POLITECNICO DI MILANO

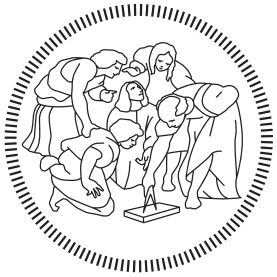
Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in

Computer Science and Engineering



SAR Image Despeckling from Unpaired Image-to-Image translation

Advisor: PROF. MATTEO MATTEUCCI

Co-advisor: PROF. FRANCESCO LATTARI

Master Graduation Thesis by:

VINCENZO SANTOMARCO

Student Id n. 914688

Academic Year 2019-2020

POLITECNICO DI MILANO

Facoltà di Ingegneria
Scuola di Ingegneria Industriale e dell'Informazione
Dipartimento di Elettronica, Informazione e Bioingegneria

Corso di Laurea Magistrale in
Computer Science and Engineering

SAR Image Despeckling from Unpaired Image-to-Image translation

Relatore: PROF. MATTEO MATTEUCCI

Correlatore: PROF. FRANCESCO LATTARI

Tesi di Laurea Magistrale di:

VINCENZO SANTOMARCO
Matricola n. 914688

Anno Accademico 2019-2020

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

This template has been adapted by Emanuele Mason, Andrea Cominola and Daniela Anghileri: *A template for master thesis at DEIB*, June 2015. This version of the paper has been later adapted by Marco Cannici, March 2018.

A papà

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Nowadays, a large number of satellites allows to systematically observe the surface of our planet from space. Among these, particularly interesting are satellites equipped with the Synthetic Aperture Radar (SAR) systems, which can capture high-quality images under different weather conditions and during nighttime, overcoming optical systems limitations. SAR satellites allow the collection of copious images by systematically revisiting the points composing their ground trace. This data is used to perform various tasks, e.g., earth and other planet surface monitoring, emergency response, military surveillance or the stability of civil infrastractures such as bridges or buildings. However, even if this technology does not suffer from optical sensors limitations, images acquired by this kind of satellites are contaminated by a peculiar noise called speckle. The speckle is caused by the interaction of out-of-phase waves with a target, and its reduction is necessary to interpret these images correctly. Thus, various algorithms have been proposed in the literature to reduce the speckle intensity.

In the last years, Deep Learning models demonstrated state-of-the-art performances over various computer vision tasks and, recently, they have been succesfully applied to SAR image despeckling. However, these models often rely on supervised learning, which requires a massive amount of labelled data to be successfully carried out. Although, there are no noise-free SAR images in reality, making it impossible to build labelled datasets composed of matching speckled and corresponding clean image pairs. Thus, the solutions proposed in the literature rely on synthetic datasets, built by sampling multiplicative SAR speckle from a known distribution, or on multi-temporal analysis. Even though these solutions reached outstanding performances over the despeckling task, the assumptions they make have some limitations. For example, synthetic datasets do not consider that the speckle distribution changes accordingly to the underlying scene, i.e., an urban area has a different speckle distribution than a wasteland. On the other side, the multi-temporal analysis uses a stack of images taken over the same scene at different times. Therefore, performing this analysis is possible only if the scene does not present significant changes over the observed period.

In this work of thesis, we propose a model trained in an unsupervised fashion without any simulated data or matching pair of clean and speckled images. Instead, we took inspiration from the CycleGAN [26] presented by Zhu et al. and developed a generative model capable of learning a speckle distribution correlated with the underlying scene. We trained side by side two networks capable of learning a despeckling and a speckle generation

function, mapping speckled SAR images into the domain of the clean ones and vice-versa. By doing so, our model is capable of generating its own pairs of images by sampling SAR speckle from the distribution associated with the clean one. Finally, we carried out a large set of experiments to validate the developed models.

SOMMARIO

Al giorno d'oggi, un gran numero di satelliti consente di osservare sistematicamente la superficie del nostro pianeta dallo spazio. Tra questi, particolarmente interessanti sono i satelliti dotati di sistemi SAR (Synthetic Aperture Radar), in grado di catturare immagini di alta qualità in diverse condizioni meteorologiche e durante le ore notturne, superando i limiti dei sistemi ottici. I satelliti SAR consentono la raccolta di numerose immagini rivisitando sistematicamente i punti che compongono la loro traccia al suolo. Questi dati vengono utilizzati per eseguire varie attività, ad esempio, monitoraggio della superficie della terra e di altri pianeti, risposta alle emergenze, sorveglianza militare o stabilità di infrastrutture civili come ponti o edifici. Tuttavia, anche se questa tecnologia non soffre dei limiti dei sensori ottici, le immagini acquisite da questo tipo di satelliti sono contaminate da un rumore particolare chiamato speckle. Lo speckle è causato dall'interazione di onde sfasate con un bersaglio e la sua riduzione è necessaria per interpretare correttamente queste immagini. Pertanto, in letteratura sono stati proposti vari algoritmi per ridurre l'intensità dello speckle.

Negli ultimi anni, i modelli di Deep Learning hanno dimostrato prestazioni all'avanguardia in vari compiti di visione artificiale e, recentemente, sono stati applicati con successo alla rimozione dello speckle di immagini SAR. Tuttavia, questi modelli spesso si basano sull'apprendimento supervisionato, che richiede un'enorme quantità di dati etichettati per essere eseguito con successo. Tuttavia, in realtà non ci sono immagini SAR prive di rumore, il che rende impossibile costruire set di dati etichettati composti da coppie di immagini pulite e corrispondenti corrotte dallo speckle. Pertanto, le soluzioni proposte in letteratura si basano su set di dati sintetici, costruiti campionando speckle moltiplicativo da una distribuzione nota, o su analisi multi-temporali. Anche se queste soluzioni hanno raggiunto prestazioni eccezionali nel compito di eliminazione dello speckle, le ipotesi che fanno hanno alcuni limiti. Ad esempio, i set di dati sintetici non considerano che la distribuzione dello speckle cambia di conseguenza alla scena sottostante, ovvero un'area urbana ha una distribuzione di speckle diversa rispetto a una terra desolata. Dall'altro lato, l'analisi multi-temporale utilizza una pila di immagini riprese sulla stessa scena in momenti diversi. Pertanto, l'esecuzione di questa analisi è possibile solo se la scena non presenta cambiamenti significativi nel periodo osservato.

In questo lavoro di tesi, proponiamo un modello addestrato in modo non supervisionato senza dati simulati o coppie di immagini pulite e macchiate. Invece, ci siamo ispirati al CycleGAN [26] presentato da Zhu et al. e abbiamo

sviluppato un modello generativo in grado di apprendere una distribuzione speckle correlata con la scena sottostante. Abbiamo addestrato fianco a fianco due reti in grado di apprendere una funzione di despeckling e di generazione di speckle, mappando immagini SAR corrotte dallo speckle nel dominio di quelle pulite e viceversa. In questo modo, il nostro modello è in grado di generare le proprie coppie di immagini campionando lo speckle SAR dalla distribuzione associata all'immagine pulita. Infine, abbiamo condotto un'ampia serie di esperimenti per convalidare i modelli sviluppati.

1 INTRODUCTION

Nowadays, numerous satellites observe the Earth surface, making it possible to collect data of dangerous or inaccessible areas from a wider perspective than what can be achieved at ground level. Taking optical images from outside the atmosphere, though, could be a challenging task. For example, many clouds and various weather phenomena can interfere with the light reflected by the surface of our planet, or there could be no light at all if the image is taken during nighttime. In this context, Synthetic Aperture Radars (SAR) propose a solution to overcome the limitations of optical sensors. A SAR is an imaging radar mounted on a moving platform capable of sending electromagnetic waves and collecting the echoes of those waves, storing all the related information for future processing. According to the time a SAR wave takes to travel forth and back, it is possible to map the surface hit by this wave. This technology is capable of high-resolution remote sensing, independent of flight altitude and weather, as SAR can select frequencies to avoid weather-caused signal attenuation. Also, since the SAR itself provides the electromagnetic wave, no sunlight is needed to capture an image. SAR technology is applied in numerous tasks, like land cover classification, environmental monitoring, emergency response, and military surveillance.

Even though SAR is a powerful instrument, it suffers from a peculiar grainy noise effect, called speckle. The speckle is caused by the resolution of each cell being associated with an extended target, containing several scattering centres whose elementary returns, by positive or negative interference, originate light or dark image brightness. This phenomenon gives the image a grainy look, making it challenging to identify the main features of the surface imaged by the SAR. Thus, the need to find despeckling algorithms for reducing speckle intensity while preserving information needed for the tasks that take advantage of this technology.

In the literature, various approaches have been introduced to reduce the speckle of SAR images, from classical algorithms to modern Deep Learning (DL) using Convolutional Neural Networks (CNN), which have been carefully discussed later in the document. DL largely demonstrated to be a game-changer in numerous tasks in the field of computer vision, like image classification and segmentation. In the last years, DL approaches have also been successfully investigated in the Earth Observation field. However, these approaches require many labelled data to work, like a matching pair of image and class it belongs to, or a mask associating a class for each pixel of the associated image. Unfortunately, in SAR image despeckling, no ground truth

is available due to the impossibility of gathering speckle-free images. Approaches from literature tackled this problem by building synthetic datasets sampling multiplicative speckle realizations from a known distribution or by performing multi-temporal analysis over static scenes. Although synthetic datasets allow learning single-image processing algorithms that outperformed classical approaches, the assumptions made over the speckle distribution do not consider how different surface features exhibit different scattering characteristics. For example, the speckle showed over urban areas presents a stronger backscatter than smoother surface due to artificial structures. At the same time, performing multi-temporal analysis over a stack of SAR images is limited for those scenes that do not differ significantly over the time in consideration. In late years, generative models such as Generative Adversarial Networks (GANs) showed outstanding performances over various computer vision tasks. Also, in SAR literature, various approaches have exploited the GAN capabilities to learn speckle reduction functions. In this work of thesis, we propose a solution belonging to this class of generative models, paying particular attention to solving the scarcity of labelled SAR data. We took inspiration from unpaired image-to-image translation models, especially that proposed by Zhu et al. [26], and trained our model without any pair of SAR image and corresponding speckle-free realization. Due to the lack of paired speckled and speckle-free images, we designed and trained a Speckle Generator network capable of mapping a clean image into a speckle corrupted one. We exploited the CycleGAN model of Zhu et al. to concurrently train a Despeckler network that can take advantage of the images generated by the Speckle Generator for learning a despeckling function. We did this without making any assumption on the speckle distribution or using any target image obtained through a multi-temporal analysis. Instead, we designed and trained those networks to learn an image-to-image translation model, to map a SAR image into the clean domain and vice-versa.

In this document, we present all the phases of our exploration that led to our final solution. First, we started from the more straightforward task of generating SAR speckle for corrupting clean images and then moved to a CycleGAN-like model trained for learning a one-to-one mapping from SAR images to the corresponding clean one and vice-versa. Finally, we empowered the Speckle Generator taking inspiration from the CycleCVAE model of [31] making it learn the one-to-many mapping from the clean domain to the SAR one. The proposed final solution consists of a Despeckler showing performance comparable with other state-of-the-art approaches, preserving edges and details over both edges and details over both smooth and heterogeneous areas. Furthermore, the training framework allowed us to obtain a good speckled images generator from the multi-temporal Means extracted using [9]. The Speckle Generator has been modelled using the CVAE [30] architecture allowing the generation of multiple samples of speckle,

given the same underlying scene, conditioned over a latent vector, giving the possibility to generate realistic data for building a synthetic dataset. The proposed solution proves how unpaired image-to-image translation models can be successfully used for SAR image despeckling, as shown during the experiment conducted, and provides a good baseline for future developments.

1.1 OUTLINE

The rest of the document is organized as follows:

- Chapter 2: introduces the main deep learning models and concepts needed to understand the work done, from the Generative Adversarial Networks (GANs) to the Conditional Variational Auto Encoders (CVAE) and their applications for solving one-to-many mappings in CycleGANs.
- Chapter 3: its first part describes in detail the distribution of the speckle of a SAR image. The rest of the chapter is structured to give an overview of the state-of-the-art approaches by analyzing both classical algorithms and Deep Learning approaches.
- Chapter 4: gives a detailed description of the proposed method, focusing on the Despeckler and Speckle Generator architectures and the training process.
- Chapter 5: describes the main experiments which summarize the various steps leading to the proposed solution, analyzing the results of every intermediate model and justifying every subsequent improvement.
- Chapter 6: summarizes our work by retracing the main stages and the solved problems. Finally, some future developments are proposed.

2

BACKGROUND

In this chapter, we describe we describe the theoretical background useful to understand the proposed work. In particular, we give an overview on Generative Adversarial Networks and Variational Auto Encoders, on which this work of thesis focuses.

2.1 GENERATIVE ADVERSARIAL NETWORKS

The base model we decide to start our analysis from is proposed by Goodfellow et al. In [21], the authors propose a model for generating images starting with a random vector z. The idea is to find a Generator network G that is capable of learning the mapping $f : Z \to X$, where Z is a known distribution, e.g., a standard Gaussian, and X the actual domain of the images we aim at generating whose distribution is $p(x)$. Thus, an optimal G should be able to produce images $\hat{x} = G(z),\ z \sim N(0, 1)$ whose distribution $p(\hat{x})$ coincides with the actual one $p(x)$. For doing so, a Discriminator network D is included during the training. The Discriminator network is trained in order to learn to recognize whether an input image comes from $p(x)$ or from the generated distribution $\hat{p(x)}$ so that when G is optimally trained the generated images $\hat{x}$ are indistinguishable from the real ones $x \sim p(x)$. The two networks, D and G, are then trained concurrently to make D learn how to classify real and fake images and G to trick D and maximise its error. For doing so, those networks are trained using an *adversarial loss*:

$$\mathcal{L}_{gan} = \max_{G} \min_{D} \mathbb{E}_{x \sim p(x)}[\log D(x)] + \mathbb{E}_{z \sim N(0,1)}[\log(1 - D(G(z)))] \,. \quad (2.1)$$

These networks are known as `Generative Adversarial Networks` (GANs). They are proven to be extremely powerful and showed excellent performance in various computer vision tasks. Still, their training is very unstable, and those models suffer from various problems during training. Discriminator networks have Sigmoid or Tanh activation functions on the last layer. When the Discriminator is optimal, the output values of those functions are close to +1 and -1 or 0, for Tanh and Sigmoid respectively, where the gradient is almost zero. This leads to a vanishing gradient problem, making the training of the Generator network impossible. An optimal discriminator does not provide enough information for the Generator to make progress. Another common failure of GANs is the *mode collapse*. The mode collapse happens when a Generator learns how to produce a single or a small set of outputs

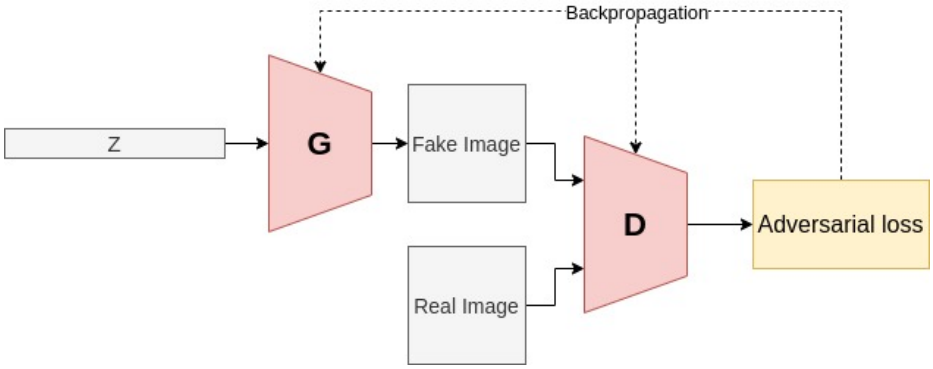


Figure 2.1: Schema of a GAN training. The Generator (G) is fed with a random vector z, sampled from a standard Gaussian distribution, and outputs a generated image. The Discriminator (D) is fed with both image extracted from the training dataset and those generated by G, assign each input a class between "real" and "fake". On the output of the Discriminator is computed the adversarial loss, which D aims at minimizing, while G, whose objective is to fool D, at maximizing.

that are the most plausible for the Discriminator independently from the input vector z. Also, finding the right point to stop the training is difficult. When G is optimal, the output of the network D is random; thus, if the training continues past this point, the generator starts to train on this random feedback, and its own quality may collapse. The training process of a GAN is schematized in Figure 2.1.

2.2 WASSERSTEIN GENERATIVE ADVERSARIAL NETWORKS

Arjovsky, Chintala, and Bottou propose a solution to the limitations showed by the GANs. In [23], the authors propose to use Critic networks instead of Discriminators. Critic networks do not have any Sigmoid or Tanh activation function as output, but instead their output is a linear function whose co-domain is $\mathbb{R}$ and gradient 1. Given the distribution $\mathbb{P}_x$ of real data and P_θ of generated data, authors propose to train a generative model minimizing the *Earth Moving* (EM) distance, or Wasserstein-1

$$\mathcal{W}(\mathbb{P}_x, \mathbb{P}_\theta) = \inf_{\gamma \in \Pi(\mathbb{P}_x, \mathbb{P}_\theta)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||] \,,$$

where $\Pi(\mathbb{P}_x, \mathbb{P}_\theta)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_x$ and $\mathbb{P}_\theta$. Thanks to the Kantorovich-Rubinstein duality [24], the previous becomes

$$\mathcal{W}(\mathbb{P}_x, \mathbb{P}_\theta) = \sup_{||f||_L \leqslant 1} \mathbb{E}_{x\sim\mathbb{P}_x}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

where the supremum is over all the 1-Lipschitz functions $f : X \to \mathbb{R}$. In this paper, the authors propose approximating the function f with a neural

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Figure 2.2: Training algorithm for WGANs [23]

network, the Critic network, referred to as D for continuity with the GAN model. This model is called `Wasserstein GAN` (WGAN) due to the EM distance, or Wasserstein-1, used for training those networks. The adversarial loss for WGANs is

$$\mathcal{L}_{wgan} = \max_{G} \min_{D} \mathbb{E}_{z\sim p(z)}[D(G(z))] - \mathbb{E}_{x\sim p(x)}[D(x)] \tag{2.2}$$

For forcing the Lipschitz constraint over these functions, authors propose to clip the weights of the Critic in a fixed range. Because of the nature and gradient of the Critic, it has been proven that this network has better training stability. Thus, the authors propose a training algorithm where the weights of G are updated only once every n steps on D, always to have an optimal Critic. The algorithm proposed in [23] for training this model is showed in Figure 2.2.

In [21], authors point out how the weight clipping is not an optimal solution to force the 1-Lipschitz constraint over the Critic network. In [25], Gulrajani et al. analyze how the choice of the clipping limit is crucial for a successful training of WGANs, that can lead to both vanishing or exploding gradients. Also, those critics are only capable of learning simple functions. In this paper, the authors also propose an improved training for WGANs. The solution proposed in [25] is to add a *gradient penalty* loss term for penalizing the gradient norm of the Critic networks to make them have it as close as possible to 1. The WGAN-GP loss then becomes

$$\mathcal{L} = \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\sim\mathbb{P}_r}[D(x)] + \lambda_{gp}\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}}D(\hat{x})\|_2 - 1)^2] \,. \tag{2.3}$$

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.
while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

Figure 2.3: Training algorithm for WGAN-GPs [25].

The authors define $\mathbb{P}_{\hat{x}}$ sampling uniformly along straight lines between pairs of points sampled from the data distribution $\mathbb{P}_r$ and the generator distribution $\mathbb{P}_g$. After various experiments over different tasks, they also define $\lambda_{gp} = 10$ that resulted in the optimal weight for the Gradient Penalty term. The improved training for the WGAN-GP is shown in Figure 2.3.

2.3 CYCLE CONSISTENT GENERATIVE ADVERSARIAL NETWORKS

The improved training for the WGAN-GP is shown in Figure 2.3. In this work of thesis, we will take as reference for the unpaired image-to-image translation task the `CycleGAN` model proposed by Zhu et al. in [26]. In this paper, the authors aim at mapping images from a given domain A to another one B, and vice-versa, without any given matching pair of images. The authors suggest using two GANs for learning the two mapping functions $f : A \rightarrow B$ and $g : B \rightarrow A$, G_B and G_A respectively. For training, they use two Discriminator networks, D_B and D_A, and two adversarial losses [21], for making the two generators learn the f and g mappings and produce outputs that are indistinguishable from images taken from the two domains. The adversarial loss for the images produced by the g loop is:

$$\mathcal{L}_{gan,A} = \mathbb{E}_{a \sim p(A)}[\log D_A(a)] + \mathbb{E}_{b \sim p(B)}[\log(1 - D_A(G_A(b))] \; ;$$

a similar loss is introduced for those produced by f. The discriminators aim at minimizing those losses, while the generators at maximizing them. For forcing the networks to learn a meaningful mapping, i.e., given $a \in A$

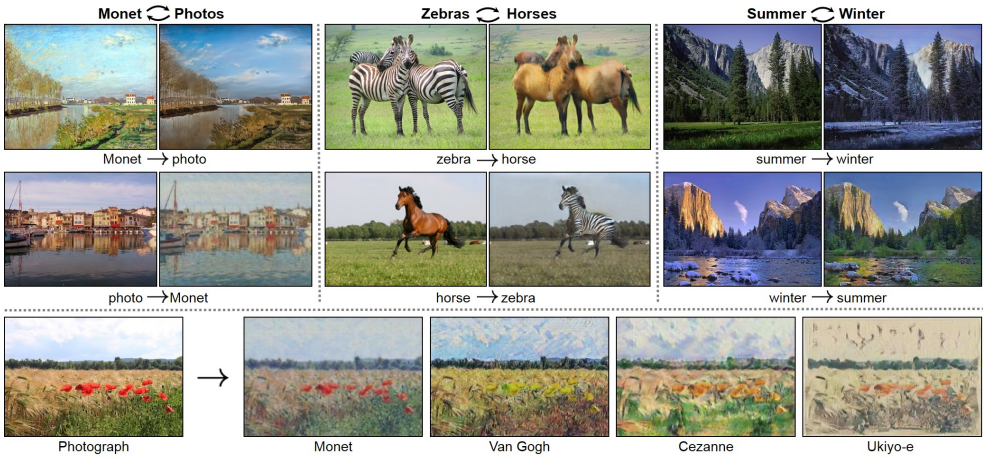


Figure 2.4: Example of tasks performed by a CycleGAN [26]. The top row shows, in order, examples of mapping between optical pictures and paintings domain, transforming zebras to horses and vice-versa and mapping images from one season to another. The second row shows example of mapping between pictures and paintings domain, showing how this model can learn how to translate a picture into a painting with the respective style.

produce the corresponding $b \in B$ and vice-versa, the networks are made cycle consistent, i.e., $f(g(b)) = b$ and $g(f(a)) = a$. The authors do this by adding a *cycle consistency* term to the two adversarial losses:

$$\mathcal{L}_{cycle} = \mathbb{E}_{a\sim p(A)}[\|G_A(G_B(a)) - a\|_1] + \mathbb{E}_{b\sim p(B)}[\|G_B(G_A(b)) - b\|_1] \,.$$

The final loss used for training the whole model is:

$$\mathcal{L}_{cycleGAN} = \mathcal{L}_{gan,A} + \mathcal{L}_{gan,B} + \lambda_{cycle}\mathcal{L}_{cycle} \,, \tag{2.4}$$

where λ_{cycle} is the weight of the cycle consistency term that authors set to be 10. The authors demonstrate the superiority of their approach from previous methods, also on tasks where paired training data does not exist. Figure 2.4 shows some examples of tasks that can be carried out with a CycleGAN. This Figure shows, on the top row, from left to right, how the network can learn how to translate landscape pictures into Monet paintings, zebras into horses, summer into winter and vice-versa. On the bottom row is showed how this model can learn, given a collection of painting of famous artists, how to render a picture into the respective style.

2.4 MANY-TO-ONE MAPPINGS

Using a simple cycle-consistent GAN may restrict the learning of a function where there is a many-to-one mapping. The cycle loss aims at minimizing the

point-wise distance between the original x and the reconstruction f(g(x)), and so implicitly is supposing a bijective mapping between the two domains X and Y. The problems that may arise in case CycleGANs are used for learning such non-bijective mappings are discussed by Bashkirova, Usman, and Saenko in [27].

In this paper, the authors analyze the one-to-many mapping problem in image-to-image translation. In such a scenario, a cycle-consistent model is not optimal for learning the function for mapping images from the lower dimensionality domain into one of the many correspondings. They show how the model learns how to hide information useful to reconstruct the input image, sampled from the higher dimensionality domain, in the generated images. Suppose the mappings $f : X \to Y$ and $g : Y \to X$, where for every $y \in Y$ there is only one corresponding $x \in X$ while for every $x \in X$ there are many $y \in Y$ corresponding images. The cycle loss introduced in [26] aims at minimizing $\|y - f(g(y))\|_1$, thus given a generated $\tilde{x} = g(y)$ the network used for approximate g must learn how to reconstruct the one y taken as input at the beginning of the cycle among the many. To accomplish this task better, the network approximating the function f may learn to hide some useful information for reconstructing the original y in a low amplitude noise that cannot be seen by human eyes, which is low enough to be ignored by the Discriminator. Those models are then susceptible to introducing low amplitude noise to the input that can destroy the hidden signal. In this paper, the authors also present two possible self-defences. They propose to include a low amplitude Gaussian noise before performing the reconstruction

$$\|y - f(g(y) + k\tilde{n})\|_1 \;\; \tilde{n} \sim N(0, 1), \;\; k \,\text{amplitude}$$

for hiding any low amplitude noise carrying information encoded by the network f and forcing g to ignore those signals. Another proposed solution is to add a Guess Discriminator to distinguish between actual input image y and reconstructed ones $f(g(y))$ to identify the hidden information stored on the generated one. Thus the network g must then learn how to produce authentic-looking reconstructions indistinguishable from the sampled ones with no low amplitude noise.

