MVA - Remote sensing project

Mathieu Rita
Ecole Polytechnique
mathieu.rita@polytechnique.edu

Tristan Dot
ENS Paris-Saclay
tristan.dot@mines-paristech.fr

Abstract

The aim of the project is to denoise SAR images contaminated with speckle noise. Deep-learning approaches (eg. DnCNN, FFDNet) show great performance for image denoising, outperforming other classical methods (BM3D).

In this project, we are going to adapt FFDNet, the state-of-the-art DL algorithm for image denoising, to SAR images. We will have to face several challenges: the statistic of the speckle noise, its multiplicative aspect, the high noise level, etc. While several approaches are proposed (raw denoising, homomorphic approach,...), we are mainly going to work with amplitude images, wisely adapting the FFDNet input noise map in order to get the best possible results.

1. Introduction

1.1. Speckle noise

SAR images are complex-valued, but only their modulus, corresponding to their amplitude, is informative. In order to manipulate it easily, we are going to work on their intensity, corresponding to the square of their amplitude. Following the model of Goodman [1], this intensity follows a gamma distribution, with probability density:

$$p_I(Y|X) = \frac{L^L Y^{L-1}}{\Gamma(L) X^L} \exp(-L\frac{Y}{X}) \tag{1}$$

With: Γ the gamma function, Y the image intensity, X the image reflectivity (i.e. the image without noise, related to the radar cross-section), and $L > 0$ the number of looks.

Formally, the intensity Y can be decomposed as a product of the reflectivity and a speckle component S, which follows a standard gamma distribution. We therefore get the following equations:

$$Y = X \times S, \quad \mathbf{E}(Y) = X \quad and \quad Var(Y) = \frac{X^2}{L} \tag{2}$$

The variance depends on the expectation: fluctuations are signal dependant. Interestingly, the gamma distribution has a very heavy right-tail, linked to the bright outliers observed in SAR intensity images.

In order to transform multiplicative fluctuations to additive ones, the log-transform can be applied: $y = \log Y$. y then follows a Fisher-Tippett distribution:

$$p_y(y|x) = \frac{L^L}{\Gamma(L)} e^{L(y-x)} \exp(-Le^{y-x}) \tag{3}$$

with $x = \log X$. We then have the following properties:

$$\begin{aligned} y &= x + s \\ \mathbf{E}(y) &= x - \log L + \psi(L) \\ Var(y) &= \psi(1, L) \end{aligned} \tag{4}$$

with s following a standard Fisher-Tippett distribution, $\psi()$ the digamma function, and $\psi(., L)$ the polygamma function of order L. To note: the variance no longer depends on the expectation: fluctuations are now signal independant. Importantly, as it can be seen in Eq. 4 the noise has a non-zero mean on log-transformed data.

1.2. State of the art on despeckling

In order to efficiently despeckle images, several methods can be considered: multi-looking processing [2], filtering methods [3], wavelet-based methods [4], block-matching 3D (BM3D) [5] and Total Variation (TV) [6] methods, but, more recently, Deep Learning methods have given gold-standard results [7, 8, 9, 10]. Some of these deep-learning methods apply homomorphic processing (log-transform) to the data before working on it [9] (as presented in Sec. 1.1), while others work directly in the original domain of the image [7, 8, 10]. Some papers use U-Net Convolutional Neural Networks (CNN) [8], while others are based on more "classical" CNN. Interestingly, in [7], the training loss function is modified to solve the despeckling problem: the Euclidian loss and the Total Variation (TV) loss are jointly minimized to provide smooth results.

Generally speaking, concerning image denoising, different losses can be considered [11], from a classical ℓ_2 loss, to a ℓ_1 loss, to a structural similarity index (SSIM) [12] based

loss. Citing Zhao et al. [11], it appears that the ℓ_1 loss could have better local convexity properties: it may be easier for ℓ_1 to reach a better minimum.

In order to evaluate denoising results, different error measures can be considered, apart from qualitative human judgement. The Peak-Signal-to-Noise Ratio, PSNR (which corresponds to the simple ℓ_2 error function), is widely used, but is known for not capturing well the characteristics of the human system [13]. SSIM is a popular index, which is based on the idea that the Human Visual System is sensitive to changes in local structures. In the following study, we will evaluate our denoising results using our human qualitative perceptions, and sometimes the SSIM measure in order to confirm our choices. Nevertheless, we found not very relevant to spend a lot of time using those measures since they often do not provide relevant information.

