Convolutional Neural Network (SRCNN) is a convolution-based network used

for Super-Resolution as its name implies. Its general structure is depicted in Figure 26.

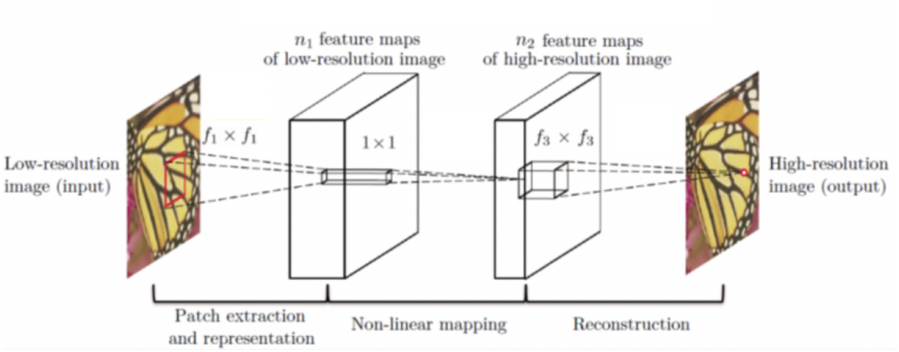


Figure 26: Super-Resolution Convolutional Neural Network (SRCN) structure

It is composed of 3 layers described below.

Patch Extraction and Representation First, the Low-resolution image is upscale to the size aimed for Super-Resolution using bicubic interpolation. Then, the obtained image X is given as input into a convolutional layer with RELU non-linearity. The output of this layer $F_1(X)$ is given by :

$$F_1(X) = max(0, W_1 * X + B_1) \tag{12}$$

- $W_1 \in \mathbb{R}^{c \times f1^2 \times n1}$ weight matrix
- $B_1 \in \mathbb{R}^{n1}$ bias vector
- c number of channels i image
- c number of channels i image
- $f1$ filter size
- $n1$ number of filters

Non-Linear Mapping With $Y = F_1(X)$, the output of this layer $F_2(F_1(X))$ is given by :

$$F_2(Y) = max(0, W_2 * Y + B_2) \tag{13}$$

- $W_2 \in \mathbb{R}^{n1 \times 1 \times 1 \times n2}$ weight matrix
- $B_2 \in \mathbb{R}^{n2}$ bias vector
- $n2$ number of filters

Usually we have $n1 > n2$, which can be interpreted as a Principal Component Analysis as the dimension of image remains a constant and the number of feature maps decreases.

Reconstruction This last layer's output is the high quality reconstructed image. This time there is no Non-Linearity applied, only a convolution and a bias :

$$F_{out}(Y) = W_3 * Y + B_3 \tag{14}$$

- $W_3 \in \mathbb{R}^{n2 \times f3 \times f3 \times c}$ weight matrix
- $B_3 \in \mathbb{R}^{c}$ bias vector

5 Method

5.1 Training

Preprocessing The preprocessing pipeline is summarized in figure 16 and explained in detail in part 3.4.

Optimizer We used the Adam optimizer. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

According to [Kingma and Ba(2014)], the method is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters".

We used the default learning rate which is 10^{-3}.

Regularisation To train our SwinIR we used the stochastic depth concept. This method resembles classic Dropout technique in which the key idea is that neurons are randomly dropped during training which prevents a neural network from over-fitting [Pham et al.(2019)Pham, Nguyen, Niehues, Müller, and Waibel]. For each mini-batch we randomly select sets of Swin Transformer Layers and remove their corresponding transformation functions, only keeping the identity skip connection. This is illustrated on Figure 27.