In our case, we want to find a way to prevent this self-attack from happening and be able to learn a function that can identify the many corresponding images, given one from the lower dimensionality domain, and lets us sample from this distribution. The idea is to store information necessary to perform the operation of sampling a certain image among those in a vector. This way, it is possible to use this vector to condition the one-to-many network and generate different images given the same input. There are in literature various approaches aiming at resolving this problem but we will focus on two of them in particular: one proposed by Guo et al. [31], that lays on the

idea introduced by the *Variational Auto Encoders* (VAE) [29], and one by Li et al. [32].

2.5 VARIATIONAL AUTO ENCODER

In [29], Kingma and Welling propose a method for training a network for generating data belonging to an unknown distribution $p(x)$ given a set X of known $x \sim p(x)$. The idea is to approximate a function $f(z)$ such that $\hat{x} = f(z) \sim p(x)$, where z is a random variable sampled from a known distribution, e.g., $z \sim N(0, 1)$. Training such network means finding the function f_θ, parameterized over θ, that maximizes the probability of generating data in X

$$p(x) = \int p_\theta(x|z)p(z)dz \, ,$$

where $p_\theta(x|z)$ is the probability distribution of the data generated by f_theta. Here $p_\theta(x|z)$ is modeled as a Gaussian, i.e., $p_\theta(x|z) = N(x|f_\theta(z), \sigma^2)$, so that there is a set of z that can generate something similar to each $x \in X$ and we can use gradient descent to train f_θ and gradually increase $P(x \in X)$. For training f_θ, may be possible to sample n $z \sim N(0, 1)$ for each $x \in X$ and then maximize

$$p(x) = \frac{1}{n} \sum_i p(x|z) \, .$$

Since $p(x|z)$ is an isotropic Gaussian, the negative log probability of x is proportional to the squared Euclidean distance between $f_\theta(z)$ and x. Since almost every z has $p(x|z)$ almost zero, performing a train like this requires sampling an extremely high number of z vectors. To overcome this, VAEs aim to improve the sampling strategy, restricting possible z values to those that will likely produce x. We want then to add a new network $q_\theta(x)$ trained for producing a distribution $p(z|x)$ of those z values that maximizes the probability of generating a given x. If we model the distributions produced by q as $N(\mu_q, \sigma_q^2)$, we can train the whole model by sampling $x \in X$ and $z \sim q_\theta(x)$ and maximizing

$$\log(p(x|z) - \mathbb{KL}[p(z|x)||p(z)] \, ,$$

where $\mathbb{KL}$ represents the Kullback-Leibler (KL) divergence. Considering $p(x|z) = N(f(z), \sigma^2)$, $p(z|x) = q_\theta(x) = N(\mu_q, \sigma_q^2)$ and $p(z) = N(0, 1)$, Kingma and Welling prove that we can train a VAE by minimizing the objective

$$\mathcal{L}_{vae} = \|f_\theta(x) - x\|_2^2 + 0.5(\sigma_q + \mu_q^2 - 1 - \log(\sigma_q)) \, . \tag{2.5}$$

Performing a gradient descent over this loss function may be impossible due to the impractical gradient descent over the operation of sampling z. Thus the authors introduce the *reparameterization trick*. The reparameterization trick

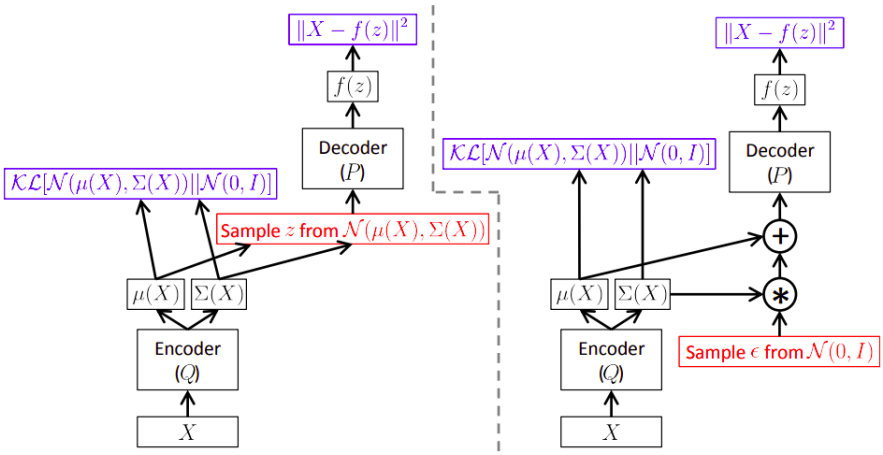


Figure 2.5: VAE without reparameterization trick (left) and with it (right) [28]. The part in red represents the sampling operation through which the gradient operation cannot be performed.

is how they allow the propagation of the gradient through the sampling of z operation by sampling a vector $\epsilon \sim N(0, 1)$ and then element-wise multiplying and adding this vector with the output of the network $q_\theta(x) = \mu_q, \sigma_q$:

$$z = \mu_q + \sigma_q \odot \epsilon \, .$$

The reparameterization trick and why it is needed is schematized in Figure 2.5. The VAE is so-called because of the architecture composed by an encoder for generating z distribution and a decoder for generating the output, that reminds the classical *Auto Encoders*. When this model is optimally trained we expect that the distribution $p(z|x) = N(0, 1)$, thus at inference time we can sample $z \sim N(0, 1)$ and feed the generator with it, producing a $x \sim p(x)$.

2.6 CONDITIONAL VARIATIONAL AUTO ENCODERS

Sohn, Lee, and Yan propose in [30] a method for applying the VAE architecture to the one-to-many mapping. The idea is to create a model capable of learning a function $f_\theta(x, z)$ that, given an input x, samples one of the many corresponding y of the codomain associated with x. In our SAR despeckling problem, this is to sample one of the possible speckled images y from the distribution whose underlying scene is x. The architecture proposed is called *Conditional VAE* (CVAE) [30]. Like VAEs, CVAEs use an encoder to obtain a z distribution that is the one that maximizes the probability of generating a given y and then feed an input x together with the z to a Generator network that should produce y. Thus, we want to produce an output y that is conditioned by z over the input x. Sohn, Lee, and Yan present an encoder that

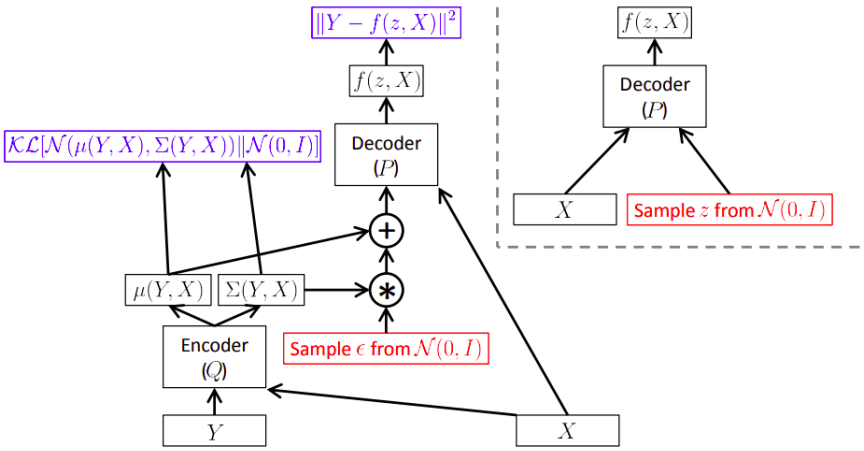


Figure 2.6: CVAE architecture at training (left) and inference (right) time [28].

takes as input both the input x and the desired output y and produces a distribution from which a z is sampled using the reparameterization trick, then the decoder is fed with x and z to produce something that should be as similar as possible to the original y. Similarly to how is introduced in [29], during training, given x and a corresponding y, we can minimize the loss

$$\mathcal{L}_{cvae} = \|f_\theta(x, z) - y\|_2^2 + 0.5(\sigma_q + \mu_q^2 - 1 - \log(\sigma_q)) \ ,$$

$$\mu_q, \sigma_q = q(x, y) \qquad \epsilon \sim N(0, 1) \qquad z = \mu_q + \sigma_q \odot \epsilon \ .$$

After an optimal training, we can expect that $q(x, y) = N(0, 1)$, so at inference time, given an x, we can sample $z \sim N(0, 1)$ and use this to condition the generation of $y = f_\theta(x, z)$. The process of training and inference of a CVAE is schematized in Figure 2.6. Training this model requires labelled data such as a dataset of corresponding pairs of input x and expected output y. Thus we want to go one step further and exploit this architecture for learning a one-to-many mapping in an unsupervised fashion, where pairs $\{x, y\}$ are not available.

2.7 CYCLE CONDITIONAL VARIATIONAL AUTO ENCODERS

Guo et al. propose a method for using a CVAE architecture in a cycle consistent model for resolving the many-to-one mappings problem called *CycleCVAE* [31]. In [31], the authors propose a cycle consistent model where the output of the one-to-many mapping are conditioned over a latent variable $z \sim N(0, 1)$. Given two domains X and Y, where Y has higher dimansionality than X, we

want to learn two mappings $f : X \times Z \to Y$ and $g : Y \to X$ that are cycle consistent. The two cycles that compose this model are

$$\{x, z\} \to \tilde{y} \to \hat{x}$$

$$y \to \{\tilde{x}, z\} \to \hat{y} \, ,$$

where we want to obtain reconstructions that are as closer as possible to the original input, i.e., $\hat{x} \approx x$ and $\hat{y} \approx y$. Stochastically handling the first cycle is easy, after sampling a $z \sim N(0, 1)$ we can generate $\tilde{y} = f_\theta(x, z)$ and then uniquely define $\hat{x} = g_\theta(\tilde{y})$. We can then train those networks by minimizing

$$\mathcal{L}_{cyclecvae,x} = \|\hat{x} - x\|_1$$

as in 2.4 [26]. Handling the second cycle is, instead, less trivial. Given y, we may compute $\tilde{x} = g_\theta(y)$ and then sample $z \sim N(0, 1)$ to find the $\hat{y} = f_\theta(\tilde{x}, z)$ that minimizes $\|y - \hat{y}\|_1$. In [29] authors prove how stochastically handling this cycle is impossible, since it would require an intractable number of z samples. To train this model, the authors propose structuring this second cycle using a `CVAE` architecture. Thus, they define a new network q_θ for learning a function that given any finds a distribution $p(z|y)$ of zs that maximizes the probability of recreating the same y through $f_\theta(\hat{x}, z)$ through the reparameterization trick. Thus the second cycle becomes

$$y \to \{\tilde{x} = g_\theta(y), z \sim N(\mu_q, \sigma_q^2)\} \to \hat{y} = f(\tilde{x}, z)$$

$$\mu_q, \sigma_q = q(y) \qquad \epsilon \sim N(0, 1) \qquad z = \mu_q + \sigma_q \cdot \epsilon \, .$$

Note that the $q(y)$ function proposed here differs from that proposed in [30] $q(x, y)$, since conditioning the generation of z also over $\tilde{x}$ does not give any new information to the network, since $\tilde{x}$ is merely a function of y. This second cycle is then trained by minimizing both the reconstruction error and KL divergence

$$\mathcal{L}_{cyclecvae,y} = \|y - \tilde{y}\|_1 + \mathbb{KL}[q_\theta(y)\|N(0, 1)] \, .$$

The total loss then becomes

$$\mathcal{L}_{cyclecvae} = \|x - \hat{x}\|_1 + \|y - \hat{y}\|_1 + \mathbb{KL}[q_\theta(y)\|N(0, 1)] \tag{2.6}$$

$$x \sim p(x), \quad y \sim p(y), \quad \hat{x} = g_\theta(f_\theta(x, z \sim N(0, 1)), \quad \hat{y} = f_\theta(g_\theta(x), z \sim q_\theta(y)).$$

As in [30] and [29], at inference time new samples of y are generated, starting from the same x, by sampling different $z \sim N(0, 1)$.

2.8 ASYMMETRIC GENERATIVE ADVERSARIAL NETWORKS

Another solution we consider for solving the hidden embedding problem is that proposed by Li et al. in [32]. In this paper, the authors propose an architecture similar to the `CycleCVAE` called `AsymmetricGAN`. Given two domains X and Y, where Y has a higher dimensionality than x, we want to learn the two mappings $Y \to X$ and $X \to Y$, that is a one-to-many. Thus the idea is to augment the X domain by mapping each $y \in Y$ with a pair $\{x \in X, z\}$, z sampled from a known distribution like a standard Gaussian. We want then to learn the two functions $f : X \times Z \to Y$ and $g : Y \to X$ using two cycle consistent `GAN`s f_θ and g_θ, with their Discriminators d_y and d_x respectively, together with a third function $q : Y \to Z$ for embedding the information necessary to generate a given y into an *auxiliary variable* $z \in Z$, learned by $q_t heta$. We can then define the two cycles

$$\{x, z\} \to \tilde{y} \to \{\hat{x}, \hat{z}\}$$

$$y \to \{\tilde{x}, \tilde{z}\} \to \hat{x} \, .$$

The first cycle is handled by sampling $z \sim N(0, 1)$ for then generating $\tilde{y} = f_\theta(x, z)$. The networks are trained in order to generate the $\tilde{y}$ that can reconstruct both $\hat{x} = g_\theta(\tilde{y})$ and $\hat{z} = q_\theta(\tilde{y})$ that minimizes the errors $\|x - \hat{x}\|_1$ and $\|z - \hat{z}\|_1$. The second cycle aims at reconstructing a given y via generating $\tilde{x} = g_\theta(y)$ and $\tilde{z} = q_\theta(y)$ that minimize the error $\|y - \hat{y}\|_1$, where $\hat{y} = f_\theta(\tilde{x}, \tilde{y})$. During the second cycle, an adversarial loss is introduced for forcing the Encoder q_θ to learn how to map the information necessary for reconstructing y into an auxiliary variable z that must be indistinguishable from those sampled by a standard Gaussian. This way, a Discriminator d_θ is introduced for identifying whether a given input z has been generated by q_θ or sampled from $p(z) = N(0, 1)$. The loss used for training this model is

$$\mathcal{L}_{asym} = \mathcal{L}_{a,x} + \mathcal{L}_{a,y} + \lambda_1 \mathcal{L}_{a,z} + \lambda_2 \mathcal{L}_{c,x} + \lambda_3 \mathcal{L}_{c,y} + \lambda_4 \mathcal{L}_{c,z} \, , \tag{2.7}$$

where the $\mathcal{L}_a$ terms refer to the adversarial loss [21] and the $\mathcal{L}_c$ to the cycle consistency terms introduced by [26], each properly weighted by a λ parameter.

3

BACKGROUND ON SAR IMAGE DESPECKLING

This chapter is provided to give an overview of the state-of-the-art approaches used to reduce the speckle in SAR images. After a brief introduction to standard filtering algorithms, we focus on Deep Learning approaches, which is the class of methodologies our method belongs to. Deep Learning literature is presented by discriminating generative and non-generative models. While the former make use of Generative Adversarial Networks for learning a mapping from the Speckled SAR domain to the Speckle-free one, the latter are trained using standard supervised learning.

3.1 SPECKLE MODEL

One of the biggest problems of the SAR despeckling task is to identify the distribution of the speckle that corrupts the clean underlying image containing the information of interest. In the SAR despeckling literature, the speckle that corrupts those images is supposed to be multiplicative. Let us assume y to be a speckle-corrupted image and x the corresponding clean one, then the former is corrupted by a speckle n such that

$$y = n \cdot x \,,$$

or equivalently, in the logarithm domain,

$$\log y = \log n + \log x \,.$$

In supervised methods, a dataset of SAR images is built by supposing the speckle n to be gamma distributed with unit mean and variance $\frac{1}{L}$, L being the number of looks of the SAR image:

$$p(n) = \frac{L^L n^{L-1} e^{-Ln}}{\Gamma(L)} \,, \qquad n \geqslant 0 \,, \qquad L \geqslant 0 \,.$$

However, this model only applies to homogeneous regions of the SAR image, e.g., grassland region, and it is not suitable in moderate and extremely heterogeneous areas. Indeed, the actual speckle distribution depends on the underlying scene, e.g., it changes from urban to wastelands areas. Thus, a starting point for our method is to find a function able to find the relation between the underlying scene and the speckle distribution and generate noise accordingly.

3.2 STATE-OF-THE-ART APPROACHES

3.2.1 *Classical Algorithms*

The first class of methodologies are those which make use of adaptive filters like [1], [2] and [3]. In those papers, the authors propose algorithms that use a moving window over the entire image. [1], and [2] compute a linear combination of the central pixel intensity and the average intensity of neighbour pixels. Instead, [3] use an exponentially damped kernel that behaves in a fashion similar to a low-pass filter or an identity filter, depending on whether the local coefficient of variation is small or large. Even if those algorithms have good results under some circumstances, they are highly constrained by choice of the window and, in general, are applicable only over homogeneous areas and characterized by blurry artefacts.

Another class of algorithms are those which reduce the noise by thresholding the coefficients of the Discrete Wavelet Transform (DWT) of the log-transformed single look image. Xie, Pierce, and Ulaby [4] outperforms the enhanced filter of [1] by integrating the wavelet denoising technique with a regularisation procedure based on Markov random fields (MRF). Argenti and Alparone [5] apply a Minimum Mean-Square Error (MMSE) filtering in the undecimated wavelet domain, reaching better performances than [2]. Those methods still fail in preserving the backscatter mean over homogeneous areas and details, and generate artificial artefacts.

Recently, Non-Local (NL) filtering methods have been introduced. Unlike local filters, NL means filtering algorithms, like those proposed in [6] and [7], take a mean of all pixels in the image, weighted by how similar these pixels are to the target. This class of algorithms gets better results it terms of post-filtering clarity and detail than local mean algorithms. Nevertheless, those algorithms are very computationally expensive. A NL filtering algorithm is the Block-Matching 3-D (BM3D) denoising algorithm proposed in [6]. In this paper, authors group image patches into 3-D arrays based on their similarity and perform a collaborative filtering procedure to obtain the 2-D estimates for all grouped blocks. This idea is then extended by Parrilli et al. to deal with SAR images. The SAR-Block-Matching 3-D (SAR-BM3D) [7] algorithm groups similar image patches through an ad hoc similarity measure that takes into account the actual speckle statistics and by adopting the local linear MMSE (LLMMSE) criterion. Given that it is considered one of the best approach among non-learnable filtering methods, we used SAR-BM3D as one of the baselines to compare our results with (see Chapter 5).

Approaches using a Total Variation (TV) term are also popular [8]. Those methods combine a data fitting term with a TV regularization for encouraging smooth results while preserving edges. In those algorithms, the weight given to the regularization term is crucial. Large values may lead to over-smoothed

results without properly preserving edges and details, while small values may not sufficiently remove the noise.

Finally, it is worth mentioning those frameworks that use multi-temporal stacks of SAR images that use the extracted temporal statistics to develop space-adaptive filters for the single image. Ferretti et al. [9] propose an algorithm that aims at identifying Punctual Scatters (PS), with high reflectivity values and phase stability over the whole period of observation, and Distributed Scatters (DS), composed of many neighbouring pixels sharing similar values but having low average temporal coherence. The proposed solution wants to spatially average the data over statistically homogeneous areas without compromising the identification of coherent point-wise scatterers. It identifies DSs by applying the two-sample Kolmogorov–Smirnov (KS) test within an estimation window where pixels share similar statistics of the considered centre pixel. Then, the obtained DSs identify homogeneous areas in the image and their intensities are averaged to reduce the speckle while preserving PSs. We suppose that, at least under some circumstances in which the underlying scene does not change significantly over time, this algorithm will calculate an exact estimate of the despeckled image. This approach is currently considered as the state-of-the-art for SAR image despeckling. However, it needs entire temporal data stacks and it is based on the assumption that the underlying scene does not change significantly over time. The approach proposed in this work of thesis, instead, propose a solution for achieving similar results with respect to [9] over a single speckled SAR image, allowing comparable de-speckling performances even in those scenes where a certain time stability cannot be achieved.

3.2.2 *Deep Learning Approaches*

In recent years, Deep Learning showed outstanding performances over various computer vision tasks. This class of algorithms has then been proposed for the exploration of the remote sensing field, for tasks like the SAR despeckling problem investigated in this thesis. In the literature, we can distinguish among two different DL approaches: non-generative models and generative models.

NON-GENERATIVE MODELS Non-generative models, which use classical supervised learning techniques, need paired data to be trained. In SAR domain there is no such pair of speckled and speckle-free images available, so the way the train dataset is built is a critical choice for those methods. In literature we can find various methods that propose to build this dataset relying on the knowledge that can be gathered over a temporal stack. Chierchia et al. propose a deep CNN trained using a dataset where the clean image is obtained by averaging a temporal stack and keeping only the regions with

no significant temporal changes. Those models suffer from the limitations coming from a dataset built this way and may have to deal with data scarcity at training time.

One of the methods that aim at overcoming this limitation is that proposed by Lattari et al. [11]. The authors propose an encoder-decoder architecture that extracts high-level features from the input image and then upsamples those features until the exact dimensions of the input. For not losing any information during the upsampling process, a set of skip connections concatenate each downsampling step to the corresponding upsampling one, i.e., the one with equal dimensions. The network is trained accordingly to the residual paradigm: the function learned does not aim to extract the clean image directly, but rather a residual n such that, given x as input and y as ground truth,

$$y = x + n \, .$$

This approach is proven to be effective in previous related works [12]. Differently from other approaches which work on the logarithm domain, even if they are assuming a multiplicative noise model, they are not calculating the residual on the logarithms of the synthetic image and the ground truth. During the training phase, they feed the network with corrupted images and the corresponding targets. This dataset is built by making assumptions on the speckle distribution and then sampling from this distribution a noise mask multiplied to a ground truth greyscale aerial optical image, obtaining the corresponding noisy. The model aims at minimizing a Minimum Squared Error loss between the ground truth residual $n = y - x$ and the one computed by the network $\tilde{n} = \phi(x, \Theta)$

$$L_{mse} = \frac{1}{2N} \sum_{i=1}^{N} \|(\phi(x_i, \Theta) - n_i)\|_2^2 \, .$$

Then, the network is fine-tuned by feeding it with images taken from the actual SAR domain. Thus, starting from a clean image obtained by computing the mean of a temporal stack of real speckled SAR images, they corrupt each patch with artificial noise as done in the previous step. During this finetuning, the network can learn how to deal with higher resolution images from the actual SAR domain. For improving the performances of the network, a properly weghted Total Variation term is added to this loss in order to have smoother outputs and remove undesired artifacts, this TV term is computed as

$$L_{tv} = \sum_{i,j} e^{-|\nabla_h x_{ij}|} |\nabla_h \hat{x}_{ij}| + e^{-|\nabla_v x_{ij}|} |\nabla_v \hat{x}_{ij}|$$

where $\nabla_h \hat{x}$ and $\nabla_v \hat{x}$ are gradients computed on reconstructed image on both horizontal and vertical directions and defined as

$$\nabla_h \hat{x}_{ij} = \hat{x}_{i,j+1} - \hat{x}_{i,j}$$

$$\nabla_v \hat{x}_{ij} = \hat{x}_{i+1,j} - \hat{x}_{i,j}$$

while $\nabla_h x$ and $\nabla_v x$ are the same gradients but computed on the speckle-free reference image. Even if this work demonstrated outperforming despeckling capabilities, it is trained on simulated speckled images which prevents the model from learning the real distribution of SAR data. Our work proposes a solution for overcoming this limitation. We compared the results obtained by our approach with the one presented in [11] during the conducted experiments (see Chapter 5).

One of the works that propose an alternative solution to the data simulation is that proposed by Dalsasso, Denis, and Tupin [18]. Here, the authors present a semi-self supervised approach, called `sar2sar`, based on the rationale of `noise2noise` [17]. The `noise2noise` model wants to deal with a situation in which pairs (y, x) of noisy and corresponding clean images are not available for training a model in a supervised fashion, minimizing the ℓ_2 loss

$$\mathbb{E}_X[\|\phi(y, \Theta) - x\|^2] \, .$$

For doing so, the authors propose to use noisy pairs (y_1, y_2), where both samples are drawn from the same conditioned distribution $p_{Y|X}$, i.e., are two different noisy realizations with the same underlying scene x, aiming at minimizing an equivalent loss

$$\mathbb{E}_X[\|\phi(y_1, \Theta) - y_2\|^2] \, .$$

The authors make two assumptions: the scene x is static,i.e., does not change significantly over time, and the noise centred, i.e., $\mathbb{E}_{Y|X}[y] = x$. This method is then applied to the SAR despeckling problem in [18]. The authors propose to apply the method proposed in [17], taking into account changes that may occur between two different SAR images taken over the same area in distinct moments. Given a network that can estimate $\hat{x}_1$ and $\hat{x}_2$ from y_1 and y_2, the authors propose to partially compensate changes that may occur in y_2 with the image

$$y_1 - \hat{x}_1 + \hat{x}_2$$

which more closely resembles y_2. The `sar2sar` approach then combines the idea introduced in `noise2noise` with the compensation for changes over a temporal series of SAR images, performing the restoration into the log-domain by a deep network. Authors divide the training into three steps. At first the despeckling network is pre-trained in a supervised fashion over a synthetically generated dataset, then on pairs of images extracted randomly from a time-series, compensating the second image for changes based on reflectivities estimated with the network trained in previous step. Finally, a refinement step is performed using the updated network weights to obtain better compensation for changes. Since the first step uses a supervised approach, the authors refer to the `sar2sar` as a semi-supervised algorithm. This

work aims at overcoming issues related to those models trained on artificial noise. Even though the network is pre-trained over a synthetic dataset, the actual training phase tries to make the network learn how to deal with actual SAR speckle by compensating images for changes. While performing this compensation, though, it is essential to point out that the noise mask extracted from the compensated image may be highly correlated with the objects contained in the underlying scene. For optimal compensation, the stack of images used should not have significant changes that drastically influence the speckle over time, a condition that may be hard to achieve in particular areas, e.g., urban areas.

Another semi-self supervised approach is that proposed by Mullissa et al. in [20]. In this paper, the authors present a model that not only aims at learning a function for cleaning speckled images but at the same time attempt at estimating an accurate SAR speckle distribution. In particular, their `DeSpeckNet` model does not use any synthetic dataset for training. This model is composed of two Convolutional Neural Networks: one for learning a function that takes a noisy image y as input and outputs the corresponding clean one x, and another CNN for extracting the noise n, such that they can reconstruct the original y as

$$y = n \cdot x \, .$$

Those networks do not use any pooling layer, in order to avoid upsampling layers with all their extra computational burden and complexity, and are trained using three different losses properly weighted:

$$\mathcal{L} = \lambda_{clean}\mathcal{L}_{clean} + \lambda_{noisy}\mathcal{L}_{noisy} + \lambda_{tv}\mathcal{L}_{tv} \, .$$

The first element of this loss function is a Mean Squared Error loss between a clean label x and the output of the despeckling network $\hat{x}$:

$$\mathcal{L}_{clean}(x, \hat{x}) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \, ,$$

where N is the number of pixels in a training patch. The second one is a MSE loss between the original noisy image y and the reconstruction, obtained via the noise estimation n computed by the noise extraction network, $\hat{y} = n\hat{x}$

$$\mathcal{L}_{noisy}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \, .$$

A Total Variation loss is then added for smoothing the clean image:

$$\mathcal{L}_{tv}(\hat{x}) = \sum_{i} |\nabla \hat{x}_i| \, .$$

This term minimizes the absolute differences between neighboring pixel-values, enforcing smoothness while preserving edges. The authors split the

training into two steps: first, they train the model in a supervised fashion by feeding it with pairs of clean and speckled images, then the model is finetuned with speckled SAR images only. In the first step, the pairs of images are extracted from multi-temporal stacks of SAR images, finding noisy patches stable in time, i.e., the scene must not have large temporal variation. The algorithm then pairs those images with the corresponding one computed via a mean over the stack. In the unsupervised step, the algorithm feeds the network with speckled images only, without any clean label; calculating the clean term of the loss function $\mathcal{L}_{clean}$ as an MSE between the clean estimation $\hat{x}$ and the noisy input y and reducing the associated weight λ_{clean}:

$$\mathcal{L}_{clean,2}(y, \hat{x}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{x}_i)^2 \, .$$

This loss has the effect of maintaining the solution close to the original image to preserve spatial structures while the other losses smooth ($\mathcal{L}_{tv}$) and denoise ($\mathcal{L}_{noisy}$). The training may be highly biased on the choice made while building a dataset for step one. It may present the same limitations described for [18]. By looking for time-invariant images, the dataset will lack those captured in particular areas that cannot be stable over time. This way, it is impossible to train the network over a complete set of images capturing all the possible scenarios.

GENERATIVE MODELS Another class of Deep Learning approaches is that of generative models. The idea behind those approaches is to exploit the GANs capabilities to learn those high level features that characterize the clean images domain that a simple MSE loss cannot capture.

The first generative model we want to analyze is that proposed by Wang, Zhang, and Patel in [15]. This approach lays on the idea that the MSE loss used in previous methods measures the error for each pixel independently from the rest of the image without putting any attention on higher-level features. They propose to use a GAN to overcome this issue and force a network to learn how to map speckled images into the clean domain and preserve those features. The dataset used for training this model is generated by sampling noise n from a Gamma distribution and then multiplying it to the clean image x:

$$y = n \cdot x \, .$$

The model aims at minimizing a loss function composed by three terms:

$$\mathcal{L} = \mathcal{L}_e + \lambda_a \mathcal{L}_a + \lambda_p \mathcal{L}_p \, .$$

The $\mathcal{L}_e$ term is a per-pixel Euclidean loss between y and the predicted $\hat{x} = G(y)$

$$\mathcal{L}_e = \frac{1}{WH} \sum_{w=1}^{W} \sum_{h=1}^{H} \|G(y^{h,w}) - x^{h,w}\|_2^2$$

where $W \times H$ is the size of the input images. The term $\mathcal{L}_a$ is the adversarial loss defined in [21], and serves as a guide for the Generator network to learn how to produce images that are indistinguishable from those sampled from the set of ground-truth clean images:

$$\mathcal{L}_a = -\frac{1}{N}\sum_{i=1}^{N} \log D(\hat{X}_i) \,,$$

where $\hat{X}$ is a set of N images produced by the network G. The last term $\mathcal{L}_p$ is a perceptual loss, computed as an euclidean distance between the two high level feature vectors of the input and output images of G, and helps the generator network learning to keep unchanged those features of the input:

$$\mathcal{L}_p = \frac{1}{WH}\sum_{w=1}^{W}\sum_{h=1}^{H} \|V(G(y^{h,w})) - V(x^{h,w})\| \,.$$

The V function extracts the feature vector from the images, those vectors are obtained from the output of the *relu7* layer of a pretrained `VGG16` network. Combining those two losses with the Euclidean distance makes it possible for the despeckling network to learn how to effectively reduce the speckle while preserving those features that give the clean image a smoother look, preserving edges.

Liu, Li, and Jiao [16] propose a different approach for exploiting the efficiency of GANs: they combine a Euclidean per-pixel distance between predicted output and ground-truth with an adversarial loss for preserving higher-level features. In [16], the authors present a GAN with a Generator architecture composed of several convolutional layers without any downsampling and a 70×70 `PatchGAN` [22] Discriminator. For the training phase, authors build a dataset via sampling multiplicative noise n from a gamma distribution and applying it to a clean greyscale optical image x:

$$y = n \cdot x \,.$$

Moreover, use a combination of two different losses for making the generator learn the despeckling function. The first loss is a weighted sum of an Euclidean per-pixel distance and a Total Variation term, and serves for making the generator network learn the mapping from the speckled domain to the corresponding clean image while at the same time preserving sharp edges and details:

$$\mathcal{L}_g = \mathcal{L}_e + \lambda_{tv}\mathcal{L}_{tv}$$

where

$$\mathcal{L}_e = \frac{1}{WH}\sum_{w=1}^{W}\sum_{h=1}^{H} \|G(y^{w,h}) - x^{w,h}\|_2^2$$

$$\mathcal{L}_{tv} = \sum_{w=1}^{W} \sum_{h=1}^{H} \sqrt{(\hat{x}^{w+1,h} - \hat{x}^{w,h})^2 + (\hat{x}^{w,h+1} - \hat{x}^{w,h})^2}$$

and $\hat{x} = G(y)$. The second is an adversarial loss: this loss is used for training the Discriminator and making it learn high-level features of the clean images and to use them for distinguishing those generated by G from those coming from the ground-truth dataset. At the same time G aims at maximizing the error of the Discriminator, thus learning how to produce clean images that D is not able to distinguish from the real ones. The D network takes as input a noisy/clean pair, that may be composed by concatenating noisy images with the corresponding clean label, i.e., the $s = \{y, x\}$ pairs, or by concatenating input and corresponding output of the Generator, i.e., $t = \{y, \hat{x}\}$. Thus the adversarial loss[21] becomes:

$$\mathcal{L}_a = \max_D \mathbb{E}[\log(D(s)) + \log(1 - D(t))] \, .$$

Final loss is then

$$\mathcal{L} = \mathcal{L}_a + \lambda_g \mathcal{L}_g \, .$$

Both [15] and [16] are assuming a fixed speckle distribution. As already discussed for [11], the speckle distribution of real SAR images depends on the underlying scene, thus those models will result in having lower performances over those areas whose speckle is very dissimilar to the synthetic one.

Gu, Zhang, and Wang [14] propose a generative model that learns how to sample noise patch from the real speckle distribution $p(x)$, without making any assumptions about the distribution itself like previous works, by training a noise generation network using an adversarial loss. In [14] the authors build a dataset of noise samples starting from homogeneous patches taken out of real SAR images, e.g., areas of oceans, wastelands or farmlands. The formula used for extracting the speckle from those patches is:

$$\log(n) = \log(\hat{y}) - \mathit{mean}(\log(\hat{y})) \, ,$$

where n is the multiplicative speckle, $\hat{y}$ is a homogeneous area, and the operation *mean* calculates the mean value. With a large enough dataset of noise samples, a noise generator network can be trained using the adversarial loss of Equation 2.1 [21]:

$$\min_G \max_D \mathcal{L}_{gan}(D, G) = \mathbb{E}_{x \sim p(x)}[\log D(x)] + \mathbb{E}_{z \sim N(0,1)}[1 - \log D(G(z))] \, ,$$

where $p(x)$ is the distribution of real SAR noise samples, G is the noise generator network and D the discriminator. An optimally trained network G can be able to sample noise patches from the distribution $p(x)$ itself. Using this network, the authors build a dataset of clean/speckled image pairs for proceeding in the training of the despeckling network. The authors propose to

use a dataset of grayscale optical aerial images with a runtime generated noisy counterpart, obtained by feeding G with a z vector, sampled from a standard Gaussian distribution, and adding the result to the image. The despeckler is a CNN, with a downsampling and an upsampling path connected via residual connections, which add the corresponding downsampling layer output to the upsampling one. The loss used for the training is

$$\mathcal{L}_\Theta = \frac{1}{2WH}\|f(x,\Theta) - y\|_2^2 \,,$$

where x and y are the noisy input image and its corresponding clean label, both with a size of $W \times H$. The generator network trained this way will generate noise samples sampled from the noise distribution of the homogeneous areas extracted before training. This way, despeckler will struggle on areas with a different speckle distribution from that learned by the generator.

Based on this state of the art analysis, we want to propose a solution that can learn how to despeckle SAR images in an unsupervised fashion. In particular we take inspiration from the unpaired image to image translation method proposed by Zhu et al. [26]. We want to overcome the lack of speckled/clean SAR image pairs by training two cycle-consistent networks: a Speckle Generator and a Despeckler. The Speckle Generator is capable of learning the actual SAR speckle distribution for generating speckle corrupted images belonging to the same distribution of the real ones while the Despeckler can exploit this knowledge for learning how to despeckle those images.

In the following Chapter we will describe in detail the proposed solution and the dataset used for training and validate the model.

4

PROPOSED SOLUTION

In this chapter we describe in detail the architecture and the training of the solution we propose in this thesis work. The proposed solution is composed by two networks: the Despeckler (G_d) and the Speckle Generator (G_n). The Despeckler is trained in order to learn how to reduce the speckle of real SAR images. The Speckle Generator is trained in order to learn the speckle distribution over the input clean patch and to sample one of the infinite possible speckle realizations via a Gaussian vector. The two networks are trained in an unsupervised fashion, using the unpaired image-to-image translation method proposed in [26] and handling the one-to-many mapping on the speckle generation side using the technique of [31]. Differently from the CycleGAN model of [26], the training of the two networks is not performed using two Generative Adversarial Networks, due to the well known training stability problem they are subject to, as pointed out in [23]. Instead, we use the improved Wasserstain GAN training, with Gradient Penalty, proposed in [25].

4.1 ARCHITECTURE

For the networks architecture, we take inspiration from the `UNet` [13] architecture. This architecture, originally implemented for image segmentation tasks, has proven to have awesome performance for SAR image despeckling, as showed by [11]. Both Despeckler and Speckle Generator are composed by 5-step down-sampling and an up-sampling paths, connected via skip connections that concatenate the output of a down-sampling step with the corresponding up-sampling one. Each step is composed by a set of two 3×3 convolutional layers with 32 starting filters and doubling this number on every step. Those steps also comprise Batch Normalizations and ReLU activation functions. The down-sampling operation is performed by a 2×2 MaxPooling layer, while the upsampling operation by a 2×2 with a stride of 2 Transpose Convolution. The last layer of the network is a 1x1 convolutional layer for reducing the number of the last convolutional layer filters from 32 to 1. The Despeckler takes as input a logarithmic SAR image y and outputs the logarithmic residual speckle n and calculates the despeckled patch x via

$$\log(x) = \log(y) - \log(n) \ .$$

The architecture of the Despeckler is shown in Figure 4.1.

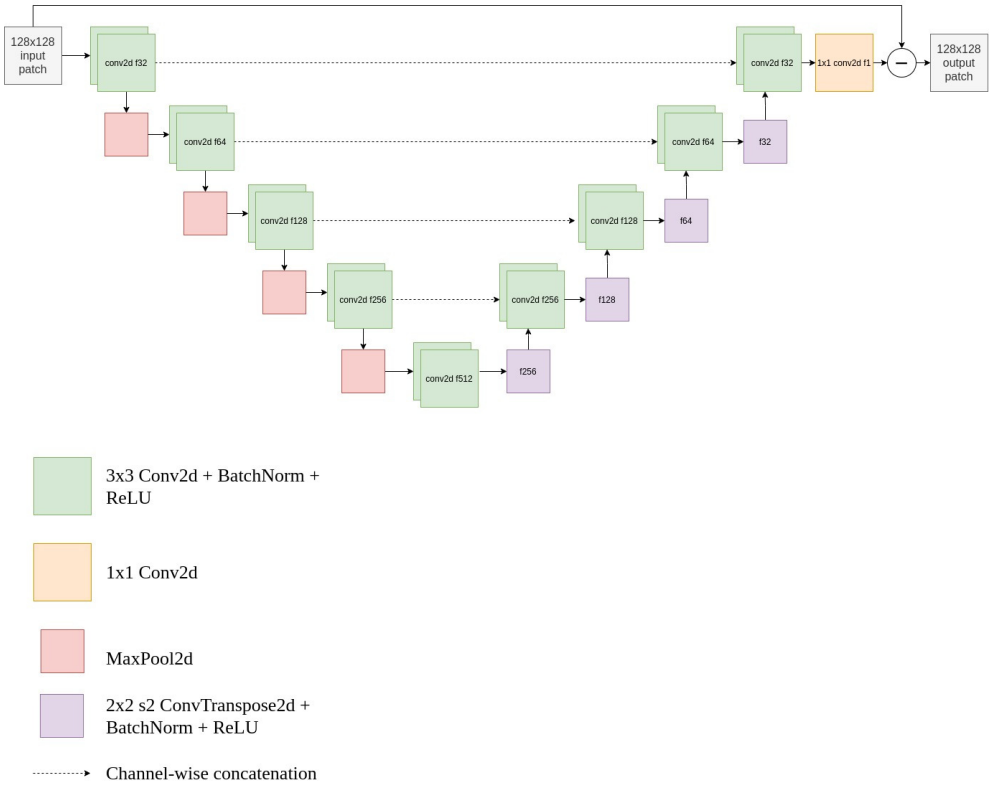


Figure 4.1: Architecture of the Despeckler. The network takes as input a 128×128 logarithmic SAR patch x and outputs the residual speckle $\tilde{n}$ for computing the despeckled image $\tilde{x} = y - \tilde{n}$. The architecture is composed by a down-sampling and an up-sampling path, with 5 steps each connected via channel-wise concatenation skip connections. Each step comprises a set of two blocks made of 3×3 Convolution, Batch Normalization and ReLU activation function. The down-sampling operation is performed by a 2×2 MaxPooling layer, while the up-sampling by a 2×2 Transpose Convolution with a stride of 2. Finally, a 1×1 convolution reduces the channels of the last convolution from 32 to 1.

The Speckle Generator takes as input a logarithmic clean patch x and a gaussian vector of length 512 and outputs the logarithmic residual speckle n. The inputs of this networks are combined by replicating the vector 128×128 times forming a tensor of shape $512 \times 128 \times 128$ and channel-wise concatenating this tensor with the input patch. The generated SAR patch y is calculated via

$$\log(y) = \log(x) + \log(n) \, .$$

The architecture of the Speckle Generator is shown in Figure 4.2.

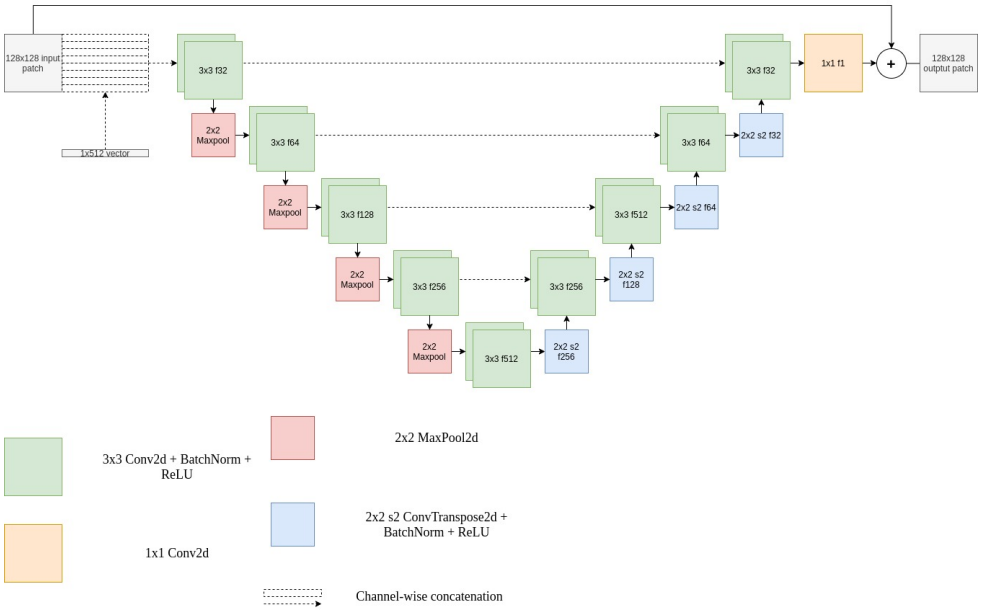


Figure 4.2: Architecture of the Speckle Generator. The network takes as input a 128×128 logarithmic clean patch x and a vector of length 512 and outputs the residual speckle $\tilde{n}$ for computing the generated SAR image $\tilde{y} = x + \tilde{n}$. The input tensor is composed by the channel wise concatenation of the patch with the vector replicated until the shape of $512 \times 128 \times 128$. The architecture is composed by a down-sampling and an up-sampling path, with 5 steps each connected via channel-wise concatenation skip connections. Each step comprises a set of two blocks made of 3×3 Convolution, Batch Normalization and ReLU activation function. The down-sampling operation is performed by a 2×2 MaxPooling layer, while the up-sampling by a 2×2 Transpose Convolution with a stride of 2. Finally, a 1×1 convolution reduces the channels of the last convolution from 32 to 1.

4.2 TRAINING

We trained our networks using the WGAN-GP algorithm [25], due to the well known training problems of classical GAN models, as shown in [23], the improved version of the WGANs [23] which showed to resolve vanishing and exploding gradient problems. Thus, we introduced two Critics networks, D_d and D_n, as defined in the reference paper [23]. Those networks are composed by 5 down-sampling steps, which comprises a 3×3 convolutional layer with a number of channels that starts from 32 and doubles at every step, a ReLU activation function and a 2×2 MaxPooling. The output of those network is a real number and is obtained trough a one-layer MLP. The architecture of those networks is represented in Figure 4.3.

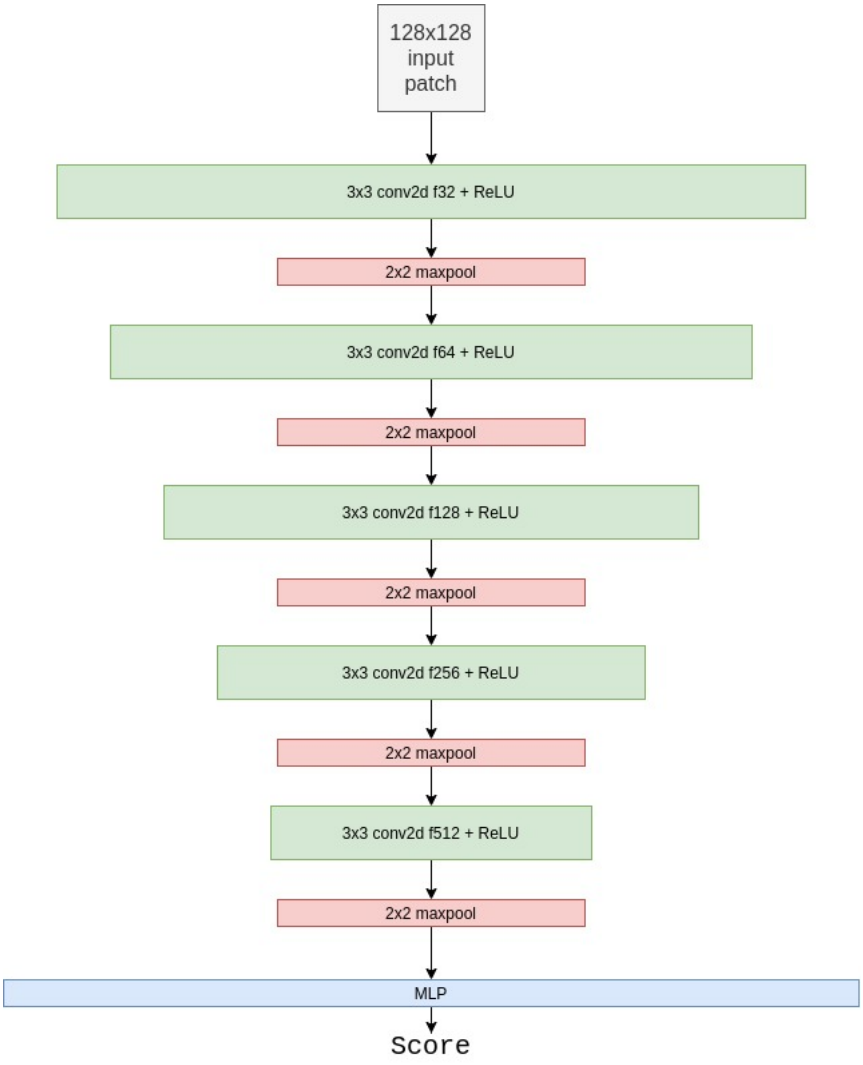


Figure 4.3: Critic networks architecture. Those networks are composed by 5 steps of 3×3 convolutional layer, ReLU activatin function and 2×2 MaxPooling. The final step is then flattened and fed to a 1-layer MLP for obtaining the score of the input patch. Those networks do not make use of any Batch Normalization due to the incompatibility of this layer with the improved WGAN training algorithm presented in [25].

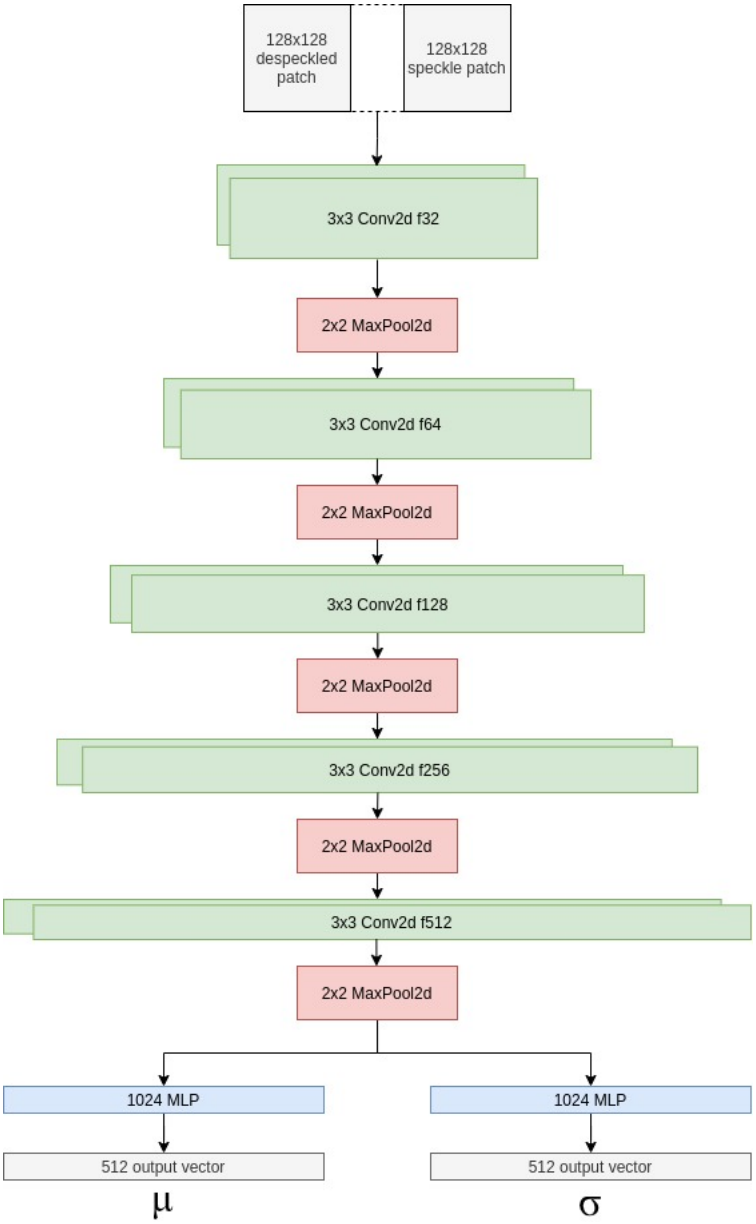


Figure 4.4: Encoder architecture. The Encoder input is the channel-wise concatenation of the despeckled patch $\tilde{x} = y - G_d(y)$ obtained from the logarithmic SAR patch y with the residual speckle $\tilde{n} = G_d(y)$. This network is composed by 5 steps interleaved by 2×2 MaxPooling layers. Each step is composed by two sets of 3×3 Convolution, Batch Normalization and ReLU activation function. The number of filters for each convolutional layer starts from 32 and doubles at each down-sampling operation. The last downsampling is flattened and fed to two 2-layers MLP, with 1024 hidden units. Those two MLPs outputs the two vectors representing μ and $\log(\sigma)$ of the $N(\mu, \sigma)$ distribution from which sample the latent vector $\tilde{z}$ using the reparameterization trick.