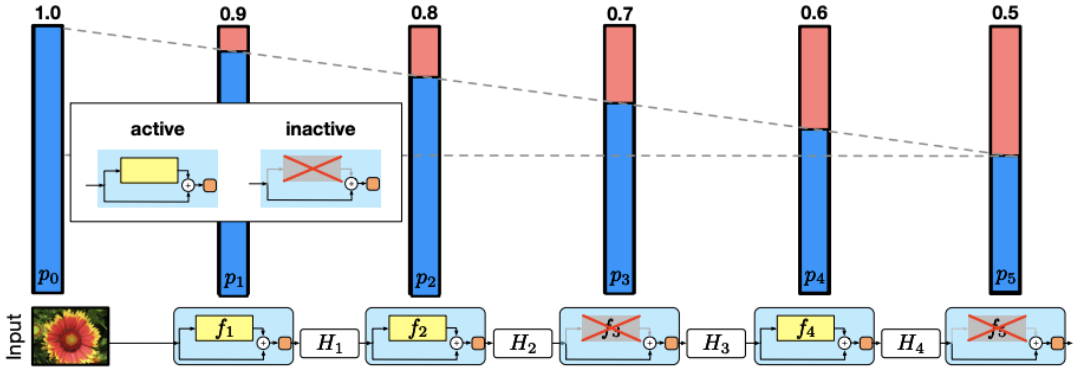


Figure 27: Stochastic depth illustration

Above on Figure 27, each bar corresponds to a Swin Transformer Layer and the blue proportion is equal to the probability for such a layer not to be skipped for each mini-batch. Indeed, the no-skip probability decreases through the network. It follows a linear decay rule from $p_0 = 1$ to p_L for the last STL block :

$$p_l = 1 - \frac{l}{L} \times (1 - p_L) \tag{15}$$

The linearly decaying rule originates from the intuition that the earlier layers extract low-level features that will be used by later layers and should therefore be more reliably present [Huang et al.(2016)Huang, Sun, Liu, Sedra, and We

We used $p_L = 0.9$ which is the default value of the SwinIR algorithm.

Hyperparameters For both SwinIR and SRCNN, the hyperparameters mentionned in part 4 Models are summarized in the following tables 1 and 2. We explicit here the values we used for our trainings.

For both models, we used an image size of 256×256 ($W = H = 256$).

Added to this, we have launched all our trainings with 50 epochs. We used a batch size of 1 or 2 given the RAM of GPUs we had access to.

Table 1: SwinIR hyperparameters

Name	Symbol	Value
Feature channel number	C	180
Window size	M	8
Attention head number	h	6
RSTB Number	N_{RSTB}	6
STL Number	N_{STL}	6

Table 2: SRCNN hyperparameters

Name	Symbol	Value
Number of feature maps after first layer	n_1	64
Number of feature maps after second layer	n_2	32
Kernel size of first layer	f_1	9
Kernel size of third layer	f_2	5

5.2 Loss

We have tried using the three following losses among our various trainings :

L1 loss

$$L_1 = \|I_{restored} - I_{HQ}\|_1 \tag{16}$$

where $I_{restored}$ is the pixels' intensities of the image restored by the algorithm and I_{HQ} of the target high quality image.

Something important to mention here is the fact that the Swin IR algorithm outputs the difference between target image and low resolution input image. Thus, if we note H_{SwinIR} the transfer function of the Swin IR algorithm, we have : $I_{restored} = H_{SwinIR}(I_{input}) + I_{input}$.

L2 loss

$$L_2 = \|I_{restored} - I_{HQ}\|_2 \tag{17}$$

Structural Similarity Loss (SSIM) This loss aims to measure the visual quality of an image compared to a target image. It is further detailed and its expression is given in part 6.1.2.

6 Results

6.1 Metrics

We have been using many different metrics to monitor the trainings. Each metric targets a specific aspect of the reconstruction of the image.

6.1.1 PSNR

The PSNR is a well-known metric widely used in signal-processing. The PSNR (Peak Signal to Noise Ratio) is a measure of distortion used in digital imaging, particularly in image compression. It is used to quantify the performance of encoders by measuring the quality of reconstruction of the compressed image compared to

the original image.
The PSNR is defined as followed :

$$PSNR = 10 * log_{10}(\frac{d^2}{MSE}) \tag{18}$$

Where MSE is the mean squared error between the 2 compared images and is defined as followed :

$$MSE = \frac{1}{mn}\sum_i \sum_j (I_1(i,j) - I_2(i,j))^2 \tag{19}$$

With m,n the width and height of the images. While PSNR is useful for measuring the proximity of the compressed image to the original at the signal level, it does not take into account the visual quality of reconstruction and cannot be considered an objective measure of the visual quality of an image.