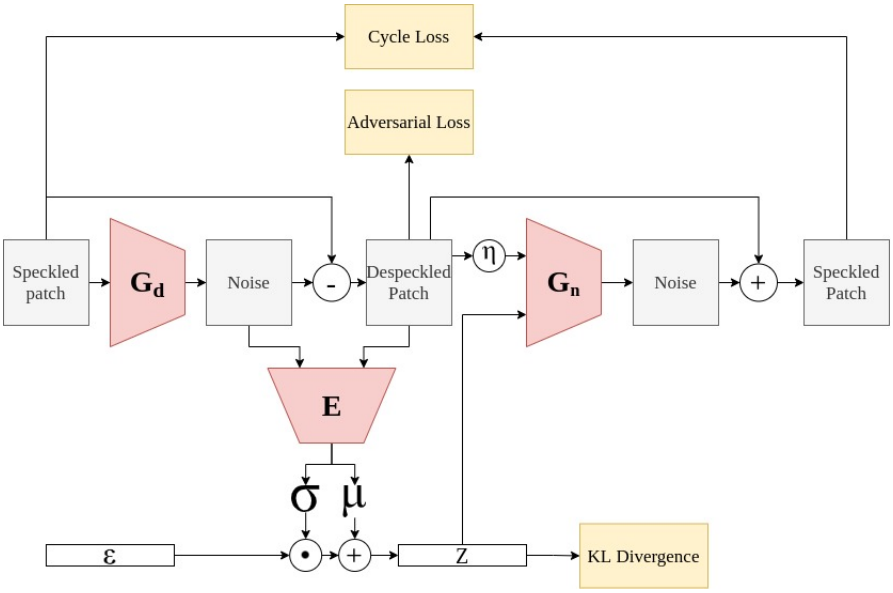


Figure 4.5: Schema of the speckled loop. A logarithmic SAR patch $y \sim p(y)$ is fed to G_d for computing the residual speckle $\tilde{n} = G_d(y)$ and the corresponding clean patch $\tilde{x} = y - \tilde{n}$ that minimizes the adversarial loss. The latent vector $\tilde{z}$ is then sampled from a Gaussian distribution $N(\mu, \sigma)$, where $\mu, \log(\sigma) = E(\tilde{x}, \tilde{n})$, such that it minimizes the KL divergence introduced in [30]. Finally $\tilde{x}$ is corrupted with a Gaussian noise η and $\tilde{z}$ and $\tilde{x}_\eta$ are used for building the reconstruction $\hat{y} = \tilde{x} + G_n(\tilde{x}_\eta, \tilde{z})$ that minimizes the cycle consistency loss $\|y - \hat{y}\|_1$.

For training our model, we split each training step in two loops, namely the clean loop and the speckled loop. Let us define x a logarithmic clean patch and y a logarithmic SAR patch taken from the training dataset. During the clean loop we sample a vector z from a standard Gaussian distribution and use this for generating a fake SAR patch $\tilde{y}$, we then despeckle this patch obtaining the reconstruction of the starting clean patch $\hat{x}$. The speckled loop starts by despeckling the y patch, obtaining a despeckled patch $\tilde{x}$, and by generating a latent vector $\tilde{z}$ through an Encoder network (E), as introduced by the `CVAE` [30] model, used for building the SAR image reconstruction $\hat{y}$. The Encoder architecture, shown in Figure 4.4, is composed by a main module and two MLPs. The main module is a 5 step CNN composed by two blocks of 3×3 Convolution, BatchNormalization and ReLU activation function, each step followed by a 2×2 MaxPooling layer. The convolutional layer have a number of filters that starts from 32 and doubles every step. The output of the main module is flattened and fed to the two MLPs. The MLPs have 1024 hidden units and outputs two vector of length 512 representing the μ and $\log(\sigma)$ parameters used for sampling the latent vector $\tilde{(z)}$ using the reparameterization trick. The Encoder input is the channel-wise concatenation of the despeckled patch $\tilde{x} = y - G_d(y)$ obtained from the logarithmic SAR

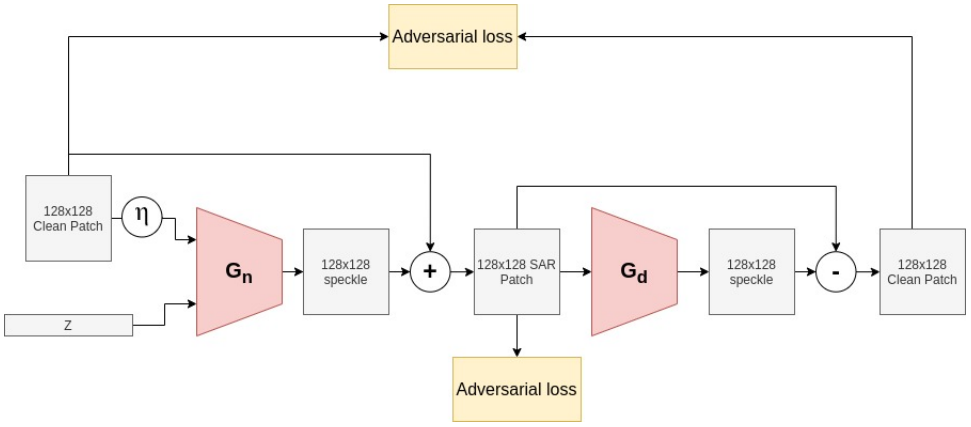


Figure 4.6: Schema of the clean loop. A logarithmic clean patch $x \sim p(x)$ is corrupted with a Gaussian noise η and fed to G_n together with a Gaussian vector $z \sim N(0,1)$ for computing the residual speckle $\tilde{n} = G_n(x,z)$ and the corresponding SAR patch $\tilde{y} = x + \tilde{n}$ that minimizes the adversarial loss. Finally $\tilde{y}$ is used for building the reconstruction $\hat{x} = \tilde{y} - G_d(\tilde{y})$ that minimizes the cycle consistency loss $\|x - \hat{x}\|_1$.

patch y with the residual speckle $\tilde{n} = G_d(y)$. However, the set of clean images we are using is computed via the algorithm proposed in [9] over a multi-temporal stack of SAR images, they presents a residual low-amplitude speckle. Since this residual resulted to influence the generation of speckle, we add, at training time, a low-amplitude Gaussian noise in order to force our networks to ignore this residual. In particular, we added this noise on the clean patch x, when feeding it to the Speckle Generator at the beginning of the clean loop, and on the despeckled patch $\tilde{x}$ right before generating the reconstruction $\hat{y}$ at the end of the speckled loop. The formula we used to corrupt a clean patch x is:

$$x_\eta = \log(\exp(x) + \eta)$$

$$\eta = 0.05 \times \epsilon \qquad \epsilon \sim N(0,1)\ .$$

The clean and speckled loop are schematized in Figure 4.6 and Figure 4.5 respectively.

The loss function we introduced is composed by 4 terms. Let us define $p(x)$ the distribution of the logarithmic clean patches, $p(y)$ the distribution of the logarithmic SAR patches, $\tilde{x} = y - G_d(y)$ the despeckled patch obtained from $y \sim p(y)$ and $\tilde{y} = x + G_n(x,z)$ the fake SAR patch generated by the Speckle Generator starting from the clean one $x \sim p(x)$ and a Gaussian vector $z \sim N(0,1)$. The first two terms represents the adversarial losses, introduced in Equation 2.3, for the Despeckler and the Speckle Generator:

$$\mathcal{L}_{a,d} = \mathbb{E}_{y\sim p(y)}[D_d(\tilde{x})] - \mathbb{E}_{x\sim p(x)}[D_d(x)] + \lambda_{gp}\mathbb{E}_{\dot{x}\sim p(\dot{x})}[(\|\nabla D_d(\dot{x})\|_2 - 1)^2],$$

$$\mathcal{L}_{a,n} = \mathbb{E}_{x\sim p(x)}[D_n(\tilde{y})] - \mathbb{E}_{y\sim p(y)}[D_n(y)] + \lambda_{gp}\mathbb{E}_{\dot{y}\sim p(\dot{y})}[(\|\nabla D_n(\dot{y})\|_2 - 1)^2].$$

As suggested in [25], we set $\lambda_{gp} = 10$. Given a despeckled image $\tilde{x}$ and a speckled generated $\tilde{y}$, let us define the reconstruction of y and x respectively $\hat{x} = \tilde{y} - G_d(\tilde{y})$ and $\hat{y} = \tilde{x} + G_n(\tilde{x}, \tilde{z})$, $\tilde{z}$ being the latent vector obtained throught the reparameterization trick

$$\tilde{z} = \mu + \sigma \odot \epsilon, \quad \epsilon \sim N(0,1), \quad \mu, \log(\sigma) = E(\tilde{x}, \tilde{n}), \quad \tilde{n} = G_d(y) \, .$$

The third term of our loss is the cycle consistency term introduced in [26]:

$$\mathcal{L}_{cyc} = \mathbb{E}_{x \sim p(x)}[\|x - \hat{x}\|_1] + \mathbb{E}_{y \sim p(y)}[\|y - \hat{y}\|_1] \, .$$

The fourth term of our loss is the KL-divergence of Equation 2.5 introduced in [29]:

$$\mathcal{L}_{kl} = 0.5(\sigma + \mu_q - 1 - \log(\sigma)) \, .$$

The complete loss of our model becomes:

$$\mathcal{L} = \mathcal{L}_{a,d} + \mathcal{L}_{a,n} + \lambda_{cyc}\mathcal{L}_{cyc} + \lambda_{kl}\mathcal{L}_{kl} \, ,$$

with $\lambda_{kl} = 0.025$ and $\lambda_{cyc} = 10$.

5

EXPERIMENTAL RESULTS

This chapter describes the large suite of experiments we conducted during this work of thesis. Experiments are presented to describe the various steps that have led to our final approach described in Chapter 4.

5.1 DATASET

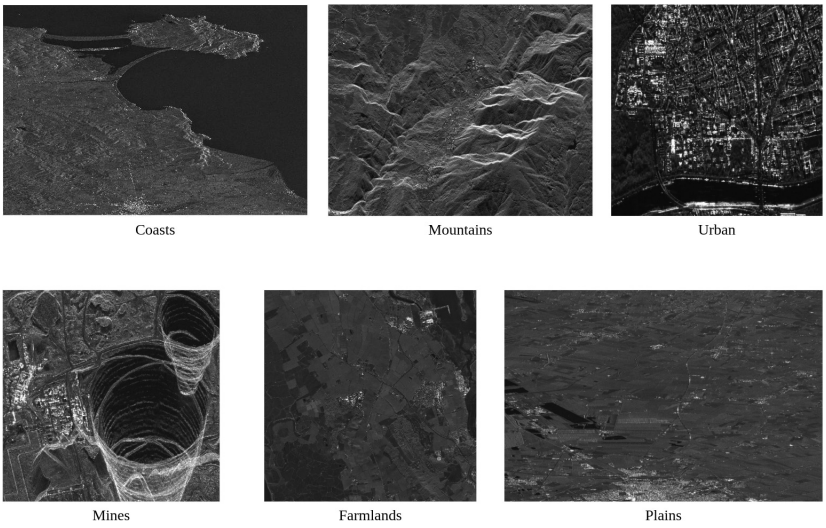


Figure 5.1: Example of clean dataset areas

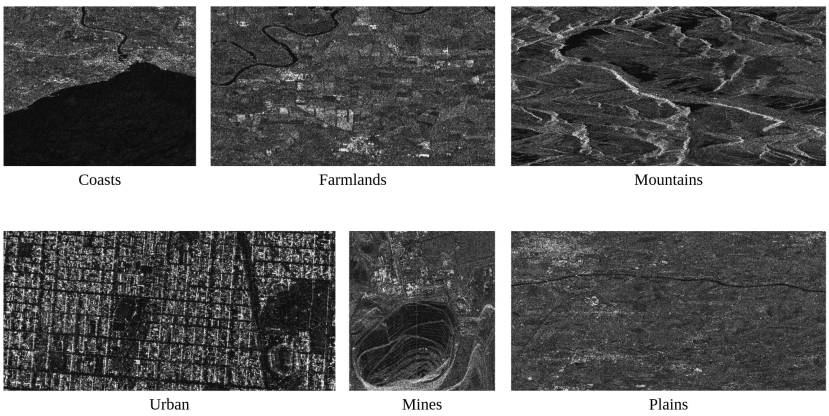


Figure 5.2: Example of speckled dataset areas

The dataset we used for training the solution proposed in Chapter 4 is composed of two subsets: a subset of speckled SAR images and one of speckle-free Means computed over multi-temporal stacks using the algorithm of [9]. The images composing the speckled dataset and the stacks are taken by Sentinel-1 satellites over different areas of Italy and represent many different areas and scenes we expect may have peculiar speckle distributions. Examples of images composing this dataset are shown in Figure 5.2, which shows some areas of the speckled subset, and Figure 5.1, which shows the areas of the speckle-free one. These images show the various areas we considered to be included in this dataset considering different surface features which exhibit different scattering characteristics of the SAR signal:

- Urban areas with very strong backscatter due to the presence of buildings;
- Mountains and forest with medium backscatter due to the presence of trees;
- Calm water with smooth surfaces and low backscatter;
- Rough sea with increased backscatter due to wind and current effects;
- Plains and farmlands, with homogeneous patches representing the fields;
- Mine areas, with peculiar scatters in the presence of the mine itself.

The two subsets are disjointed, i.e., there is no speckled SAR image whose corresponding speckle-free Mean is in the clean subset and vice-versa. In total, we gathered eleven Means and eight speckled SAR images. The validation dataset is, instead, composed of two images only. This dataset comprises a SAR image and a speckle-free Mean computed over the temporal stack of this image. We made this choice to have better visual feedback when evaluating the performance of our experiments.

We built the dataset using images of different scenarios to cover all the possible speckle distributions. Examples of These images are shown in Figure 5.1 and Figure 5.2. The dataset is composed of patch extracted from These images using a sliding window. Given a patch size of $W \times H$, we use a sliding factor of $\frac{W}{2} \times \frac{H}{2}$, in order to have some overlapping over adjacent patches. Our final model uses 128×128 patches extracted from high dimension SAR images and Means using a 64×64 sliding window. By doing so, we obtain 106204 speckled patches and 305844 clean ones on the training dataset, plus 900 and 900 patches for validation.

We are supposing a multiplicative speckle model

$$y = nx \,, \tag{5.1}$$

where y is the speckled image, x the corresponding clean one and n the speckle. In particular, we trained our network by feeding them with the log transform of the amplitude images, in order to make this multiplicative speckle additive

$$\log(y) = \log(n) + \log(x) \, . \tag{5.2}$$

We train our network for letting them output the speckle $\log(n)$, both for speckle generation and despeckling path.

From now on, the distribution of the log-transformed clean patches will be addressed as $p(x)$, the distribution of the noisy ones as $p(y)$ and the conditioned distribution of the speckle as $p(n|x)$ and $p(n|y)$, for the speckle generation and despeckling path respectively. Also, Y will be the log-transformed speckled SAR images and X the log-transformed means.

5.2 EXPERIMENTS

The solution to the SAR despeckling problem proposed in this work of thesis results from a well-designed suite of experiments. In particular, we tackled the problem during the thesis, starting from more straightforward approaches. Then, we built the proposed solution step by step based on the results we obtained during the experiments. We started by building a Speckle Generator network capable of generating speckle samples from a clean image. We then moved to an image-to-image translation model, as proposed in [26], introducing a Despeckler for forcing our Speckle Generator to produce samples that are more correlated with the input. Finally, we deeper explored the speckle generation task by improving our Speckle Generator to produce multiple speckle samples from the same input patch, conditioning these outputs over a Gaussian vector.

5.2.1 *First experiments on speckle generation*

We designed our first model to learn a speckle generation function to obtain a speckle realization from a clean input SAR image. In particular, we want our network to learn how to generate a speckle whose distribution is as close as possible to the real one. We considered a simpler dataset composed of a subset of these images presented in the previous section during this earlier stage. More precisely, the validation dataset comprises two images that are uncorrelated to each other, i.e., the validation Mean is extracted from a temporal stack representing a coast area, and the validation speckled image represents a mine area. At the same time, the training dataset does not contain the many different surfaces considered on the final dataset.

Due to the known limitations of the Generative Adversarial Networks, we decided to start our exploration by training this model with the algorithm

proposed in [23] that showed a more stable training process. Considering $x \sim p(x)$ a clean logarithmic patch and $y \sim p(y)$ a speckled logarithmic patch, G is the Generator network that takes as input a logarithmic clean patch and outputs a residual speckle for calculating the generated logarithmic speckled patch with Equation 5.2, and D the Critic that we train using the Wasserstein GAN adversarial loss introduced in Equation 2.2:

$$\mathcal{L}_{wgan} = \max_{G} \min_{D} \mathbb{E}_{x \sim p(x)}[D(x + G(x))] - \mathbb{E}_{y \sim p(y)}[D(y)] \, .$$

We see, from this formula, that the Critic network aims at giving a score for each input, giving higher scores to authentic SAR images and lower to fake ones. At the same time, the Generator network is trained in order to fool the Critic and make it giving higher scores to the generated images. Thus, we expect that this loss should be around zero when the G network is trained to optimality. We train our Speckle Generator using the same optimizer, learning rate, weight clipping limit, and Critic iterations used in [23].

The Speckle Generator architecture is shown in Figure 5.3. The Speckle Generator has a *U-Net*-like architecture [13], with 4-steps down-sampling and up-sampling paths connected via skip connections that channel-wise concatenate down-sampling outputs to the corresponding up-sampling step. Each step is composed of two 3×3 convolutions, each followed by batch normalization and a ReLU activation function and composed by twice the number of filters of the previous step, starting from 32. A 2×2 max-pooling layer performs the down-sampling operation. After considering using bilinear interpolation on the up-sampling side, we decide to perform the up-sample operation via a 2×2 transpose convolution with a stride of 2. Finally, a 1×1 convolution outputs is used for transforming the 32 channels of the last step into a one channel output patch.

The Critics shown in Figure 5.4 are simple CNN, composed by 5 convolutions followed by a ReLU activation functions, and 5 2×2 max-pooling layers. The last step is then flattened and, via a one-layer MLP, mapped to the $\mathbb{R}$ set of real numbers.

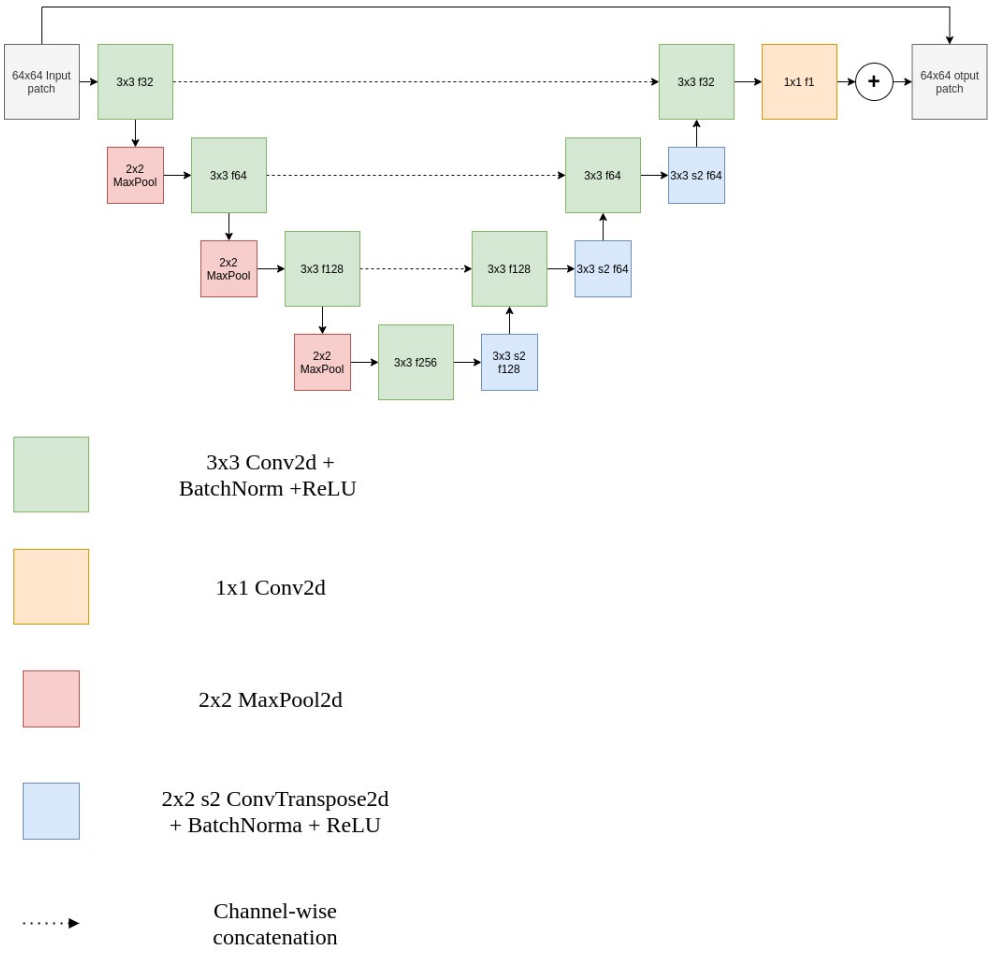


Figure 5.3: Speckle Generator architecture diagram. The Speckle Generator has a `UNet`-like [13] architecture, composed of 3 down-sampling and up-sampling steps connected via channel-wise concatenations. The upsampling is performed using 2×2 transpose convolutions with a stride of 2. The final layer of the network is a 1×1 convolution for mapping the 32 filters of the last up-sampling layer into the single channel of a SAR image. The network output represents the logarithm speckle generated for corrupting the input logarithmic clean patch, and the final generated patch is obtained using Equation 5.2.

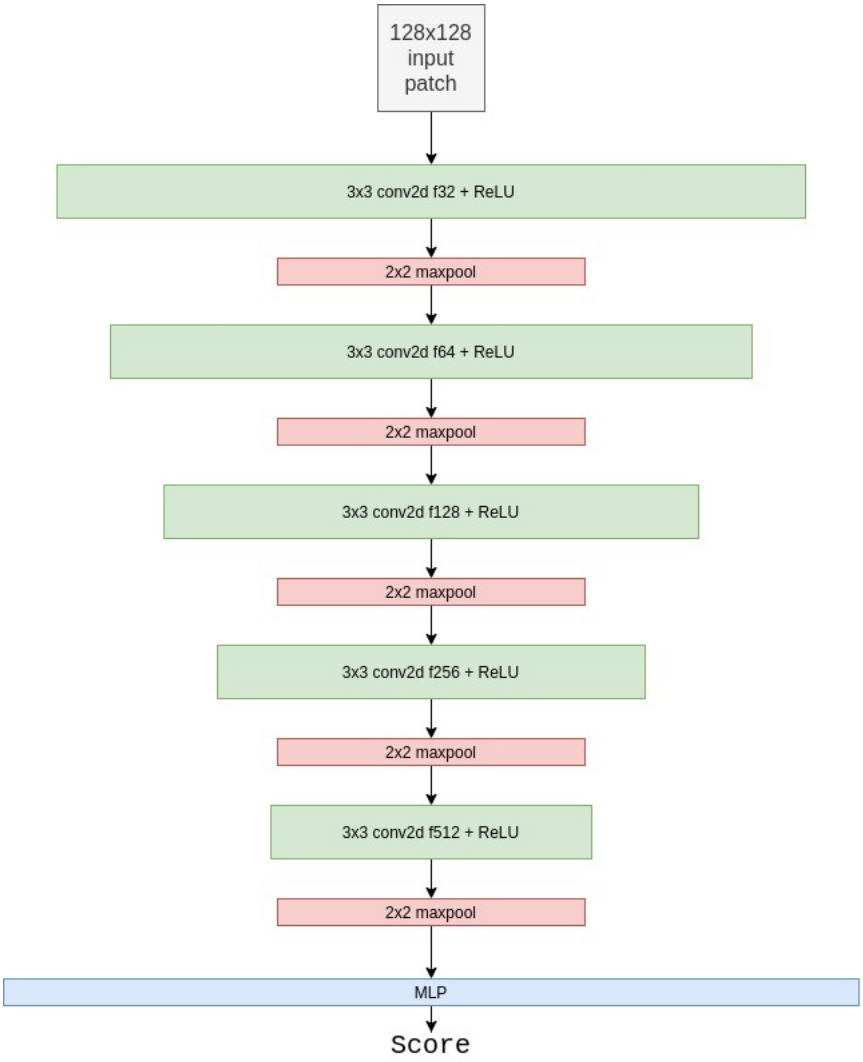


Figure 5.4: Critic network architecture. The Critic network is a simple CNN composed by a series of convolutional layers with a kernel of 3×3 and 2×2 max pooling layers. The output is computed by flattening the last down-sampling layer and using a MLP for mapping these features into a real number, representing the score of the input patch.

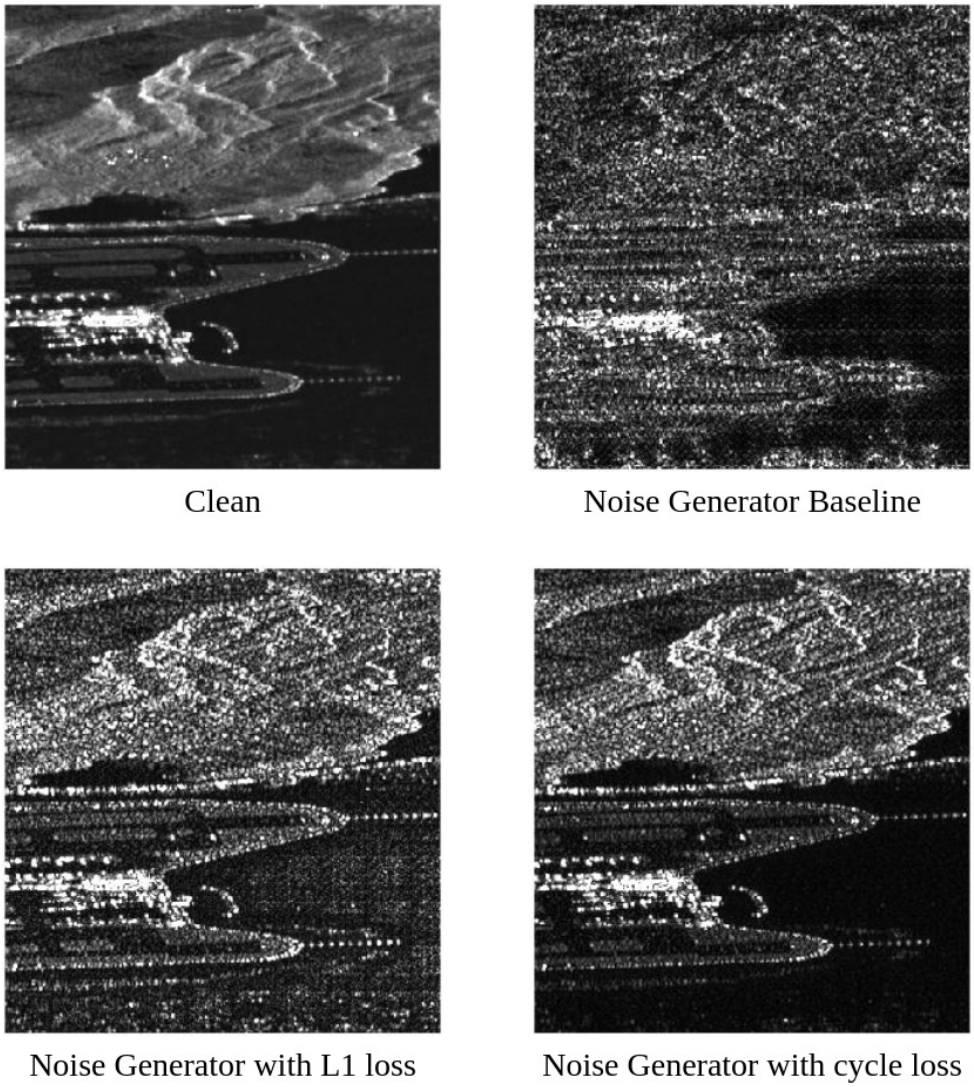


Figure 5.5: A comparison between the speckle generated by our first Speckle Generator networks and the CycleWGAN model. The first image shows the underlying scene of the clean reference image. The second image shows the speckle generated over that area by our first Generator trained using only the adversarial loss proposed in [23]. The third image shows the detail and edge improvement by adding an l1 loss term between the input and the output patch. The fourth image shows how using the cycle consistency and the Despeckler network, instead of the l1 loss function, improves speckle generation performance over homogeneous areas.

A visual example of the results obtained by this model is shown in Figure 5.5. In this image we can see a comparison between the underlying clean image, on the top-left corner, and the speckle generated by this model, on the top-right corner. The speckle generated by this model results loosely correlated with the underlying image, in particular it does not preserve the details and edges of the original image. Thus, as a first solution, we propose

to add an ℓ_1 loss term between the input and the corresponding output for constraining the action of the generated speckle:

$$\mathcal{L}_{\ell_1} = \mathbb{E}_{x\sim p(x)}[\|G(x) + x - x\|_1] = \mathbb{E}_{x\sim p(x)}[\|G(x)\|_1] \,. \tag{5.3}$$

This loss is properly weighted, in order to allow the Generator to learn how to keep some similarity between input and output without degenerating into the identity function. The total loss become

$$\mathcal{L} = \mathcal{L}_{wgan} + \lambda_{\ell_1}\mathcal{L}_{\ell_1} \,,$$

where $\lambda_{\ell_1} = 0.1$. The results of this network are compared with those of the previous experiment in Figure 5.5. The bottom-left image shows the results achieved by including the l1 loss. We can see how the speckle changes around the edges and in more homogeneous areas. However, the quality of the generated speckle is still cheap. In particular, looking at the darker homogeneous area, the performances of the network seem to be dropped. For further improving the Speckle Generator performances, we pair it with a new WGAN, the Despeckler, and train them concurrently using the cycle consistency loss of [21]. We give a more detailed description of this model, which we call CycleWGAN, in the next section. The cycle consistency term of the loss introduced in Equation 2.4 forces the Speckle Generator to preserve better the information contained in the original image so that the Despeckler can retrieve the original clean patch from the speckle corrupted one. By doing so, our model achieves higher quality results than the ℓ_1 loss. In the bottom-right image of Figure 5.5 we can see how the CycleWGAN resolves the problem showed over heterogeneous areas while still preserving contrast and edges as in previous models.

5.2.2 *CycleWGAN Model*

To allow the network to learn significant mappings between the clean and speckled domains, i.e., we want both the Despeckler and the Speckle Generator to learn how to produce the corresponding image of the target domain, we train two WGANs using the framework described in [26].

For applying the solution proposed by Zhu et al. to our problem, we define a Speckle Generator network G_n and a Despeckler G_d and two Critics D_n and D_d. The G_n Speckle Generator takes as input a patch x coming from the clean dataset X and outputs a speckle n, for then computing the generated noisy patch $\hat{y}$ as using Equation 5.2

$$\hat{y} = n + x \,,$$

$$n = G_n(x) \,.$$

The G_d network takes as input a 64×64 patch y coming from the speckled dataset Y and outputs a speckle n, for then computing the despeckled patch $\hat{x}$ using the 5.2

$$\hat{x} = y - n \,,$$
$$n = G_d(y) \,.$$

Then the D_n discriminator must distinguish y patches sampled from $p(y)$ from the generated $\hat{y}$, while D_d the x sampled from $p(x)$ from $\hat{x}$. In this context, the WGAN adversarial losses of Equation 2.2 used for training these networks are:

$$\mathcal{L}_{wgan,\,n} = \max_{G_n} \min_{D_n} \mathbb{E}_{x\sim p(x)}[D_n(\hat{y})] - \mathbb{E}_{y\sim p(y)}[D_n(y)] \,,$$

$$\mathcal{L}_{wgan,\,d} = \max_{G_d} \min_{D_d} \mathbb{E}_{y\sim p(y)}[D_d(\hat{x})] - \mathbb{E}_{x\sim p(x)}[D_d(x)] \,.$$

The Generator and Critic networks have the same architecture described in the previous section. For applying the cycle consistency term of Equation 2.4, we had to compute first the reconstructions of both the noisy and the clean images, the inputs of G_d and G_n respectively. We define these reconstructions as

$$x_r = \hat{y} - G_d(\hat{y}) \qquad y_r = \hat{x} + G_n(\hat{x}) \,.$$

The cycle consistency term of 2.4 then becomes

$$\mathcal{L}_{cycle} = \mathbb{E}_{x\sim p(x)}[\|x_r - x\|_1] + \mathbb{E}_{y\sim p(y)}[\|y_r - y\|_1] \,.$$

The complete loss used for training this model is

$$\mathcal{L} = \mathcal{L}_{wgan,\,n} + \mathcal{L}_{wgan,\,d} + \lambda_{cycle}\mathcal{L}_{cycle} \,.$$

Even though the WGANs proposed by Arjovsky, Chintala, and Bottou result to have improved training stability than classical GANs [21], WGAN training still has some limitations as showed in [25]. In fact, putting $\lambda_{cycle} = 10$ as proposed in [26] results in vanishing gradient. In this scenario, the Critic is optimal in distinguishing between generated and sampled images, but the Generators cannot learn anything. This problem is mainly due to the weight clipping, which is not the best way to force the 1-Lipschitz constraint for Critic networks. After trying with different values of λ_{cycle}, from 1 up to 10, it turns out that the best solution is to first train for 20 epochs the two generators independently, i.e., $\lambda_{cycle} = 0$, for then linearly increase the cycle-loss weight up to 5 for the next 30 epochs. The training ends by letting $\lambda_{cycle} = 5$ for the remaining epochs. The results of this network, with input patches of 64×64 are shown in Figure 5.6 and Figure 5.7.

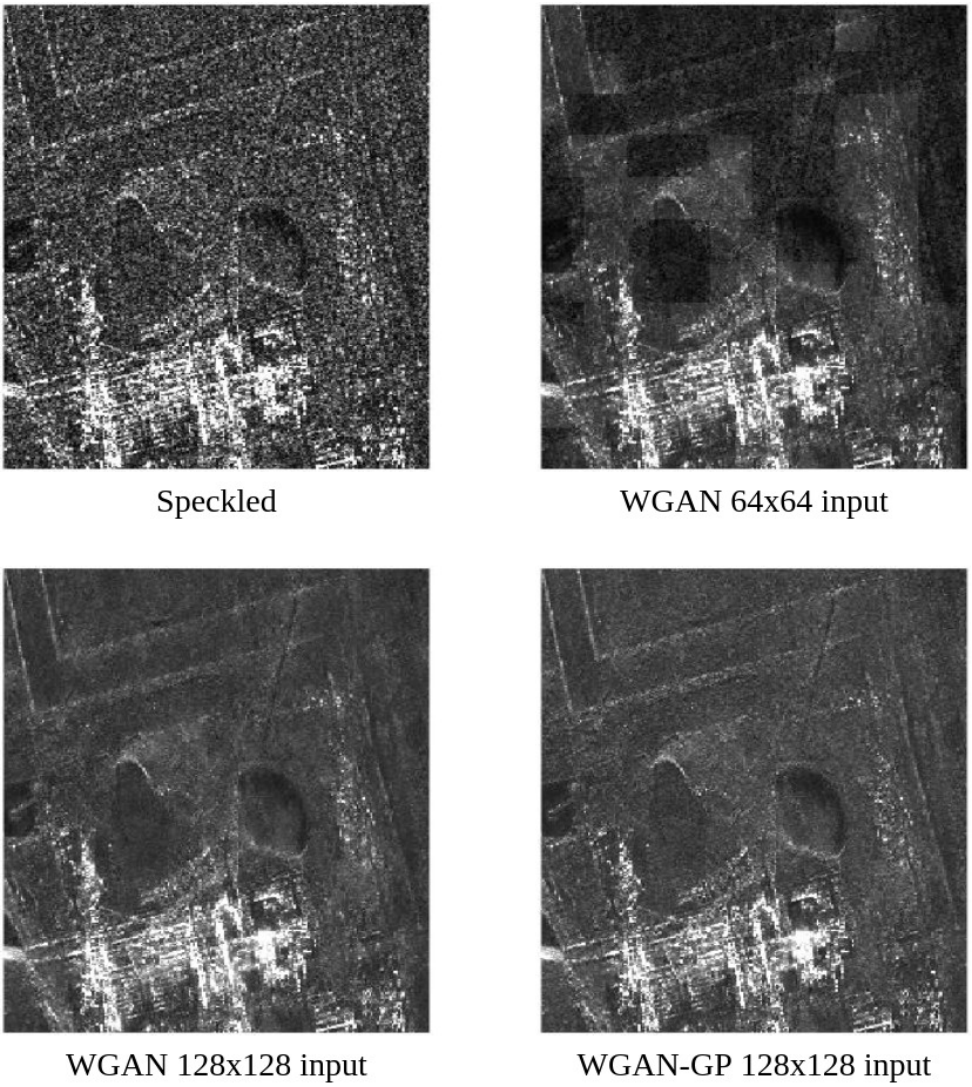


Figure 5.6: Comparison of different cycle consistent Despeckling network performances. The first image (top-left) shows the speckle corrupted SAR image. The second image (top-right) shows the despeckled patch obtained using the CycleWGAN model with 64×64 input patches. The third image (bottom-left) shows the results obtained enlarging the input patches up to 128×128. We can see how this network removes the patching problem presented in the previous one. The fourth image (bottom-right) shows the results obtained using a GP [25] loss term instead of the weight clipping for forcing the 1-Lipschitz constraint for the Critic networks. We can see how the network has similar performance concerning the previous one, but we kept exploring this model due to the faster and more reliable training.

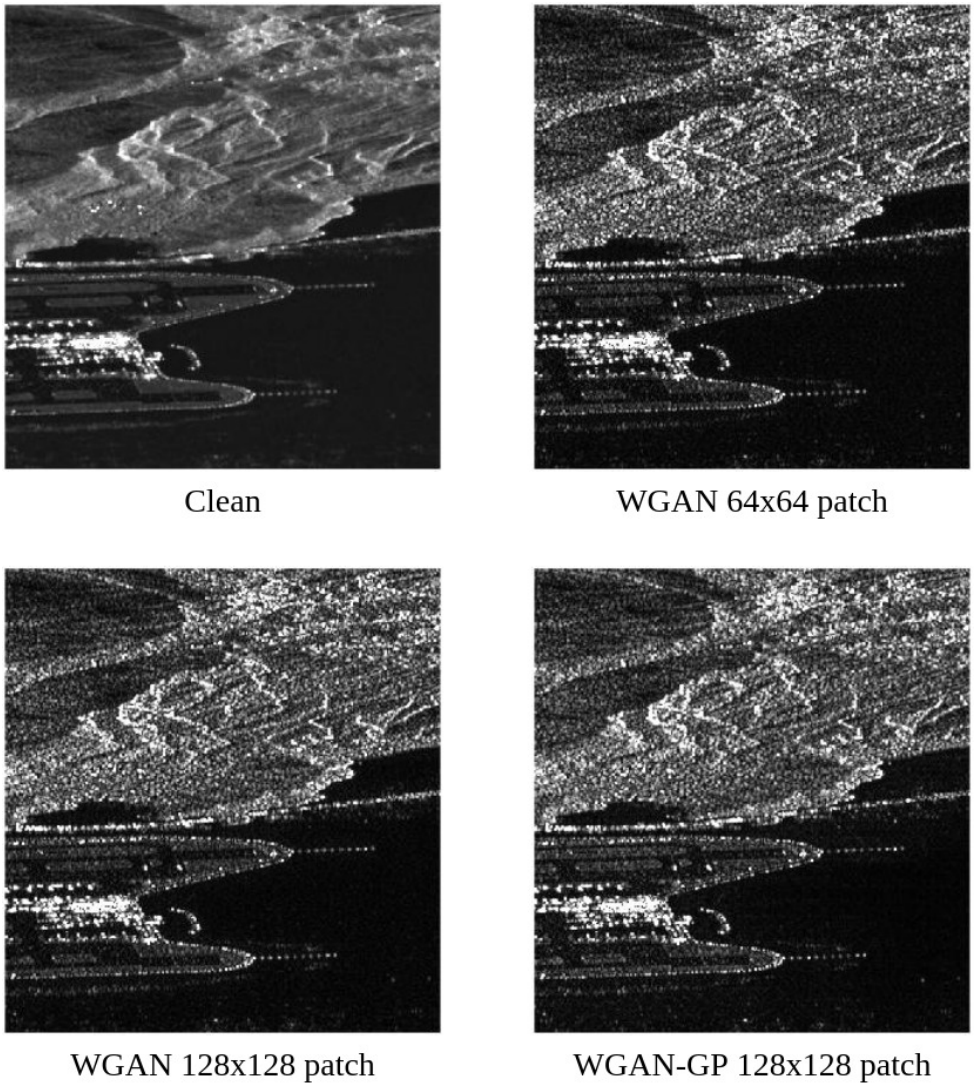


Figure 5.7: Comparison of different cycle consistent Speckle Generator network performances. The first image (top-left) shows the clean image. The second image (top-right) shows the speckle generated by a 64×64 patch CycleWGAN model. The third image (bottom-left) shows the speckle generated by a 128×128 patch CycleWGAN. The fourth image (bottom-right) shows the speckle generated by a 128×128 patch CycleWGAN trained using a GP [25] loss term instead of the weight clipping for forcing the 1-Lipschitz constraint for the Critic networks. We can see any major improvement on the speckle generation side over these models.

We can see a comparison between the different CycleWGAN models we developed in these figures. On the top row of Figure 5.6, we can see a SAR image, on the left, with the corresponding one despeckled by this network, on the right. On the top row of Figure 5.7, we can see the comparison between a clean SAR image, on the left, with the speckle corrupted generated one, on the right. We already showed, in the previous section, how the Speckle

Clean image

Speckle generated

Figure 5.8: Speckle distributions over different areas generated by 64x64 patches WGAN. The top image shows the whole Mean used for validation, while the bottom one shows the speckle generated by the Speckle Generator network over that image. We can see how the speckle generated have different distribution according to the underlying scene. In particular, we see in this image a contrast between the speckle generated over a homogeneous dark area and that generated over a brighter area made of mountains and some buildings.

Generator of the CycleWGAN model showed significant improvements from the previous experiment that used the ℓ_1 loss of Equation 5.3 for constraining the output of the WGAN Generator. Figure 5.8 shows the full validation clean image used so far and the corresponding speckle generated by our network, i.e., the term n of the Equation 5.1. We can see that our network learned how to generate a realistic speckle distribution according to the underlying scene. The result of the despeckling network, shown in Figure 5.6, are also auspicious. The network learned how to reduce the speckle while at the same time preserving edges and details. Still, this network has wide room for improvement. In particular, we can see how there is still some residual speckle over all the image and a juxtaposition of brighter and darker patches where we expected a more homogeneous tone.

At this point, we enlarged the patches, increasing the size up to 128×128 for letting the network get more context of the scene. The results of the network trained with extended patches are shown in Figure 5.6 and Figure 5.7. On the bottom-left image of these Figures, we can see the Despeckler and the Speckle Generator results, respectively, trained using a CycleWGAN with input patches of 128×128. While the Speckle Generator does not seem to have any significant change in the quality of the generated speckle, the Despeckler has shown an outstanding improvement. For example, we can see in Figure 5.6 how the image no longer has that bright-patch effect shown

in the previous experiment. Instead, it results smoother, with less residual speckle and better-defined edges and details.

After these experiments, we looked forward to an improvement in the training of our WGANs. In particular, the Critic networks introduced in [23] suffer from vanishing or exploding gradient problems depending on the choice of the weight clipping limit, as shown in [25]. Thus, instead of performing a tuning over this hyper-parameter, we decided to implement the solution proposed in this paper by adding a Gradient Penalty (GP) term to our WGAN loss. Let us define $\mathcal{L}_{\text{wgan-gp},n}$ and $\mathcal{L}_{\text{wgan-gp},d}$ the losses introduced in Equation 2.3 applied to (G_n, D_n) and (G_d, D_d) respectively, we define a new loss for our model:

$$\mathcal{L} = \mathcal{L}_{\text{wgan-gp},n} + \mathcal{L}_{\text{wgan-gp},d} + \lambda_{cycle}\mathcal{L}_{cycle} \, .$$

For using the WGAN-GP training, in [25] the authors specify to remove the Batch Normalization layers from the Critic networks. In fact, instead of mapping a single input to a single output, Batch Normalization maps from an entire batch of inputs to a batch of outputs. At the same time, penalized training objective wants to penalize the norm of the Critic gradient for each input independently.

We refer to this model as CycleWGAN-GP. We use the same optimizer, learning rate and λ_{gp} defined in [25] and showed in Figure 2.3. With the improved stability achieved thanks to the GP, we can set $\lambda_{cycle} = 10$ since the beginning of the training, reaching similar results to the Weight Clipping network but with faster training. The results of this network are shown alongside those of the previous ones in Figure 5.6 and Figure 5.7, on the bottom-right images. Overall, the new training did not significantly change the performance of our networks.

Nevertheless, looking at the training metrics of the two networks compared in Figure 5.9 we can see that we can achieve these similar results with fewer training iterations. This Figure shows the graphs representing the Speckle Critic loss (top-left), the Clean Critic loss (bottom-left) and the cycle loss (top-right). These graphs show how the number of iterations of the CycleWGAN-GP model is lower than what is needed for training the CycleWGAN one for obtaining similar performances. Also, looking at the speckled Critic (D_n) score, we can see that this decreases after some epochs when using a GP term. In 2.3, the authors show how this behaviour is caused by the Speckle Generator G_n not being complex enough for learning how to generate speckle indistinguishable from that of authentic SAR images, leading to the Critic over-fitting.

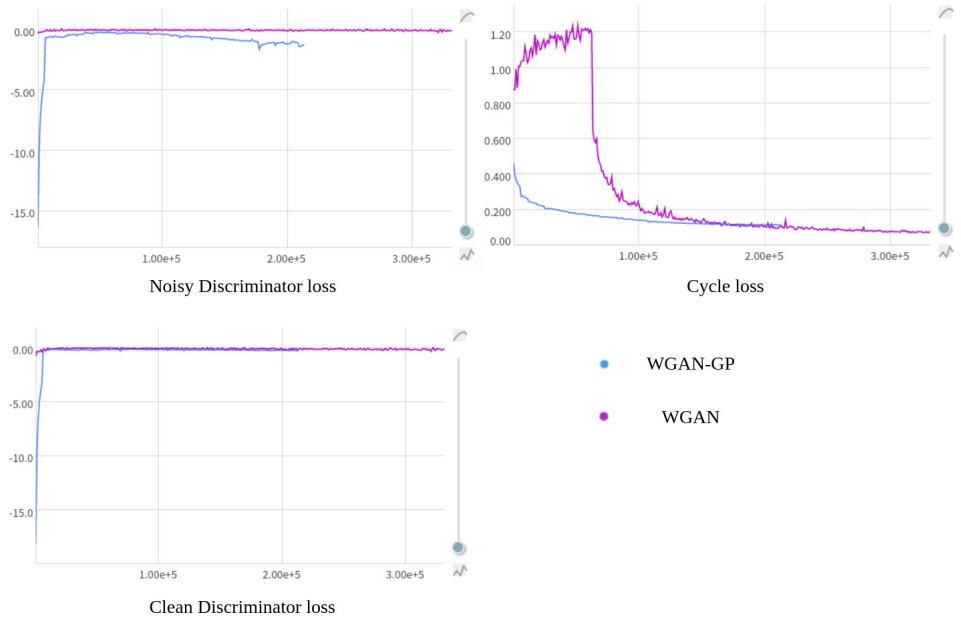


Figure 5.9: Comparison between Weight Clipping and GP network training with 128×128 patches. The x-axis of these graphs represents the number of training steps. The first image (top-left) shows the speckle Critic loss graph, the second one (top-right) shows the cycle consistency loss, and the third one (bottom-left) shows the Clean Critic loss. Looking at the lengths of these graphs, we can see how the training of the WGAN-GP model has been stopped before that of the WGAN one, achieving similar speckle generation and despeckling performances as shown in Figure 5.6 and Figure 5.7. The cycle loss graph (top-right) shows how we could set $\lambda_{cycle} = 10$ since the first epochs. Also, the Speckle Critic losses over time (top-left) differ significantly. In particular, we can see how the WGAN-GP model one decreases after some epochs, showing how the speckle Generator network is not complex enough to learn a good speckle generation function.

5.2.3 *Dataset augmentation and architecture tests*

After completing the development of the CycleWGAN-GP model, new data became available. Thus, we decided to perform a dataset refactoring. We included what has been so far our validation dataset into the training dataset and use two corresponding speckled and clean images for validating our model. We decided to extract the speckled validation image from a temporal stack, whose mean is the same validation clean image. We did that to have better visual feedback on the performance of our model. Recalling the analysis made looking at the training graph shown in Figure 5.9, we also decided to improve the architecture of both the speckle Generator and the Despeckler. We introduced one more down-sampling and up-sampling step and doubled the number of convolutions for each of these steps from the architecture

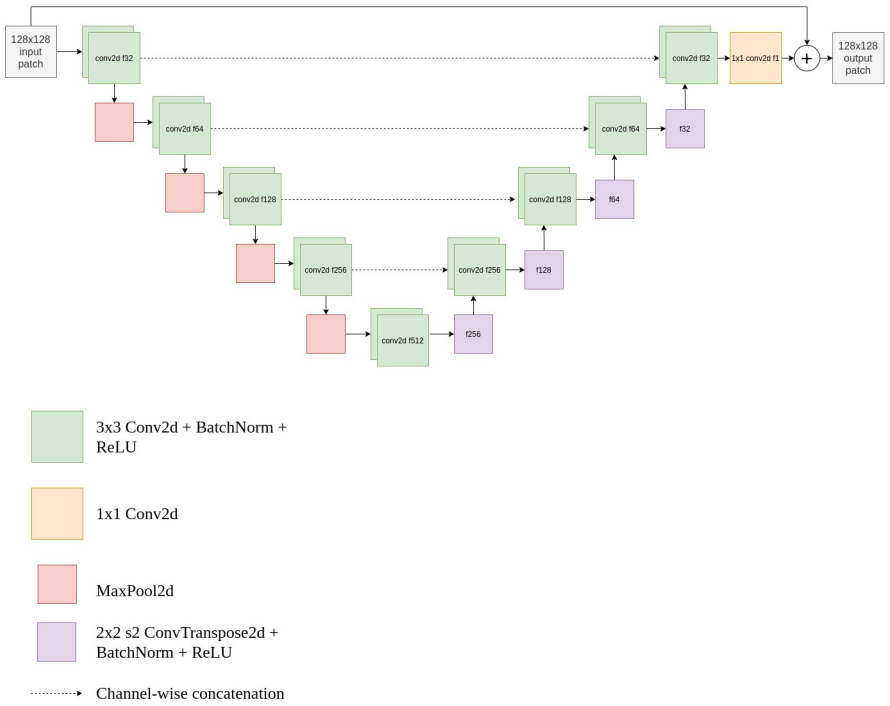


Figure 5.10: Speckle Generator and Despeckler architecture trained using the augmented dataset. Differently from the architecture shown in Figure 5.3, the new one has one more down-sampling and up-sampling step and the number of convolutions for each step has been increased to 2, in order to increase the receptive field of each pixel.

shown in Figure 5.3 to increase the receptive field. The new architecture is shown in Figure 5.10.