Therefore, we decided to use the PSNR to have a quantitative measure of the reconstruction quality. We used the SSIM to define an objective measure of the visual quality of the image.

6.1.2 SSIM (Structural Similarity)

The SSIM was developed to measure the visual quality of a compressed image, relative to the original image. The idea of SSIM is to measure the structure similarity between the two images, rather than a pixelwise difference like PSNR does. The underlying assumption is that the human eye is more sensitive to changes in the structure of the image.

The SSIM metric is computed over several windows of an image. The metric between two windows x and y of size NxN is :

$$SSIM(x,y) = l(x,y).c(x,y).s(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_x\sigma_y + c_2)(cov_{xy} + c_3)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)(\sigma_x\sigma_y + c_3)} \tag{20}$$

with

- μ_x the average of x
- μ_y the average of y
- σ_x^2 the variance of x
- σ_y^2 the variance of y
- cov_{xy} the covariance of x and y
- L the dynamic range of pixel values, 1 in our case. We decided to shift the range of pixel values of every image between [0,1] when using the SSIM for ease of use.
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$, $c_3 = \frac{c_2}{2}$, $k_1 = 0.01$, $k_2 = 0.03$ by default.

We are using in our case a window size of 5x5 pixels.

6.1.3 Other Metrics

Reconstructed Image vs Target Image : We have monitored over all trainings the absolute difference between the reconstructed and target image to have a better idea of where the differences are. It also helped us to notice some trend with this or that loss when training with different losses. Some losses as the structural similarity are leaving behind some reconstruction artefacts at the beginning due to the way the loss is defined.

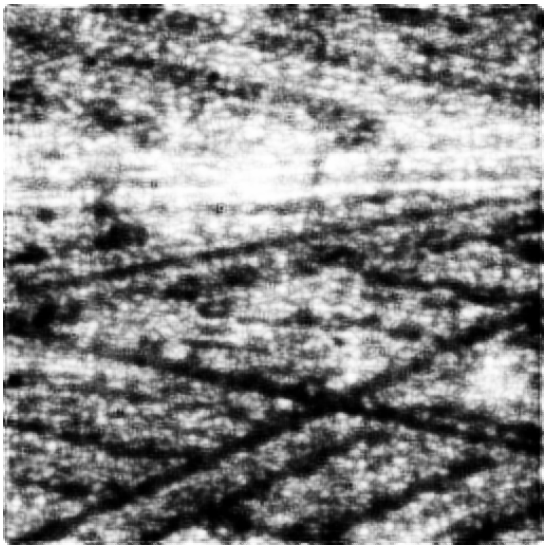
First vs Last : We have also monitored the absolute difference between the first reconstructed image and the last reconstructed image to see how much training over several epochs was useful.

Histograms : We decided to plot the distribution of pixels in a histogram for the reconstructed image and the target image. This enables to have a good overview of where the pixels values ought to fall into for the reconstructed image. We can indeed compare the distribution of the reconstructed and target image to ensure that the pixel values are consistent and that we are not being fooled by the visual representation of the images.

6.2 Results

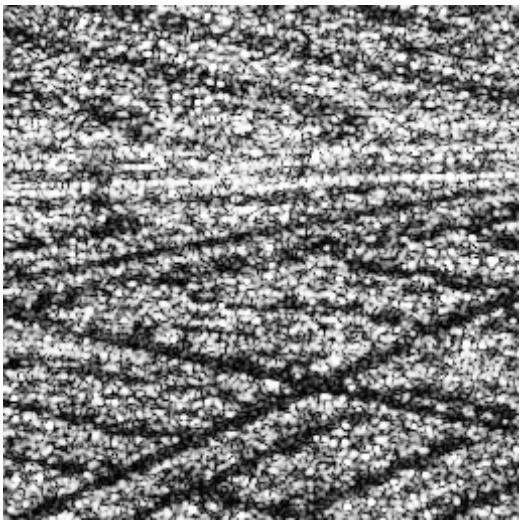
6.2.1 SwinTransformer

L1 Loss : We trained the SwinTransformer with a basi L1 pixelwise loss. We noticed that the reconstruction was possible but that many areas remained blurred and somehow smoothed. We also provided an example of a reconstruction in figure 28. The loss converged to 5.7db in the validation set