We also tried different version of the cycle consistency loss of Equation 2.4. One possible option is to calculate the reconstruction error on the logarithmic domain, i.e., given $x \in X$ logarithm of a clean patch, $\tilde{y} = G_n(x) + x$ a speckle generated over x and $\hat{x} = \tilde{y} - G_d(\tilde{y})$ its reconstruction we rewrite the Equation 2.4:

$$\mathcal{L}_{cycle,x} = \mathbb{E}_{x\sim p(x)}[\|x - \hat{x}\|_1] \, .$$

Another option is instead to compute that loss on the original amplitude domain, so that the cycle loss over x becomes

$$\mathcal{L}_{cycle,x} = \mathbb{E}_{x\sim p(x)}[\|\exp(x) - \exp(\hat{x})\|_1] \, .$$

The rationale behind this is that we have higher values for errors over low amplitude pixels on a logarithm scale. In contrast, the logarithm function tends to be almost flat for higher values, thus penalizing fewer errors over high amplitude pixels. On the other hand, the cycle loss over the actual amplitude domain penalizes errors linearly across the whole range of possible

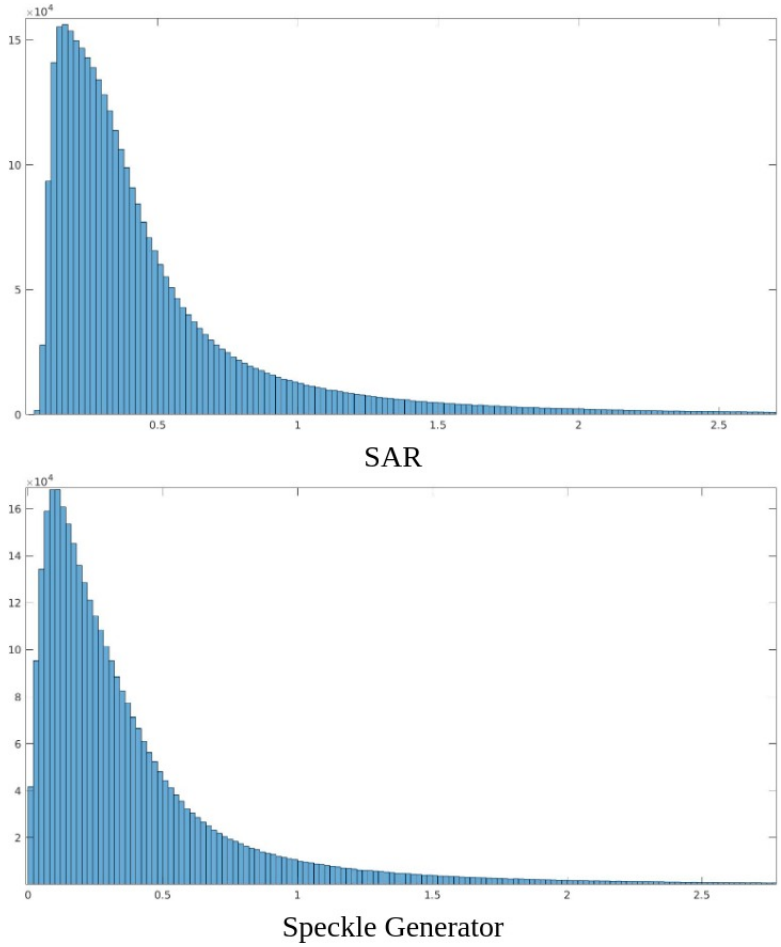
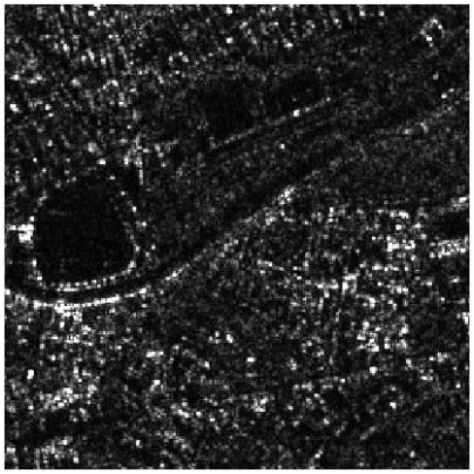
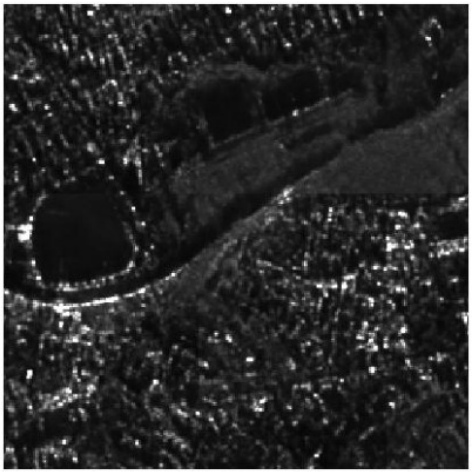


Figure 5.11: Histogram of pixels amplitude of real and generated speckled images. The first row shows the histogram of a real SAR image. The second row shows the histogram of an image generated by a 128×128 patch WGAN-GP Speckle Generator trained using the logarithmic cycle loss. The two histograms represent similar curves, but the generated images tends to have higher low amplitude pixel concentration with respect to the actual one.



Speckled

Logarithm domain



Original domain

Figure 5.12: Comparison of despeckling performances calculating cycle-consistency loss over different domains. The image on top shows the SAR speckled image. On the bottom, we can see the results of the Despeckler trained using the logarithmic cycle loss (left) compared to that trained using the cycle loss over the real SAR domain (right). We can see how the image on the right is smoother and more homogeneous but loses some detail, especially on high amplitude punctual scatterers, concerning the left one, presenting some residual speckle.

values. We can analyze the amplitude values of a SAR image by looking at the histogram on the top row of Figure 5.11. Here we can see how a per-pixel distance over the logarithm domain may be optimal for computing the cycle loss over the pixels, composing most of the image and details, whose value is relatively small. Using the actual amplitude domain helps preserve the scale of higher intensity pixels, like strong scatterers that are outliers for the distribution shown in the histogram of Figure 5.11. The results of two networks trained using these losses are shown in Figure 5.12. We can see how computing the cycle loss on the original domain results in smoother surfaces with low residual speckle but losing some information over punctual scatterers, particularly those whose amplitude value is not significantly higher than the average pixel value. The log domain loss performs better in preserving scatterers information but presents a residual speckle. In particular, it exaggerates the amplitude of bright spots that may not be considered strong scatterers. These differences show how the two networks store the information necessary to reconstruct the original speckled image, necessary for minimizing the $\mathcal{L}_{cycle}$ term. Calculating cycle consistency loss over the original domain allows the network to store this information as a low amplitude speckle, having a low impact on the cycle loss term. Using the log domain allows the network to store this information on higher amplitude pixels.

Overall, we achieved satisfactory performances on the despeckling side. Figure 5.13 shows a comparison between our model and both the `SAR-BM3D` algorithm proposed in [7] and the `UNet` of Lattari et al. [11]. In this Figure, the first image represents the SAR image on which we performed the despeckling (image 1) and the Mean computed by the multi-temporal algorithm proposed by Ferretti et al. [9] over the stack from which we extracted the speckled image as Ground-Truth (image 2). The `SAR-BM3D` algorithm (image 3) tends to be blurry, especially over high contrast edges and introduces some artefacts over homogeneous areas. The `UNet` of [11] (image 4), instead, is sharper over areas with a high density of high amplitude scatterers, i.e., buildings, and in preserving the punctual isolated ones, but still heavily introduces artefacts over the homogeneous areas. The improved version with TV loss of this network (image 5) partially solves the artefacts problem. It improves the definition of the heavy scatterers, but it over-smooths homogeneous areas, losing many details. Our CycleWGAN-GP model trained using 128×128 and logarithm domain cycle loss (image 6) does an excellent job preserving details over both high scatterers concentration and homogeneous areas, but it presents a residual speckle. However, comparing the ground-truth Mean (image 2) with the output of the `UNet` with the TV loss term of [11] (image 5) in Figure 5.13, we can notice some residual speckle is present in the first. Thus, due to the choice of the clean dataset, it will be impossible for us to remove it, but we aim at reducing its intensity in our following

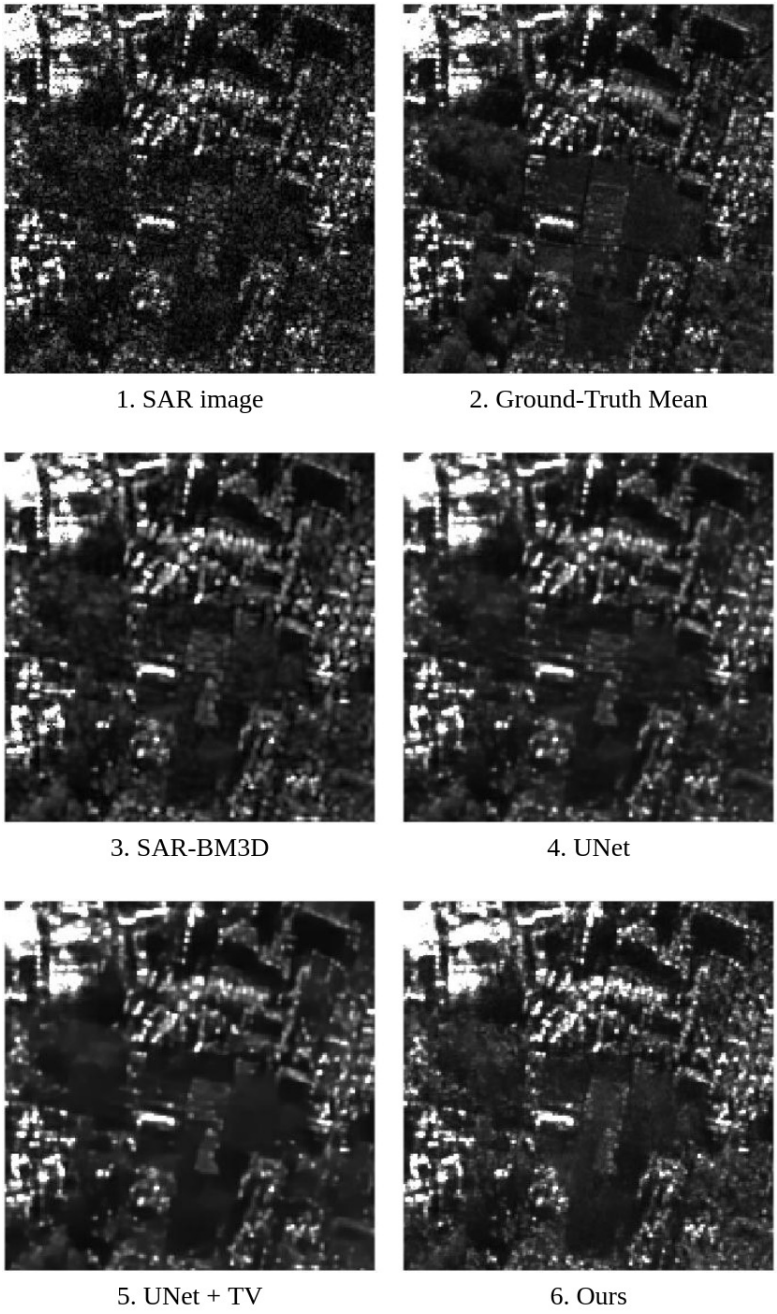


Figure 5.13: Despeckling performances of our network with respect to a Ground-Truth Mean [9] (image 2), the `SAR-BM3D` algorithm [7] (image 3) and the `UNet` architecture proposed by Lattari et al. [11] (image 4) also with the TV loss term (image 5). Our network (image 6) results in better defined edges and contrasts, with respect to the blurry images 3 and 4, and does not present any artefacts. Also it better preserves high intensity scatterers with respect to 5 but still presents some residual speckle. Comparing the ground-truth image (2) with the smoother solution proposed in [11] (5) we can see how the means we are using as ground truth also presents some residual speckle.

experiments. On the speckle generation side, Figure 5.11 shows a comparison between the histogram of an actual SAR image and one generated by ours Speckle Generator. Overall, the two curves are similar, but there is still some difference. For example, looking at Figure 5.14 we can observe how the generated speckle corrupts a mean, computed over a temporal stack with the algorithm proposed by Ferretti et al. [9], in a similar way to an actual SAR image extracted from the stack. Nonetheless, our Speckle Generator still preserves many details from the input mean.

In conclusion, we can say that we have successfully trained in an unsupervised fashion a Despeckler that reaches comparable performances with the state-of-the-art approaches on some aspects. In the following sections, we improved our model to find a Speckle Generator capable of learning, for each input patch, a speckle distribution from which we can sample one on the infinite possible speckle realizations coming from this distribution.

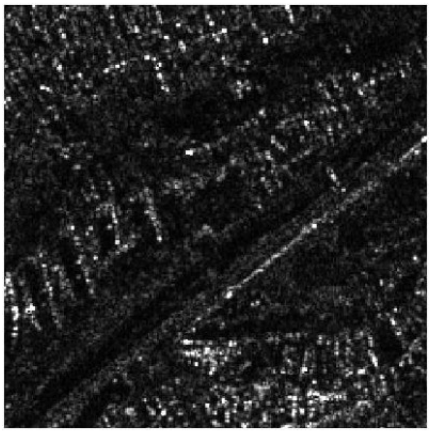
5.2.4 *Noise Embedding*

Using a simple CycleGAN [26] may restrict the learning of a function for generating the speckle. The cycle loss aims at minimizing the point-wise distance between the original x and the reconstruction f(g(x)). We may consider that the actual corresponding noisy image, given a clean patch, is not unique for our particular SAR image despeckling and speckle generation tasks. Once we can identify the speckle distribution over an area, we wanted to sample an infinite number of speckle realization for each clean patch. This problem is addressed as one-to-many mappings, and is discussed by Bashkirova, Usman, and Saenko in [27]. This paper shows how networks learning many-to-one mappings store information necessary to reconstruct the input image in the output as a low amplitude noise. The network learning the one-to-many mapping can use the information stored in this noise to minimize the cycle-consistency loss.

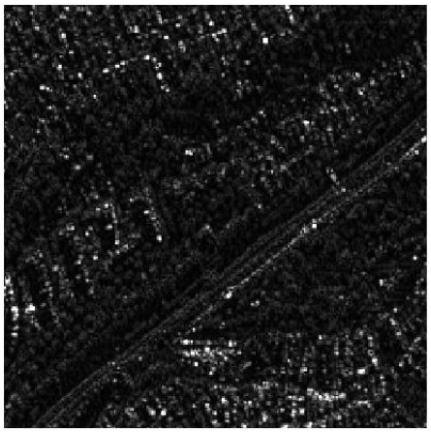
In our case, we noticed how clean images produced by the Despeckler carries a strong residual speckle, necessary for the Speckle Generator to reconstruct the same input speckled image. We want to improve our model to store this information in a separate vector. This way, we can use this vector to condition the Speckle Generator and generate different speckle realizations given the same input patch without hiding any information in the clean images. There are in literature various approaches aiming at resolving this problem but we focused on two of them in particular: one proposed by Guo et al. [31], that lays on the idea introduced by the so-called `Variational AutoEncoders` (`VAE`) [29], and one by Li et al. [32].

We defined a new Speckle Generator architecture that can take as input a clean image x and a latent vector z for conditioning the generation of the speckle. The new architecture differs from the previously introduced

Mean



SAR



Noise Generator

Figure 5.14: Comparison between a generated speckled image with a real SAR. On the top, we can see the Mean computed by the algorithm of [9] over the temporal stack to which the bottom-left real SAR image belongs. The bottom-right image shows an image generated by the 128×128 Speckle Generator of our CycleWGAN-GP model trained using the logarithmic cycle loss. The two speckle realizations look similar, even if the generated one still preserves many details of the Mean.

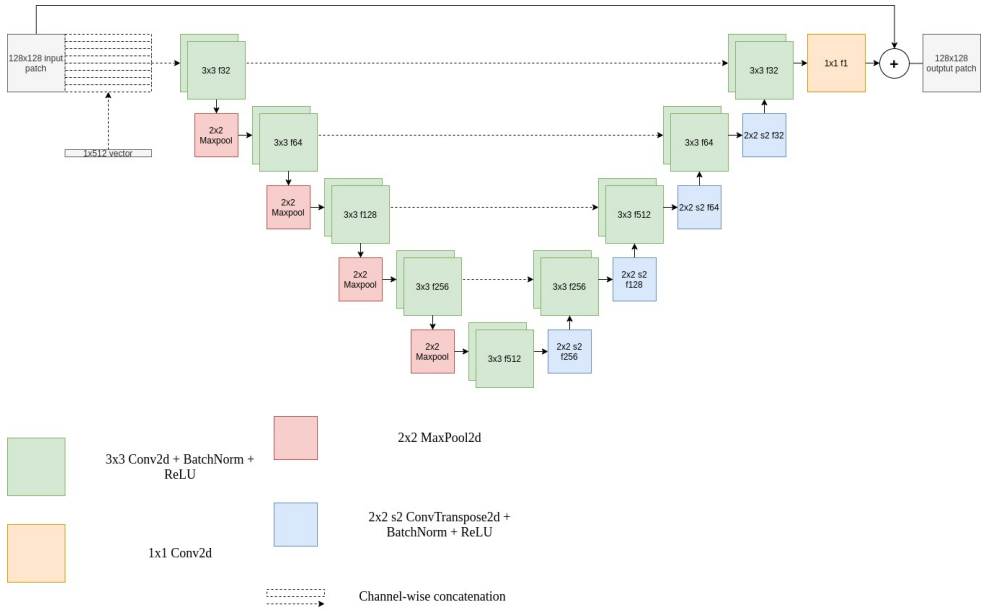


Figure 5.15: Conditioned Speckle Generator network architecture. The network differ from what proposed in Figure 5.10, used for the Despeckler, for the injection of the vector z. The 1×512 z input vector is repeated up to the shape of $512 \times 128 \times 128$ and channel-wise concatenated to the input patch.

in Figure 5.10 for letting us inject the latent vector. For doing so, we take inspiration from [33]. In this paper, Zhu et al. propose to inject z by repeating such vector $h \times w$ times for building a tensor whose shape is $d \times h \times w$, where d is the dimension of the vector and h and w are height and width of the input patches. Finally, we channel-wise concatenate this tensor with the input patch. The new Speckle Generator structure is shown in Figure 5.15.

[31] and [32] propose two different architectures for this network. A first one for training it in a `Conditional VAE`-like [30] fashion as proposed in the `CycleCVAE` [31] model, where the Encoder takes the channel-wise concatenation of a log speckle n and a clean image x, such that the corresponding logarithmic speckled one can be calculated via Equation 5.2, and outputs two vectors μ and σ, so that we can extract the z from the distribution that minimizes the reconstruction error $\|y - y_r\|_1$ using the reparameterization trick. In [31], the authors propose to feed this Encoder only with the image we are aiming at reconstructing, y. However, we decided to feed the Encoder with the channel-wise concatenation of the despeckled image x and the speckle n, which has been proved in [31] to be equivalent. Thus, feeding the network with y corresponds to using $\{x, n\}$, since they represent the same information (the speckled image y is merely the sum of the two components x and n).

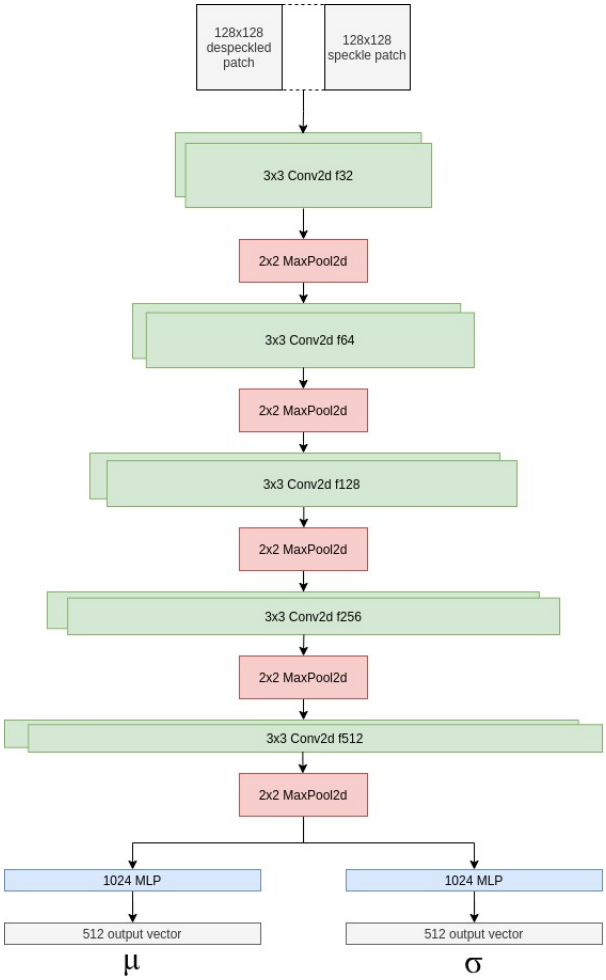


Figure 5.16: Encoder network architecture. The encoder is a CNN that takes the channel-wise concatenation of the despeckled logarithmic image $\tilde{x}$ and the logarithmic speckle $\tilde{n}$ generated by the Despeckler for computing the distribution $N(\mu, \sigma)$ from which sample the vector z for reconstructing $\hat{y} = G_n(\tilde{x}, z) + \tilde{x} \sim y = \tilde{x} + \tilde{n}$. The network comprises five encoding steps of two 3×3 convolutional layers with Batch Normalization and ReLU activation function and 2×2 MaxPooling. The output of the last down-sampling layer is flattened and fed to two MLPs. these MLPs have one hidden layer composed of 1024 units and a 0.2 dropout probability, then the output are two vectors of length 512 representing the μ and $\log \sigma$ for the reparameterization trick of [29].

A second version of the Encoder network is that proposed by Li et al. [32], where the output of this network is the vector z itself. Again, we use a new Critic (D_e) to introduce an adversarial loss to force the generated zs distribution to match the sampled ones. After analyzing both [31] and [32], we decided to implement and deeper investigate the `CycleCVAE` model. The Encoder network architecture has already been described in Chapter 4. We propose this architecture again in detail in Figure 5.16. For training our model

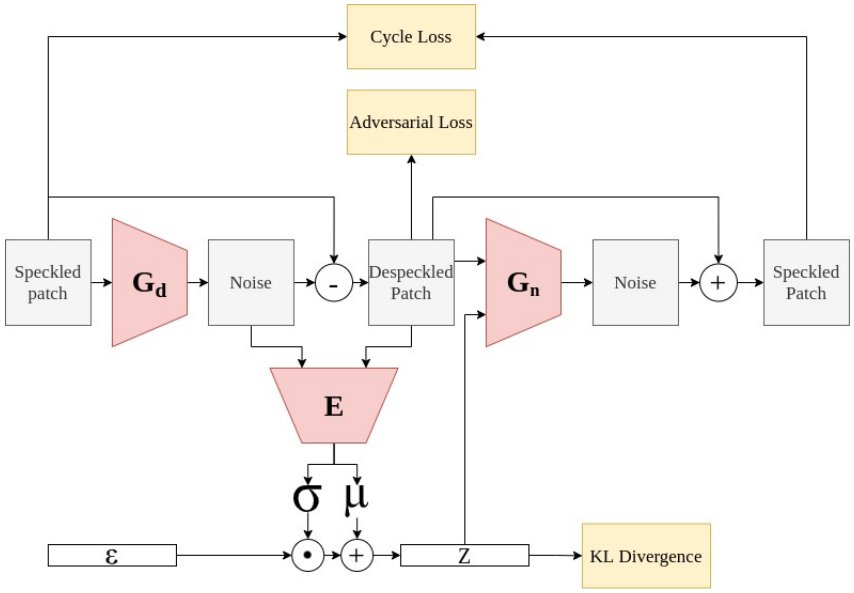


Figure 5.17: Schema of the SAR images reconstruction loop. A 128×128 speckled logarithmic patch y is taken as input by the Despeckler, which computes the speckle $\tilde{n}$ from which we can extract the despeckled logarithmic patch $\tilde{x} = y - \tilde{n}$. The adversarial WGAN loss is computed over the generated $\tilde{x}$, while the encoder network E computes the mean and the variance of the distribution of z which minimizes the reconstruction error $\mu, \sigma = E(\tilde{x}, \tilde{n})$. The $\mathbb{KL}$ divergence loss is computed over the outputs of E. these parameters are used to sample z using the reparameterization trick $z = \mu + \sigma \odot \epsilon,\ \epsilon \sim N(0, 1)$. Finally the reconstruction $\hat{y} = G_n(\tilde{x}, z)$ is generated and the cycle loss over y and $\hat{y}$ is computed.

we divide each step into two different loops: the clean loop

$$x \to \tilde{y} \to \hat{x}$$

and the noisy one

$$y \to \{\tilde{x}, \tilde{z}\} \to \hat{y}\ .$$

The training of the clean loop does not differ from what is done in the cycle-consistent baseline model. Given a clean patch x we sample a Gaussian vector $z \sim N(0, 1)$ and find the generated speckled patch $\tilde{y} = x + G_n(x, z)$ for which we compute the adversarial loss $\mathcal{L}_{wgangp,y}$. We then despeckle the generated patch finding the clean reconstruction $\hat{x} = \tilde{y} - G_d(\tilde{y})$ and computing the cycle loss $\|x - \hat{x}\|_1$. For the noisy loop, we at first despeckle the speckled patch $\tilde{x} = y - G_d(y)$ and compute the corresponding adversarial loss $\mathcal{L}_{wgangp,x}$, then sample a vector $\tilde{z}$ from the distribution that minimizes the reconstruction error. As showed in [30], this distribution is a Gaussian whose mean and logarithmic variance are calculated by the Encoder network E

$$\mu, \log(\sigma) = E(\tilde{x}, G_d(y))\ ,$$

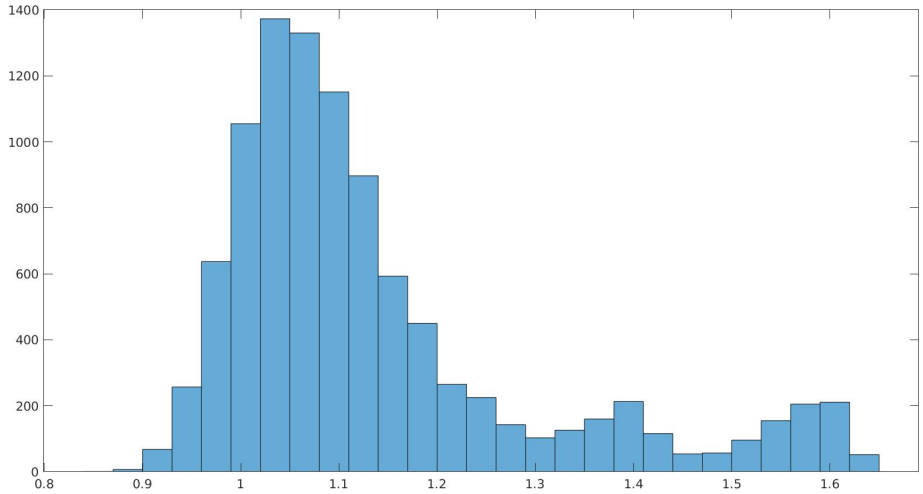


Figure 5.18: Histogram of generated speckle values obtained over the same by sampling 10000 diferent $z \sim N(0, 1)$.

then the vector $\tilde{z}$ is obtained via the reparameterization trick

$$\tilde{z} = \mu + \sigma \odot \epsilon, \ \ \epsilon \sim N(0, 1) \ .$$

Here, we compute the $\mathbb{KL}$ divergence over the Encoder distribution

$$\mathbb{KL}[E(\tilde{x}, G_d(y))|N(0, 1)] = -0.5(1 + \log(\sigma) - \mu^2 - \sigma) \ .$$

Finally, we reconstruct the original noisy patch by feeding $\tilde{x}$ and $\tilde{z}$ to the Speckle Generator $\hat{y} = \tilde{x} + G_n(\tilde{x}, \tilde{z})$ and calculate the reconstruction error $\|y - \hat{y}\|_1$. The schema of the speckled loop training is showed in Figure 5.17. The total loss we used for training this model is:

$$\mathcal{L} = \mathcal{L}_{adv,n} + \mathcal{L}_{adv,d} + \lambda_{cycle}\mathcal{L}_{cycle} + \lambda_{kl}\mathcal{L}_{kl} \ ,$$

where $\lambda_{kl} = 0.025$.

This model improved both speckle generation and despeckling performances than the simple CycleWGAN-GP. In particular, it can sample different speckle realizations accordingly to the latent input vector. We proved this by isolating a homogeneous patch from our validation dataset and performing Speckle Generation inference 10000 times using different $z \sim N(0, 1)$, then plotting all the values taken by a point with fixed coordinates. The histogram of these values is shown in Figure 5.18. We can see how different latent vectors generate different speckle realizations whose values vary from a minimum of 0.9 up to a maximum of 1.6. The histogram shows how most of these values are around one, but this distribution is not symmetric. The tail over higher values indicates that our Speckle Generator tends to raise the amplitude value of the clean analyzed pixel when generating speckled images. This

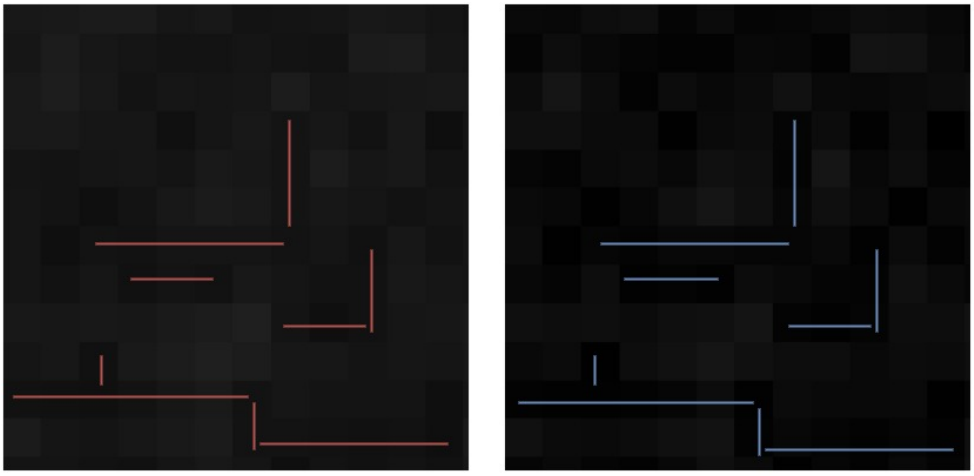


Figure 5.19: Detail of a homogeneous validation area. The image on the left represents the residual noise of a Mean. The amplitude of this image has been multiplied by a factor of 2 to emphasize this signal. The image on the right shows the speckle generated over that area. In red and blue are highlighted the patterns that show how the Speckle Generator uses this low amplitude signal during the generation of the speckle.

behaviour made us suspect that our model is still hiding some information about the speckle in the clean image. In fact, by zooming over homogeneous areas of our validation images, we can see how the generated speckle reflects some patterns of the residual noise present in the Means. These patterns show that our model is still encoding helpful information to reconstruct the speckle realization in the despeckled image itself. Observing the speckle in detail validating the results, it turned out that the speckle generated by our network is just an amplification of this low amplitude signal. Figure 5.19 shows a comparison between a detail of a homogeneous area taken from a validation Mean, whose amplitude has been doubled in value for visual purposes, on the left and the corresponding speckled generated on the right. We highlighted in red and blue the patterns of this low amplitude signal showing how our Speckle Generator amplifies this signal when generating the speckle. We only highlighted the longer lines drawn by adjacent darker pixels whose shape is identical in the two images. However, we invite the reader at taking a closer look pixel-by-pixel. It can be easily noticed that if there are elements in the clean image whose intensity is higher (or lower) than neighbours, then the speckled image presents the same behaviour over the corresponding spots. Recalling the histogram of Figure 5.18, we can confirm that our Speckle Generator learned to generate speckle by exaggerating the small amplitude fluctuations of the clean image pixels.

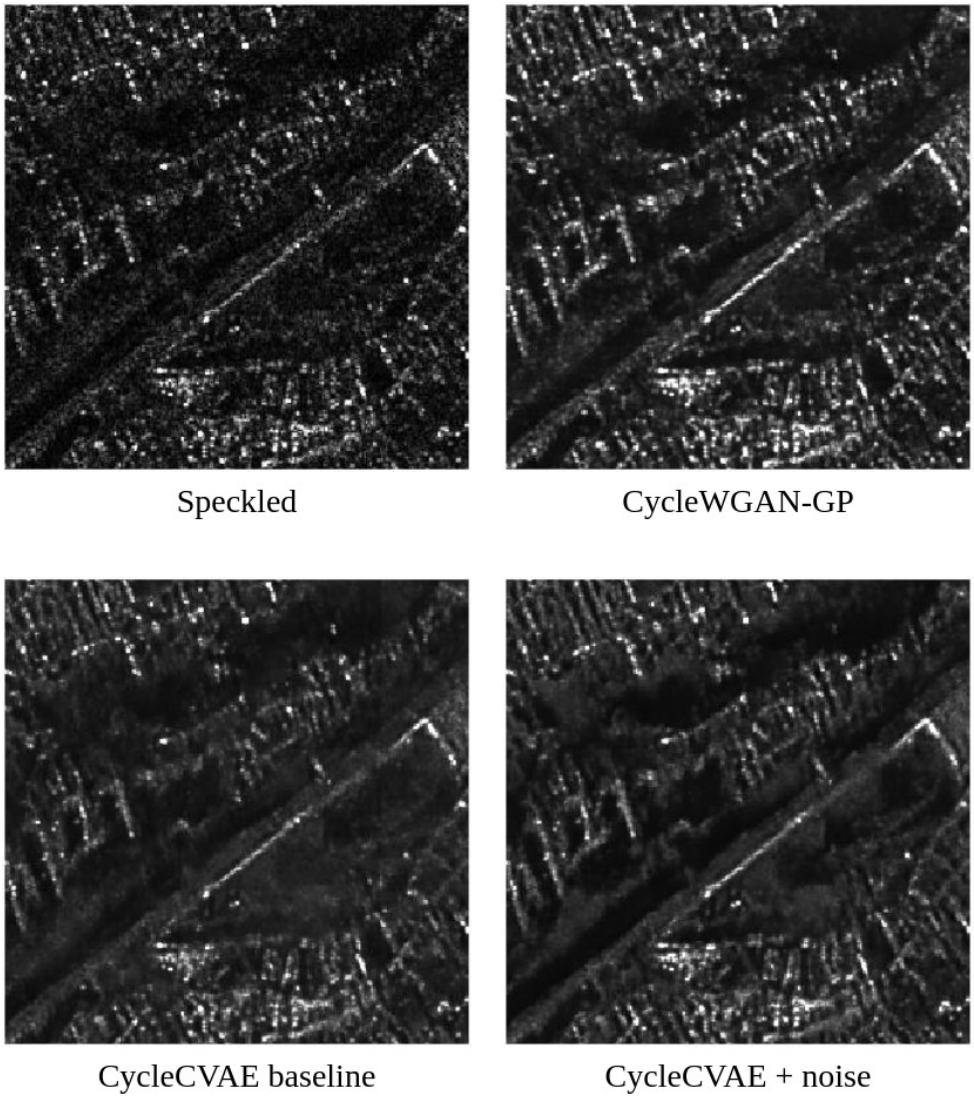


Figure 5.20: Comparison between our CycleCVAE implementations and the CycleWGAN-GP Despecklers. The top left image shows the speckled SAR image. The top right image shows the image despeckled using the Despeckler of our CycleWGAN-GP model. On the bottom row, we can see the performances of the Despecklers trained in our CycleCVAE implementation baseline (left) and using corrupted clean patches (right). The CycleCVAE baseline drastically reduces the residual noise showed by the CycleWGAN-GP model, losing detail over strong scatterers. The improved CycleCVAE implementation trained with corrupted clean patches shows an improvement in preserving edges and contrasts than the baseline.

The network outperformed the CycleWGAN-GP model on the despeckling side by drastically reducing the residual speckle. At the same time, it overall well preserved the strong scatterers but showed more blurry edges, in particular over darker areas. This comparison is shown in Figure 5.20. On the top row of this Figure, we can see the original speckled SAR image

and the corresponding despeckled using the CycleWGAN-GP model. We can see the same image despeckled by our CycleCVAE implementation on the bottom left. Examining the black diagonal that cuts these images from the bottom-left corner to approximately the top-right one, we may notice interesting differences between the two approaches. This portion of the image has better-defined edges in the CycleWGAN-GP than the CycleCVAE, which shows blurry effects in distinct darker areas, e.g., shadows on the top part of the image and black circular localities in the centre-right, below the diagonal. However, looking at the central homogeneous area of the CycleWGAN-GP despeckled image, we can see a residual noise that emphasizes the speckle pattern of the SAR image (the brighter pixels). In contrast, the CycleCVAE image shows, as expected, to overcome this problem. However, by taking a closer look, we can see this image is still preserving some information related to the presence of these spots. We already outlined how this signal is also present over the Means we use as a clean dataset at training time, making it impossible to eliminate this low-amplitude residual.

For forcing our networks to ignore this signal, we decided to apply, at training time, a low amplitude Gaussian noise to clean images before feeding them to the Speckle Generator. The amplitude of the noise is set to be small enough not to delete any detail of the original image, e.g., edges and punctual scatterers, but big enough to delete the effect of every information the Despeckler may introduce into a despeckled patch. We define this noise as

$$\eta = \gamma \times \epsilon \, ,$$

ϵ being a 128×128 patch whose values are sampled from a standard Gaussian and $\gamma = 0.05$. We then generate a corrupted patch y^1 by corrupting the original logarithmic clean patch y

$$y^1 = \log(\eta + \exp(y)) \, .$$

This operation is performed both when generating the speckled patch $\tilde{y}$, given a Mean patch x, during the clean loop and when generating the reconstruction $\hat{y}$, given the despeckled patch $\tilde{x}$, during the speckled loop.

This process improves our Despeckler, making it better to preserve edges and contrasts. We compare the two CycleCVAE implementations on the bottom row of Figure 5.20. By looking at the top part of this Figure and the black diagonal, we can notice as this model well preserves the contrasts between darker and brighter adjacent areas defining sharp edges. A high level of detail is also preserved over strong scatterers areas, while homogeneous areas, like that in the centre of the image, present a low residual speckle. Comparing this image with those produced by the previous model, we concluded that this is the best solution we obtained. On the Speckle Generation side, adding the noise η at the clean patches resulted in the generation of a speckle loosely

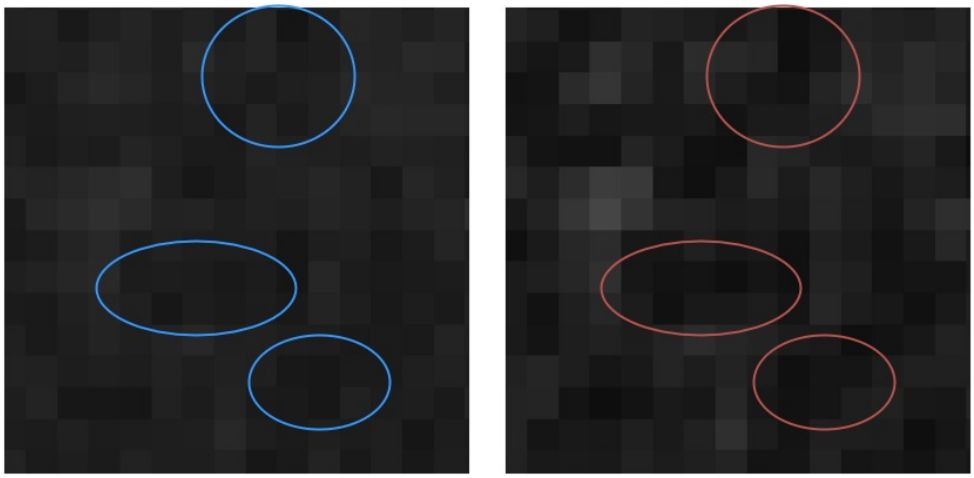


Figure 5.21: Detail of the speckle generated over a homogeneous area by the Speckle Generator trained with Gaussian noise corrupted Means. The image on the left shows a Mean patch whose amplitude. The image on the right shows the speckled one generated by feeding the Speckle Generator with the patch on the left. The amplitude of these images has been multiplied by a factor of 3 for visual purposes. We highlighted areas in which the correlation between the residual speckle and the generated one is very low. However, there are still brighter and darker pixel areas with similar shapes in both images.

correlated with the residual shown by the Means. For example, in Figure 5.21, we can see a homogeneous area extracted from a Mean (left) and the corresponding speckled generated by our Speckle Generator trained using the Gaussian noise corruption (right). We multiplied the amplitude of these images by a factor of 3 to better enhancing the fluctuations of values between adjacent pixels. We also circled in red (Mean) and blue (speckled generated) some areas in which the Speckle Generator showed to ignore the pattern present in the means. However, even in these areas, we can notice how particularly dark and bright pixels still influence the generated speckle, particularly by looking at the bright area in the left of the two images.

As for the previous model, we performed a test in which we sampled 10000 different latent vectors from a standard Gaussian and generated as many speckle realizations over one patch. The speckle intensity generated over a single pixel is showed by the histogram of Figure 5.22. As a consequence of the lower correlation between generated speckle and the residual present in the Means, these histograms show how the Speckle Generator output distribution has a broader variance, starting from 0 up to 1.6. This histogram has a more symmetric shape centred in 1, covering almost the same range of lower and higher values than 1. However, we can see how the histogram is still unbalanced on the right side, meaning that the generated speckle

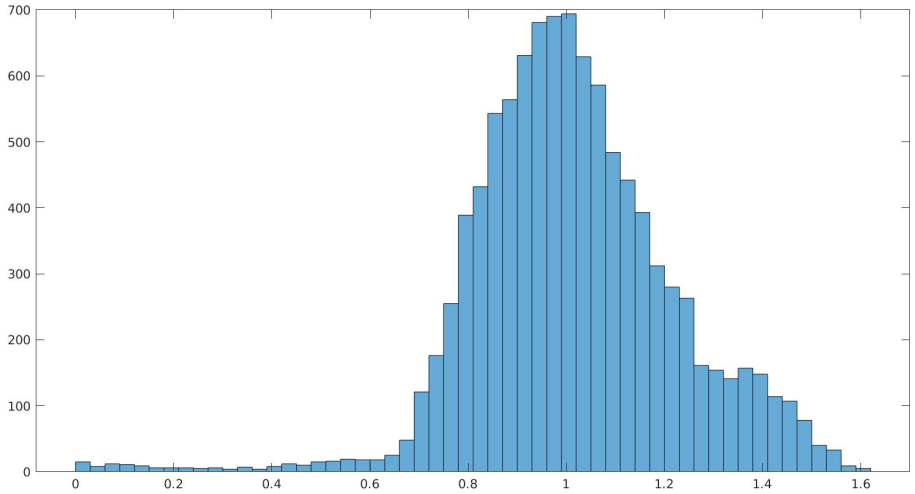


Figure 5.22: Histogram of generated speckle values obtained over the same by sampling 10000 diferent $z \sim N(0, 1)$, using the Speckle Generator trained on corrupted means.

is less correlated with the clean patch patterns. This displacement shows how the low-amplitude signal still influences the Speckle Generator output, confirming the analysis performed looking at Figure 5.21.

Finally, in Figure 5.23 we show a comparison between our Despeckler and other state-of-the-art models. Our model (image 6) achieves good performances in preserving edges and details with no blur and without introducing any artefact than the SAR-BM3D algorithm [7] (image 3) and the UNet without TV term of [11] (image 4). Still, it struggles in preserving punctual scatterers, even if we can see how the Despeckler better preserves details in areas where there is a high concentration of these objects than the model of [11] trained using the TV term. As for the CycleWGAN-GP model, we still have some residual speckle, even if its intensity has been drastically reduced. We still want to point out how this residual noise is due to our choice about the clean dataset. We can see how this choice influences the Despeckler performances looking at Figure 5.24. Here we can compare the intensity of the residual speckle of a Mean image (left), taken from the validation dataset, and the image obtained despeckling the corresponding area of a speckled SAR image taken from the temporal stack over which the Mean has been computed (right). Due to the kind of training we are performing, our model cannot eliminate this residual. Suppose our Despeckler would learn a despeckling function that generates smoother outputs over homogeneous areas. In that case, the Critic network will learn to distinguish these images from the Means in the training dataset thanks to the absence of residual speckle, giving them lower scores.

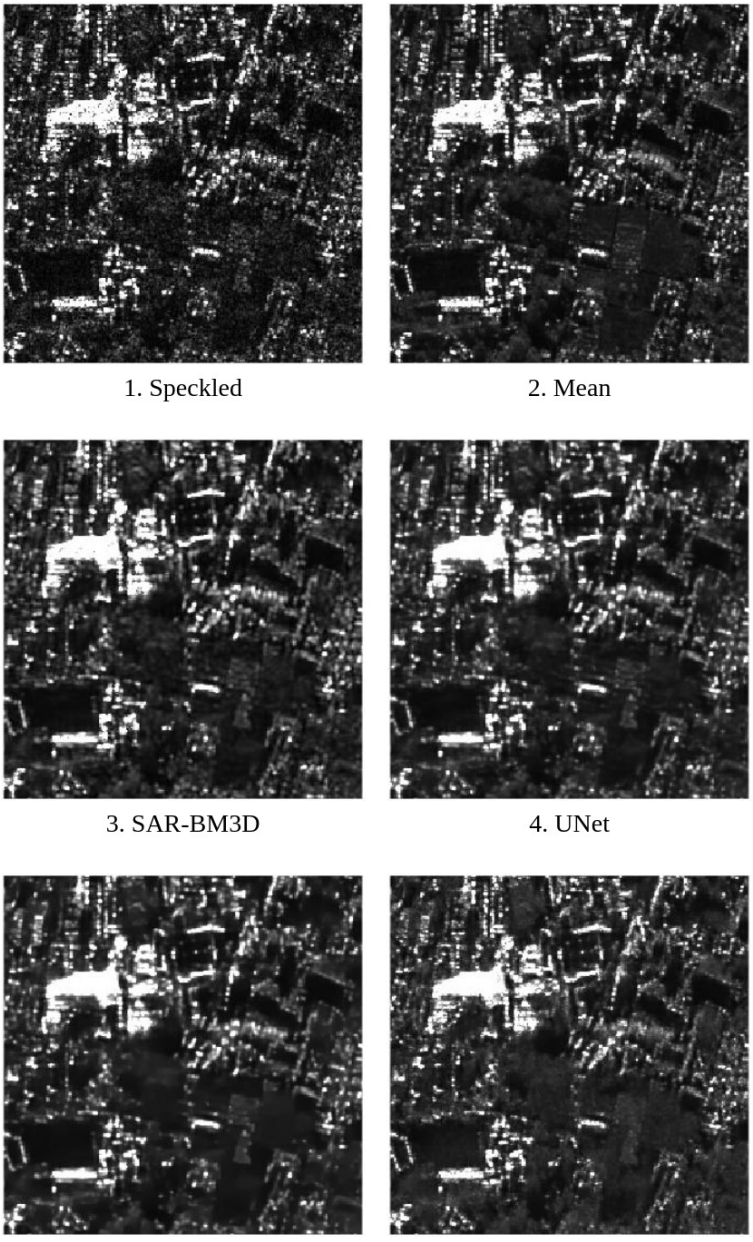


Figure 5.23: Despeckling performances of our network with respect to a Ground-Truth Mean [9] (image 2), the `SAR-BM3D` algorithm [7] (image 3) and the `UNet` architecture proposed by Lattari et al. [11] (image 4) also with the TV loss term (image 5). Our network (image 6) results in better-defined edges and contrasts regarding the blurry images 3 and 4 and does not present any artefacts. Also, it better preserves details over high-intensity scatterers than image 5 but still presents some residual speckle, even if its intensity has been drastically reduced from the CycleWGAN-GP model. Comparing the ground-truth image (2) with the smoother solution proposed in [11] (5), we can see how the means we are using as ground truth also presents some residual speckle.

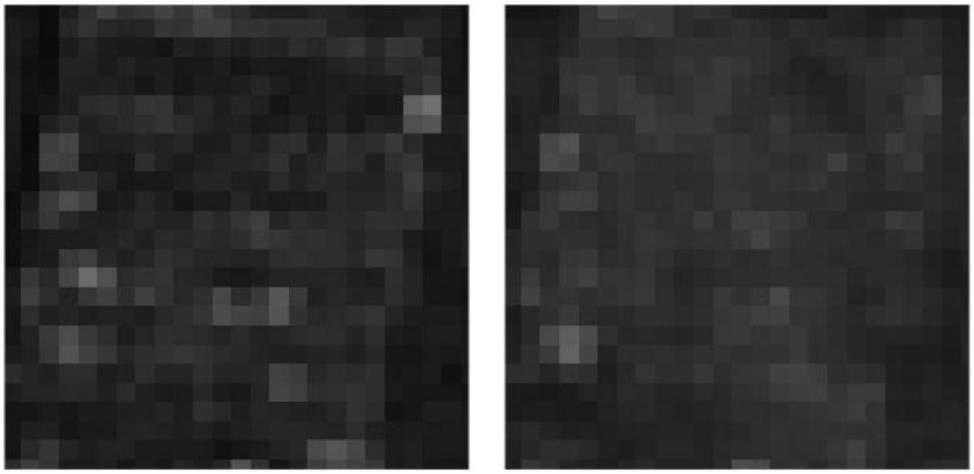


Figure 5.24: Detail of the residual speckle over a Mean image (on the left) and a SAR image despeckled using our CycleCVAE [31] implementation, trained using corrupted clean images (right).