(a) Restored Image

(b) Target Image

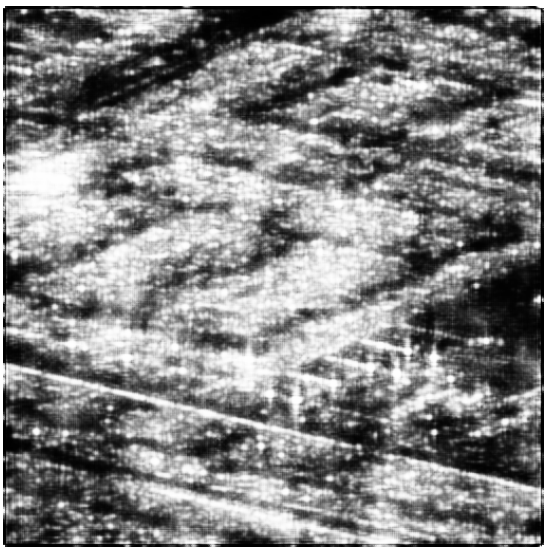


(c) Input Image

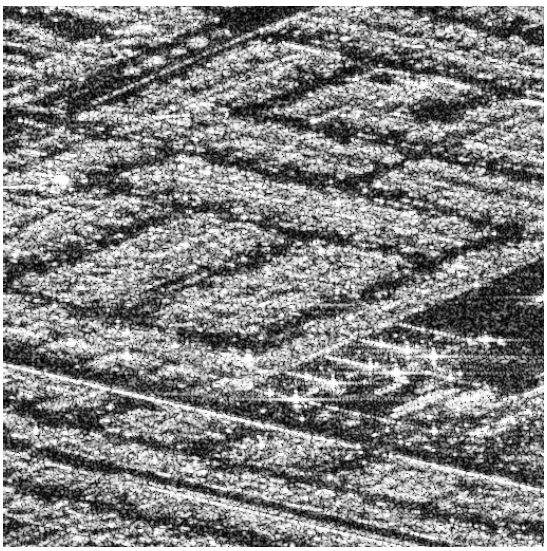
Figure 28: SWIN L1 loss Visualisation Results

L2 Loss : We trained the SwinTransformer with a mean squared error, also a pixelwise loss. The loss converges to around $50dB^2$ for the validation set which is quite big (difference of 7db betweel pixels) given the pixel range value (-200,0). We noticed that the model were behaving the same as for the L1 loss. We also provided an example of a reconstruction in figure 29.

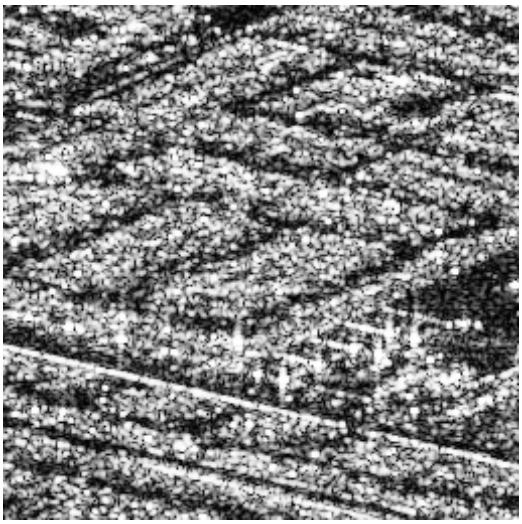
SSIM Loss : Finally, we trained the SwinTransformer optimising on the SSIM loss. We noticed that for the firsts epochs the images where harshly squared off because of the semantic of the SSIM loss. However, after



(a) Restored Image



(b) Target Image



(c) Input Image

Figure 29: SWIN L2 loss Visualisation Results

some epochs, the images becomes clearer than with the previous losses. Figure 30 shows the different metrics related to this training and figure 30 displays reconstructed images.