However, adding the Gaussian noise η at training time showed a reduced intensity of this signal. If we look at the two images in this Figure, we can notice how the one produced by our model has smoother fluctuations in amplitude among adjacent pixels.

Since for SAR image despeckling no clean images are available, we performed the validation of this model thanks to the external support of a SAR expert from the same company that provided us with the images composing our dataset. He assisted us during all the work, from the creation of the dataset to the validation of the results. He gave us positive feedback about the state-of-the-art comparison shown in Figure 5.23, which made us more confident about the obtained results.

To perform a quantitative analysis of the obtained despeckling performance, we collected an additional test set from which we selected 100 patches 128×128 selected from a real SAR datastack. These patches have been manually inspected to select homogeneous regions, which can be considered stationary during time, and for which we can consider the temporal mean as an approximated GT for computing the evaluation metrics. On this set, we compute the standard metrics used in literature for evaluating despeckling performances of proposed algorithms: the Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity Index Measure (SSIM). Among the selected patches we further selected the most homogeneous areas to compute the Equivalent Number of Looks (ENL). We compare this metric also with the ground-truth images. The PSNR is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects its representation, and is used for evaluating the quality of the de-

	PSNR	SSIM	ENL
SAR-BM3D [7]	26.5095	0.9983	30.2738
UNet [11]	27.1033	0.9985	43.1244
UNet + TV [11]	27.8735	0.9986	215.2318
CycleWGAN-GP	25.2768	0.9970	118.8199
CycleCVAE	28.4936	0.9991	69.7789
Ground-Truth [9]			56.4223

Table 5.1: Performance comparison of our CycleWGAN-GP and CycleCVAE with other-state-of-the-art models on a numerical basis. The matrics we considered are the Peak Signal-to-Noise Rateo (PSNR), the Structural Similarity Index Measure (SSIM) and the Equivalent Number of Looks (ENL). We computed these metrics over 100 homogeneous patches extracted from a real speckled SAR image and its corresponding Mean.

speckled image with respect to the ground-truth. SSIM is a perception-based model that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. The ENL is a parameter which describes the degree of averaging applied to the SAR measurements and represents the degree of smoothing in a homogeneous region. Given x a clean patch and y the corresponding despeckled one, these metrics are defined as follows:

$$\mathrm{PSNR}(x, y) = 20 \log_{10}\left(\frac{\max(y)}{\sqrt{\mathrm{MSE}(x, y)}}\right),$$

MSE being the Mean Squared Error;

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

where μ_i is the mean of i, σ_i is the variance of i and c_i is a constant introduced to avoid instabilities;

$$\mathrm{ENL}(x) = \frac{\mu_x^2}{\sigma_x^2},$$

where μ_i and σ_i are the mean and the standard deviation of the considered patch x respectively. In Table 5.1 we use these metrics to compare our model with the SAR-BM3D algorithm and the two UNet architectures proposed by [11].

The metrics confirm the qualitative analysis. In particular, from the point of view of the PSNR, the CycleCVAE model shows better performance of our CycleWGAN-GP. This is because, the use of CVAE allows to store speckle

information into the latent vector instead of the outputs of the Despeckler network, reducing the intensity of the residual noise showed by the CycleWGAN-GP. Even if the CycleWGAN-GP model showed a better ENL, we want to point out that this network showed a strong residual noise over extremely heterogeneous and rich of details areas, while on homogeneous ones showed smoother details. Also, by comparing this metric with that showed by the ground truth images, we can see how these have a strong residual speckle. Thus, even if the CycleCVAE showed worse ENL results, those are closer to the ground-truth. Then, looking at the SSIM and the PSNR, the quality of the despeckled images drastically improved from the CycleWGAN-GP model to the CycleCVAE. We can then compare our model with other state-of-the-art approaches. Here we can see how, on those metrics, our model is outperforming these already present in literature, having a PSNR that is 2 points higher than SAR-BM3D and 1.4 and 0.6 higher than the two models proposed in [11], the baseline and the UNet with TV respectively. However, our model learned a despeckling function for mapping SAR images into the domain defined by the images computed using [9], which present a residual noise. Since the same class of images has been used as ground truth for computing those metrics, we expected higher values of PSNR and SSIM, while lower ENL, than the other models which tends to oversmooth homogeneous surfaces.

On the speckle generation side, we have no other proposed method in literature we can compare our model to nor metrics that may represent the quality of the generated noise. A useful feedback on the performance of the Speckle Generator has been given by the SAR expert that assisted us by performing qualitative analysis on the produced images. We show those results by looking at Figure 5.25, which shows the performances of the Speckle Generator. On the top left image, we can see the clean area on which we generate noise. The top right and bottom left images show the speckle generated over that area by our CycleCVAE baseline and the one trained corrupting the clean patches with Gaussian noise, respectively. Finally, the bottom right image shows the same area as it is captured by one of the speckled SAR images from the temporal stack used to compute the top left Mean. Thus, we can see how the two CycleCVAE models we trained have similar speckle generation performances, but there is still room for improvements. In particular, the generated speckle preserves details coming from the underlying scene that we would expect to be hidden by a stronger speckle.

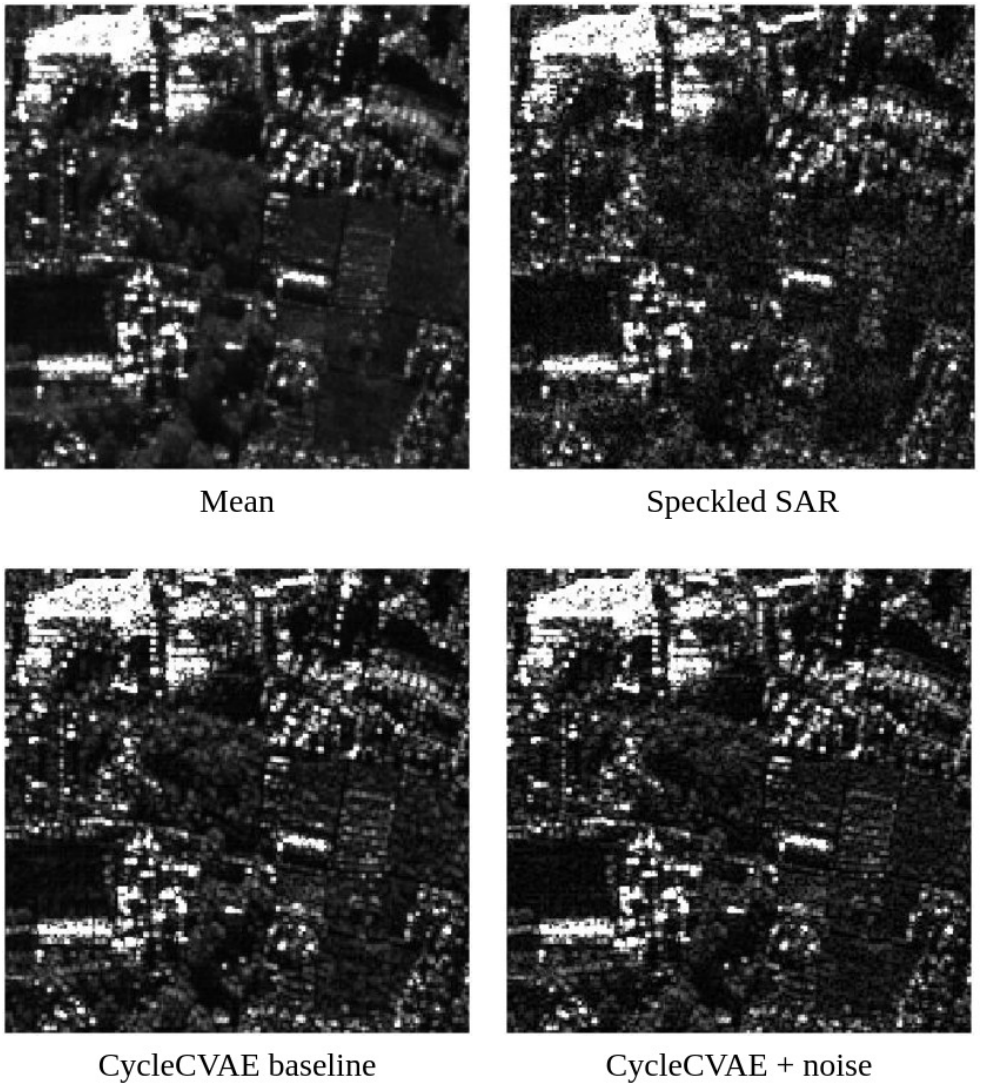


Figure 5.25: Comparison between the speckle generated by our CycleCVAE models and that of a real SAR image. The top left image shows the ground truth Mean of which we are generating speckled images. The top right shows the speckle generated by our CycleCVAE baseline, while the bottom left shows that generated by the model trained on corrupted clean patches. The bottom right images show a real speckled SAR image. We can see how corrupting clean patches do not affect the generated speckle quality. Nevertheless, the speckle quality is still cheap: generated speckled images still preserves too much detail of the underlying scene.

However, we proved how using our Speckle Generator network for building a synthetic dataset allows our Despeckler to achieve similar performances to state-of-the-art algorithms. In particular, our model showed remarkable performances in well preserving edges and details over homogeneous areas and strong scatterers clusters, even if it still struggles on isolated ones, without introducing any artefact nor adding ad-hoc loss terms as the TV of [11].

6

CONCLUSIONS

In this work of thesis, we presented a Deep Learning model for unsupervised SAR image despeckling. Given the absence of ground truth for training, we developed our method following the unpaired image-to-image translation framework presented in [26]. In particular, we designed two UNet-like networks, which we called Despeckler and Speckle Generator, for learning both a despeckling and a speckle generation function. We trained those networks using two adversarial loss, according to the WGAN-GP algorithm proposed in [25]. We used two generative models to let them learn those high-level features of both speckled and clean domains that supervised methods cannot capture. We paid particular attention to the data scarcity problem, avoiding any assumption about the speckle distribution like what is done in other methods in literature when building synthetic datasets. We trained those models concurrently, taking inspiration from the CycleGAN proposed in [26] which showed outstanding performance in various image-to-image translation tasks. This approach allowed us to train our model in an unsupervised fashion, using two sets of speckled and clean images in which no matching pair is present. We used the cycle-consistency loss term for forcing our networks to learn meaningful mappings from the speckled SAR domain into the clean one and vice-versa. By doing so, we let both the Speckle Generator and the Despeckler learn a different speckle distribution according to the areas. Furthermore, the training framework allowed us to obtain a good speckled images generator from the multi-temporal Means extracted using [9]. The Speckle Generator has been modelled using the CVAE [30] architecture allowing the generation of multiple samples of speckle, given the same underlying scene, conditioned over a latent vector, giving the possibility to generate realistic data for building a synthetic dataset. The proposed solution proves how unpaired image-to-image translation models can be successfully used for SAR image despeckling, as shown during the experiment conducted, and provides a good baseline for future developments.

In Chapter 5, we presented step by step every design choice we made and the results obtained during our exploration of the problem. We started with a more manageable task of using a WGAN [23] Speckle Generator for generating speckle patches. The WGAN choice was made due to the well-known limitations and training instability of classical GAN models. This network produced generated images loosely correlated with the input clean ones by losing detail over edges or adding strong scatterers over areas that resulted in homogeneous inputs. After introducing a ℓ_1 loss between clean

input and speckled output, we enriched our model by adding a Despeckler network and training the two using a CycleWGAN model. In terms of generated speckled images, we compared the results of these models and proved how the CycleWGAN drastically improves the output quality. The qualitative analysis demonstrated how the generated images preserved much more details over the edges than the WGAN baseline without introducing any strong scatterer that was not present in the clean input. We then illustrated the limitations of a simple WGAN model and why it was necessary to introduce a Gradient Penalty term for forcing the 1-Lipschitz in place of the Weight Clipping. The training of a CycleWGAN model resulted slow, and the quality of the information given by Critic networks was cheap, as proved in [25]. In particular, we had to schedule the weight given to the cycle-consistency term of our loss function to avoid vanishing gradients. Finally, we compared the so-called CycleWGAN-GP model with other state-of-the-art approaches. It achieved great results on both speckle generation and despeckling side but still presented a non-neglectable residual speckle, especially over extremely heterogeneous areas. We also demonstrated how our model successfully learned how to map different areas of a clean image to different speckle distributions on the speckle generation side.

Finally, we examined the one-to-many mapping problem and how this could impact the performance of a CycleGAN, forcing the many-to-one side network to hide helpful information in the produced images. We implemented a CycleCVAE model to allow the Speckle Generator to embed this information, which was necessary for generating speckled images, in a separate latent vector and showed how this drastically reduced the intensity of the residual speckle of the CycleWGAN-GP. We performed several tests to validate the quality of the generated speckle by closely analyzing the patterns presented by those images and plotting the amplitude intensity obtained over one pixel by generating 10000 speckled images sampling different latent vectors. The data gathered indicated a correlation between the speckle generated and a low-amplitude signal corrupting the clean reference images of our training dataset. In particular, we identified some patterns in this signal that the Speckle Generator amplified on images produced over the same area. Thus, we trained our final solution adding Gaussian noise to the clean images with an adequately calibrated amplitude for hiding this signal. We applied this noise before feeding them to the Speckle Generator for allowing our model to ignore the low-amplitude signal. The final model achieved outstanding performances when compared with other state-of-the-art approaches. In particular, it resulted not to be affected by artefact generation over heterogeneous areas and high performances in preserving edges and details on extremely heterogeneous surfaces in the presence of a high concentration of strong scatterers. Still, our solution struggles to preserve isolated scatterers, especially those whose intensity is not notably higher than surrounding

pixels. On the speckle generation side, this model showed wider variance on the values generated over a single pixel by sampling 10000 latent vectors and less correlation with the residual signal of the clean images than the previous model.

In conclusion, we computed the Peak Signal-to-Noise Ratio and the Structural Similarity Intex Measure over a set of 100 patches extracted from a speckled SAR image. These patches have been carefully selected to be homogeneous enough to consider the temporal stack average a good approximation of the equivalent clean. We manually identified those that resulted remarkably homogenous for an accurate estimation of the ENL. We analyzed those metrics and compared the result of our CycleWGAN-GP and CycleCVAE models with the SAR-BM3D [7] algorithm and the UNet and UNet with Total Variation of [11]. We noted better despeckling performances when considering a Mean computed over a temporal stack using the algorithm of [9] as ground truth. Finally, we computed the Equivalent Number of Looks metric over manually selected homogeneous patches. We noted how, even if a residual noise corrupts our clean dataset images and as a consequence also our despeckled ones, we achieve better performances than [7] and the baseline model, without TV, of [11].

6.1 FUTURE WORK

We showed in Chapter 5 how the model presented in Chapter 4 achieved comparable performances to other approaches present in literature. In particular, we want to point out how the unpaired image-to-image translation network we used allowed the Despeckler network to learn how to despeckle areas with peculiar speckle distributions properly. We compared our CycleCVAE performances with those of the UNet with TV proposed in [11], considered to be the state-of-the-art. We showed how our solution achieved better detail preservation in extremely heterogeneous areas with a high concentration of strong scatterers than [11], which tends to lose some details. Unfortunately, our model still struggles to preserve punctual scatterers, especially those with lower amplitude and particularly bright areas, which usually represent buildings and structures whose characteristics are crucial for performing various tasks. Lattari et al. [11] overcame this problem by adding a TV term, properly weighted accordingly to the corresponding ground-truth clean image. This term allowed their model to achieve smoother homogeneous surfaces and to preserve those scatterers better. Since we have no matching pair, we cannot use this term. We propose future work to deeper explore this problem and find a suitable solution. We suggest investigating an identity loss term between the speckled input of our Despeckler and the despeckled output. This loss must penalize outputs where the intensity of high scatterers pixels differs too much from the input. It may be possible to classify a pixel as a

scatterer if its intensity is over a certain threshold k, then compute an ℓ_1 or MSE loss term, properly weighted, between input and output input pixels. Let us define y the input speckled patch and $\tilde{x} = y - G_d(y)$ the despeckled one, let us define a patch s:

$$s_{i,j} = \begin{cases} 0 & \text{if } y_{i,j} < k \\ 1 & \text{otherwise} \end{cases} ,$$

we can then define the identity loss

$$\mathcal{L}_{id} = \|s \odot y - s \odot \tilde{x}\|_1 .$$

Considering this loss, the choice of the parameter k, which defines what a scatterer is, is crucial. Lower values of this parameter will also preserve unnecessary speckle in the clean image. Higher values will lead to losing important information about these objects.

We also suggest investigating possible improvements of the Speckle Generator further. Even if this network is doing an excellent job in recognizing different areas and accordingly generating speckle sampled from the proper distribution, the images produced by this network still preserve too many details coming from the underlying clean ones. We expected that a natural speckle would be strong enough to hide those details. However, then Despeckler would not recover them and properly minimize the cycle-consistency loss term. Thus, we propose treating the speckled-to-clean branch as a one-to-many mapping. The details of the underlying scene hidden by the speckle generated by an optimal Speckle Generator may be encoded in a latent vector, as done one the clean-to-speckled branch. One possible first step for starting this exploration could be to implement a symmetric version of the training described for our proposed solution in Chapter 4, implementing a CVAE architecture also for the Despeckler. Otherwise, there are various models in the literature which propose different approaches for dealing with many-to-many mappings, as the Augmented CycleGAN proposed in [34].

Finally, we proved how by using a CycleCVAE architecture, we could generate different speckle realizations. An interesting aspect that could be deeper investigated is whether the actual distribution showed by analyzing these realization matches what expected by the theory. Indeed, for carrying out a successful analysis of this aspect, some attention should be driven to the dataset since we proved how the low-amplitude signal present in the Means composing our clean images dataset affects the generated speckle. We also propose to test the efficiency of our Speckle Generator network, training a model in a supervised fashion using the speckle generated by this network for building a synthetic dataset.

BIBLIOGRAPHY

[1] Jong-Sen Lee. "Digital image enhancement and noise filtering by use of local statistics." In: *IEEE transactions on pattern analysis and machine intelligence* 2 (1980), pp. 165–168 (cit. on p. 18).

[2] Darwin T Kuan, Alexander A Sawchuk, Timothy C Strand, and Pierre Chavel. "Adaptive noise smoothing filter for images with signal-dependent noise." In: *IEEE transactions on pattern analysis and machine intelligence* 2 (1985), pp. 165–177 (cit. on p. 18).

[3] Victor S Frost, Josephine Abbott Stiles, K Sam Shanmugan, and Julian C Holtzman. "A model for radar images and its application to adaptive digital filtering of multiplicative noise." In: *IEEE Transactions on pattern analysis and machine intelligence* 2 (1982), pp. 157–166 (cit. on p. 18).

[4] Hua Xie, Leland E Pierce, and Fawwaz T Ulaby. "SAR speckle reduction using wavelet denoising and Markov random field modeling." In: *IEEE Transactions on geoscience and remote sensing* 40.10 (2002), pp. 2196–2212 (cit. on p. 18).

[5] Fabrizio Argenti and Luciano Alparone. "Speckle removal from SAR images in the undecimated wavelet domain." In: *IEEE Transactions on Geoscience and Remote Sensing* 40.11 (2002), pp. 2363–2374 (cit. on p. 18).

[6] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering." In: *IEEE Transactions on Image Processing* 16 (Aug. 2007), pp. 2080–2095 (cit. on p. 18).

[7] Sara Parrilli, Mariana Poderico, Cesario Vincenzo Angelino, and Luisa Verdoliva. "A Nonlocal SAR Image Denoising Algorithm Based on LLMMSE Wavelet Shrinkage." In: *IEEE Transactions on Geoscience and Remote Sensing* 50 (Feb. 2012), pp. 606–616 (cit. on pp. 18, 52, 53, 64, 65, 67, 73).

[8] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms." In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268 (cit. on p. 18).

[9] Alessandro Ferretti, Alfio Fumagalli, Fabrizio Novali, Claudio Prati, Fabio Rocca, and Alessio Rucci. "A new algorithm for processing interferometric data-stacks: SqueeSAR." In: *IEEE transactions on geoscience and remote sensing* 49.9 (2011), pp. 3460–3470 (cit. on pp. 2, 19, 33, 36, 52–55, 65, 67, 68, 71, 73).

[10] G. Chierchia, D. Cozzolino, G. Poggi, and L. Verdoliva. "SAR image despeckling through convolutional neural networks." In: (2017), 5438–5441 (cit. on p. 19).

[11] Francesco Lattari, Borja Gonzalez Leon, Francesco Asaro, Alessio Rucci, Claudio Prati, and Matteo Matteucci. "Deep Learning for SAR Image Despeckling." In: *Remote Sensing* 11 (June 2019), p. 1532 (cit. on pp. 20, 21, 25, 27, 52, 53, 64, 65, 67–69, 73).

[12] Giulia Fracastoro, Enrico Magli, Giovanni Poggi, Giuseppe Scarpa, Diego Valsesia, and Luisa Verdoliva. "Deep Learning Methods For Synthetic Aperture Radar Image Despeckling: An Overview Of Trends And Perspectives." In: *IEEE Geoscience and Remote Sensing Magazine* (2021), 2–24 (cit. on p. 20).

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: (2015) (cit. on pp. 27, 38, 39).

[14] Feng Gu, Hong Zhang, and Chao Wang. "A GAN-based Method for SAR Image Despeckling." In: *2019 SAR in Big Data Era (BIGSARDATA)* (Aug. 2019) (cit. on p. 25).

[15] P. Wang, H. Zhang, and V. M. Patel. "Generative adversarial network-based restoration of speckled SAR images." In: (2017), pp. 1–5 (cit. on pp. 23, 25).

[16] Ruijiao Liu, Yangyang Li, and Licheng Jiao. "SAR Image Specle Reduction based on a Generative Adversarial Network." In: *2020 International Joint Conference on Neural Networks (IJCNN)* (July 2020) (cit. on pp. 24, 25).

[17] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. "Noise2Noise: Learning Image Restoration without Clean Data." In: (2018) (cit. on p. 21).

[18] Emanuele Dalsasso, Loic Denis, and Florence Tupin. "SAR2SAR: a semi-supervised despeckling algorithm for SAR images." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2021), pp. 1–1 (cit. on pp. 21, 23).

[19] Andrea Bordone Molini, Diego Valsesia, Giulia Fracastoro, and Enrico Magli. "Speckle2Void: Deep Self-Supervised SAR Despeckling With Blind-Spot Convolutional Neural Networks." In: *IEEE Transactions on Geoscience and Remote Sensing* (2021), pp. 1–17.

[20] Adugna G. Mullissa, Diego Marcos, Devis Tuia, Martin Herold, and Johannes Reiche. "deSpeckNet: Generalizing Deep Learning-Based SAR Image Despeckling." In: *IEEE Transactions on Geoscience and Remote Sensing* (2020), pp. 1–15 (cit. on p. 22).

[21] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Networks." In: (2014) (cit. on pp. 5, 7, 8, 15, 24, 25, 42, 43).

[22] Michael Wand Chuan Li. "Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks." In: (2016) (cit. on p. 24).

[23] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN." In: (2017) (cit. on pp. 6, 7, 27, 29, 38, 41, 43, 47, 71).

[24] Cedric Villani. *OPTIMAL TRANSPORT : old and new.* 2016 (cit. on p. 6).

[25] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. "Improved Training of Wasserstein GANs." In: *arXiv:1704.00028 [cs, stat]* (Dec. 2017) (cit. on pp. 7, 8, 27, 29, 30, 34, 43–45, 47, 71, 72).

[26] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks." In: *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017) (cit. on pp. xix, xxi, 2, 8–10, 14, 15, 26, 27, 34, 37, 42, 43, 54, 71).

[27] Dina Bashkirova, Ben Usman, and Kate Saenko. "Adversarial Self-Defense for Cycle-Consistent GANs." In: *Advances in Neural Information Processing Systems* 32 (2019) (cit. on pp. 10, 54).

[28] Carl Doersch. "Tutorial on Variational Autoencoders." In: *arXiv:1606.05908 [cs, stat]* (Jan. 2021) (cit. on pp. 12, 13).

[29] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes." In: (2013) (cit. on pp. 11, 13, 14, 34, 54, 57).

[30] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning Structured Output Representation using Deep Conditional Generative Models." In: 28 (2015). Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (cit. on pp. 2, 12, 14, 32, 56, 58, 71).

[31] Qipeng Guo, Zhijing Jin, Ziyu Wang, Xipeng Qiu, Weinan Zhang, Jun Zhu, Zheng Zhang, and David Wipf. "Fork or Fail: Cycle-Consistent Training with Many-to-One Mappings." In: *arXiv:2012.07412 [cs]* (Jan. 2021) (cit. on pp. 2, 10, 13, 27, 54, 56, 57, 66).

[32] Yu Li, Sheng Tang, Rui Zhang, Yongdong Zhang, Jintao Li, and Shuicheng Yan. "Asymmetric GAN for Unpaired Image-to-image Translation." In: *IEEE Transactions on Image Processing* (2019), pp. 1–1 (cit. on pp. 11, 15, 54, 56, 57).

[33] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. "Toward Multimodal Image-to-Image Translation." In: *arXiv:1711.11586 [cs, stat]* (Oct. 2018) (cit. on p. 56).

[34] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. "Multimodal Unsupervised Image-to-Image Translation." In: *arXiv:1804.04732 [cs, stat]* (Aug. 2018) (cit. on p. 74).