What can be emphasized as a general trend is that one epoch is usually more than enough to converge to a plateau for all the metrics. Indeed, it seems that the amount of data we have is quite big compare to the complexity of the task. There is also the possibility that we do not have enough variety in the training data, which leads to a poor generalisation and therefore a plateau for the metrics. If all data are similar, there is not much to learn after an epoch.

L1 loss + SSIM

The philosophy of the L1 loss si to make sure that the reconstructed image is close to the target image in terms of pixel values. However, it does not take into account the structure of the image, the high level features. To tackle this issue, the SSIM loss seems to be a good candidate. Thus, we trained a SwinTransformer with a combination of the two losses. We have :

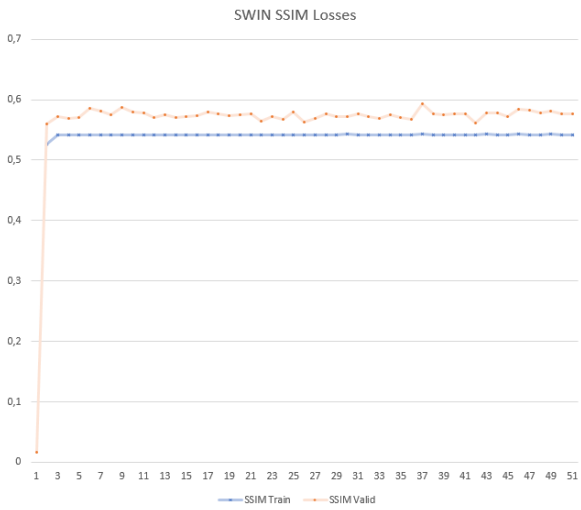
$$loss = L1 + \frac{\lambda}{SSIM} \tag{21}$$

This way we make sure that we maximize the SSIM loss by minimizing the reverse of the SSIM loss. We use the lambda scalar to scale the SSIM loss to the order of magnitudes of the L1 value. This way, we could

(a) Restored Image

(b) Target Image

(c) Input Image



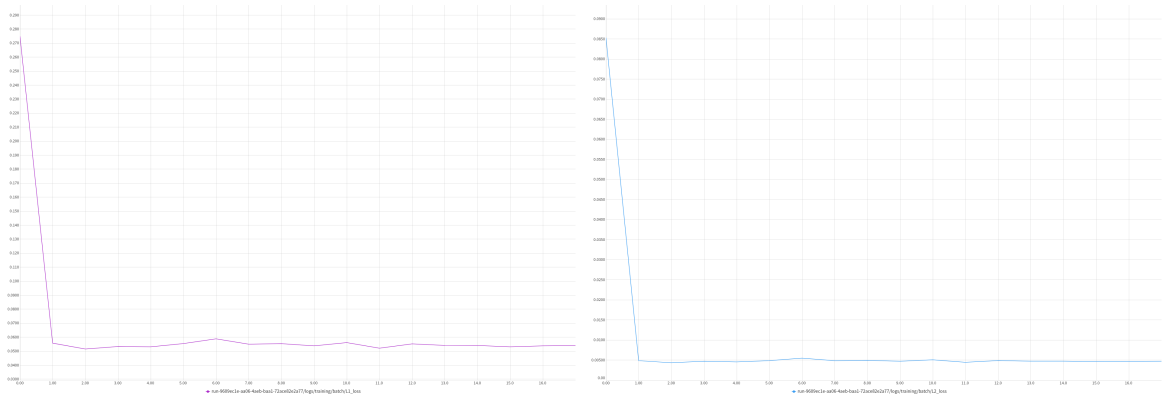
(d) SSIM Train and Valid

Figure 30: SWIN SSIM loss Visualisation results

theoritically both optimize a good reconstruction in terms of pixel values and structural similarity. The results of this experiment are shown in figure 32 and 31. We get an average PSNR = 23 at the end of the training.

6.2.2 SRCNN

SSIM Loss : To have a comparison with the state of the art SwinTransformer in Super Resolution, we training a simple SRCNN model optimising the SSIM Loss. We concluded that this model were good enough to overfit on the pixel values but lacked of structural reconstruction. The reconstructed images are indeed blurred (see figure 33) as if a low pass filter had been applied. This is a well known phenomenon for reconstruction with CNN. We can also notice that despite the small losses values, we still have a big difference between the target histogram and the restored histogram. This shows the limits of the SRCNN. This network cannot really generate good restored images.



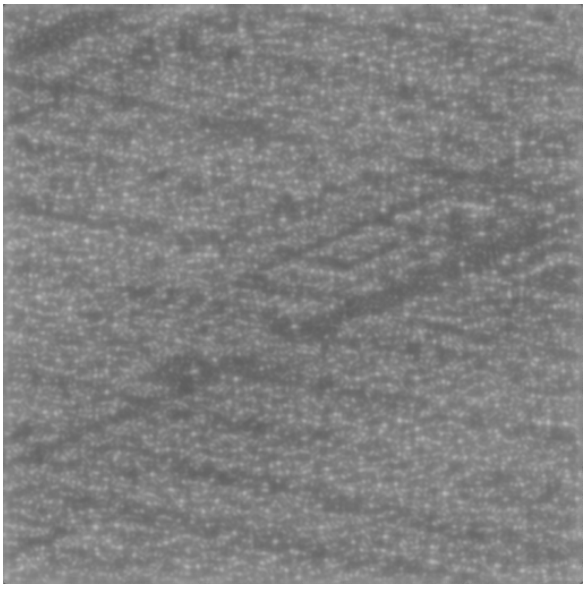
(a) L1 loss

(b) L2 loss

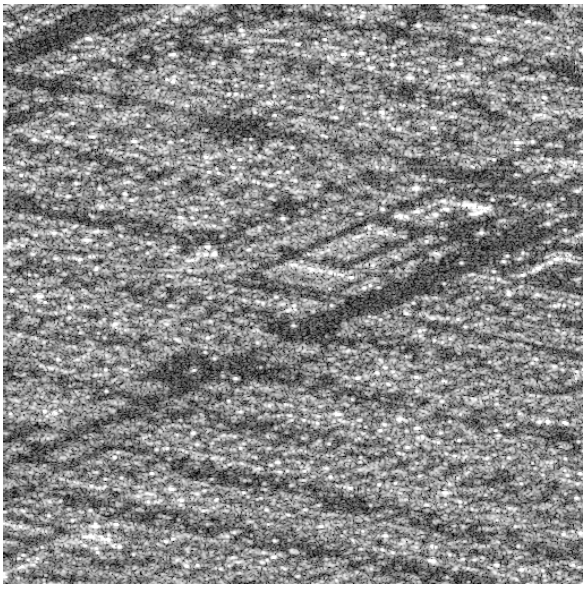
(c) L1+1/SSIM training loss

(d) SSIM valid loss

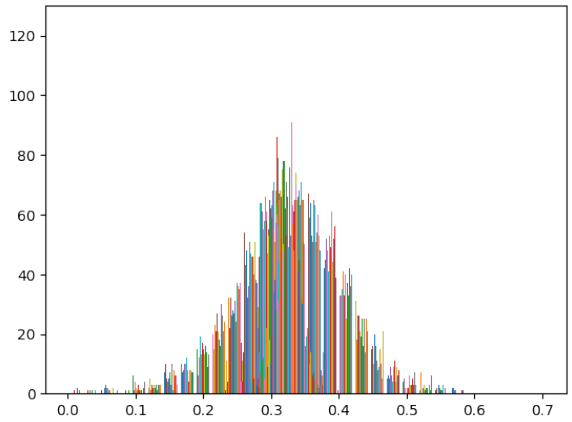
Figure 31: L1+SSIM SwinTransformer Scalar Results



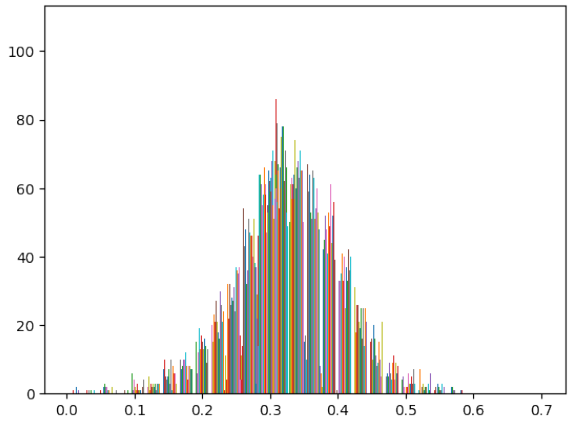
(a) Restored Image



(b) Target Image

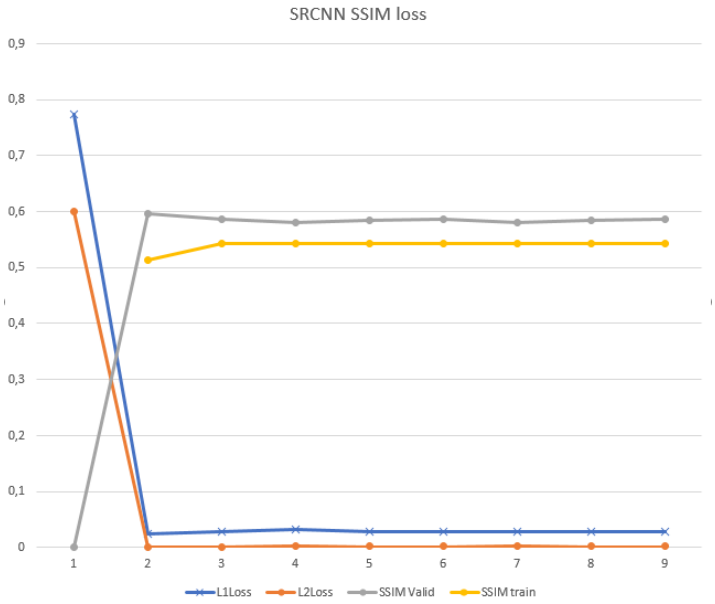


(c) Restored Histogram

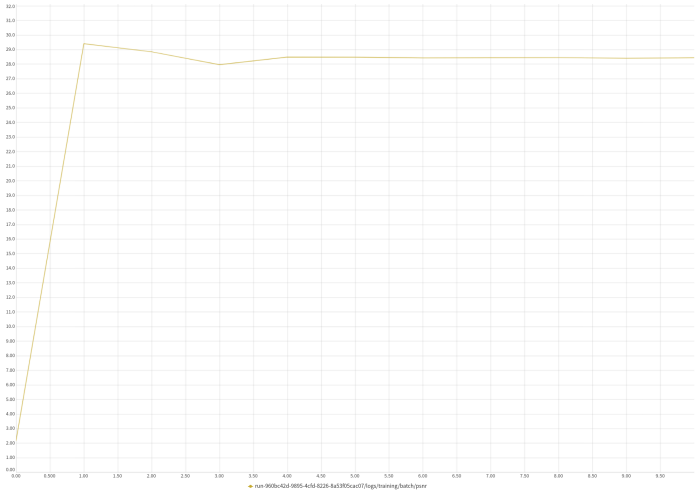


(d) Target Histogram

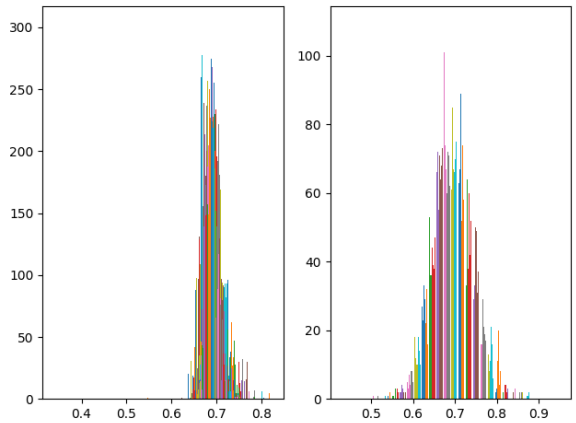
Figure 32: L1+SSIM SwinTransformer Visualisation results



(a) L1,L2,SSIM during training and validation



(b) PSNR



(c) Restored (left) and Target (right) Histogram

Figure 33: SRCNN SSIM Metrics

7 Conclusion

We have detailed a state of the art approach to perform super resolution. From degraded SAR images we could obtain, a certain reconstruction of the original SAR image. We have also investigated one of the state of the art model in computer vision in terms of backbone which is the vision transformer. Its performances combined with a precise loss can lead to a decent reconstruction provided that the loss is not a pixel wise loss. Thus, the structural similarity loss seems to be a good candidate for image reconstruction. Further improvement could be done in the way we downgrade the images, with different methods for differenr SLC slacks so that the NN does not learn the exact reverse transformation. Further work could be done regarding the model. Indeed, we did not explore the GANs.

8 Acknowledgements

We would like to thank our supervisors Jeremy Fix, Chengfang Ren and Israel Hinostroza for their constant help and their precious guidance throughout this project.

References

[Flores et al.(2019)Flores, Herndon, Thapa, and Cherrington] Africa Flores, K. Herndon, Rajesh Thapa, and Emil Cherrington. 2019. https://doi.org/10.25966/nr2c-s697 *The SAR Handbook: Comprehensive Methodologies for Forest Monitoring and Biomass Estimation.*

[Ghifary et al.(2016)Ghifary, Kleijn, Zhang, Balduzzi, and Li] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. 2016. Deep reconstruction-classification networks for unsupervised domain adaptation. *CoRR*.

[Huang et al.(2016)Huang, Sun, Liu, Sedra, and Weinberger] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. 2016. http://arxiv.org/abs/1603.09382 Deep networks with stochastic depth. *CoRR*, abs/1603.09382.

[Khandelwal()] Yash Khandelwal. https://www.analyticsvidhya.com/blog/2021/05/deep-learning-for-image-super-resolution/ Deep learningfor image super-resolution.

[Kingma and Ba(2014)] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

[Lai et al.(2017)Lai, Huang, Ahuja, and Yang] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. 2017. Deep laplacian pyramid networks for fast and accurate super-resolution. In *IEEE Conferene on Computer Vision and Pattern Recognition*.

[Ledig et al.(2016)Ledig, Theis, Huszar, Caballero, Aitken, Tejani, Totz, Wang, and Shi] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2016. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*.

[Liang et al.(2021a)Liang, Cao, Sun, Zhang, Van Gool, and Timofte] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. 2021a. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1833–1844.

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

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

[Liu et al.(2021)Liu, Lin, Cao, Hu, Wei, Zhang, Lin, and Guo] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022.

[Matcha()] Anil Chandra Naidu Matcha. https://blog.paperspace.com/image-super-resolution/ Image super-resolution: A comprehensive review.

[Pham et al.(2019)Pham, Nguyen, Niehues, Müller, and Waibel] Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Müller, and Alex Waibel. 2019. http://arxiv.org/abs/1904.13377 Very deep self-attention networks for end-to-end speech recognition. *CoRR*, abs/1904.13377.

[Ronneberger et al.(2015)Ronneberger, Fischer, and Brox] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597.

[Shi et al.(2016)Shi, Caballero, Huszar, Totz, Aitken, Bishop, Rueckert, and Wang] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883.

[Tai et al.(2017)Tai, Yang, and Liu] Ying Tai, Jian Yang, and Xiaoming Liu. 2017. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[Vaswani et al.(2017)Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

[Wang et al.(2019)Wang, Chen, and Hoi] Zhihao Wang, Jian Chen, and Steven C. H. Hoi. 2019. Deep learning for image super-resolution: A survey. *CoRR*, abs/1902.06068.

[Xiao et al.(2021)Xiao, Dollar, Singh, Mintun, Darrell, and Girshick] Tete Xiao, Piotr Dollar, Mannat Singh, Eric Mintun, Trevor Darrell, and Ross Girshick. 2021. Early convolutions help transformers see better. *Advances in Neural Information Processing Systems*, 34.

[Yifan et al.(2018)Yifan, Perazzi, McWilliams, Sorkine-Hornung, Sorkine-Hornung, and Schroers] W. Yifan, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O Sorkine-Hornung, and C. Schroers. 2018. A fully progressive approach to single-image super-resolution. In *CVPR Workshops*.