1.3. FFDNet

In the following project, a FFDNet network [14] will be used as our denoising network. It consists in a convolutional neural network with a tunable noise level map as input. Working on downsampled sub-images, it works with a wide range of noise levels (i.e. [0-75]), uses non-uniform noise level maps in order to remove spatially variant noise, and computes very fast compared to benchmark methods such as BM3D.

The **architecture** of the network is the following one. The first layer consists in a reversible downsampling operator which reshapes a noisy image into four downsampled images, concatenated with a tunable noise level map. The output of this first layer is a tensor which constitutes the input of the CNN. Then, the CNN consists in a series of 3×3 convolution layers, with: a 'Convolution + Rectified Linear Units' at the beginning, 15 'Convolution + Rectified Linear Units + Batch Normalization' layers in the middle, and a 'Convolution' layer at the end. Finally, an upscaling operation is applied. See Fig. 1 for an illustration of the network architecture. Interestingly, the downsampling operations result in obtaining a very large receptive field, of size 61×61. Moreover, the Batch Normalization step is known to improve the denoising process and accelerate the training.

The **noise map**, which specifies a particular noise level for each pixel of the image, is used to manage the trade-off between noise reduction and detail preservation, in the case of spatially variant noise. In the Goodman's model of Speckle noise, as seen in Eq. 2, the variance depends on the expectation. It would then appear logical to adapt noise reduction to the expectation of each pixel. Ideally, we could therefore use the real (without noise) image as our input noise map. Nevertheless, since the real pixel values are not known to us, the following oracle strategy could be used: first denoising the image with a uniform noise map, and then using this approximate denoised image as our noise map. Approximating the denoised image by a local means method could also be considered. To note: the use of such a non-uniform noise map is only useful when working on raw images, without applying log-transform to the image, since, in this last case, the variance no longer depends on the expectation, as reflected in Eq. 4.

2. Methodology

We are going to consider images with a number of looks $L = 1$, linked to a very high noise level. Moreover, as speckle corresponds to spatially variant noise, and FFDNet is trained on Gaussian noise, we must build a strategy to adapt FFDNet to speckle noise. Importantly, we should always smartly deal with the trade-off between denoising quality and detail preservation.

In this section, we are going to present the whole pipeline that allows us to denoise SAR images. We are going to present:

- The preprocessing of the data
- The different approaches and choices tested
- The methodology we will use to evaluate our results
- The training details (that we will use for all our approaches)

2.1. Preprocessing of the data

The nature of SAR images requires some pre-processing steps. For a SAR image, the range of values is very large (black areas and very shiny scatterers) and is different between all the images of our dataset. In order to get a normalized range of values without introducing bias from an image to another, we normalized our samples taking the maximum/minimum of all the images. Consequently, if we note M the maximum value of the pixels of all the images and m the minimum value:

$$\text{im\_normalized} = 255 * \frac{im - m}{M - m} \tag{5}$$

(**Rk:** Here, our images are mapped to the classical range $[0, 255]$. When fed into the networks, the networks are sent to the range $[0, 1]$ to fit FFDNet optimization constraints (merely by dividing by 255)).

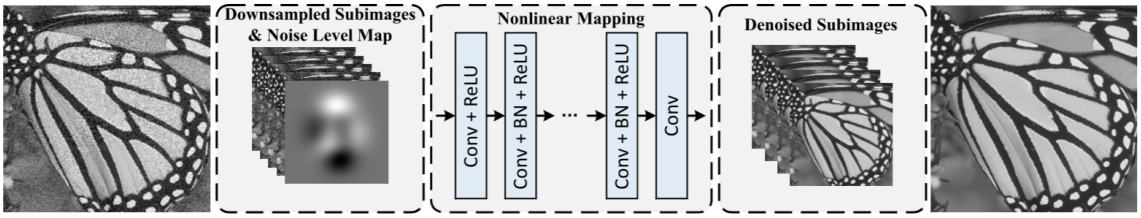


Figure 1: Architecture of the FFDNet Network

2.2. Approaches to solve the problem

As shown in Sec. 1.2, there are two main strategies studied when working with speckle noise: working in amplitude and working with the homographic approach. We can summarize quickly the advantages and drawbacks of the two methods:

2.2.1 Working in amplitude

Working with the raw images seems to be the most natural method when working with convolutional networks. Indeed, the aim of the training is to find the best convolutional kernels that will enable the network to reconstruct the edges of the images.

In the amplitude case, the noise is spatially variant: $\sigma[Y|X] \propto X$, and two main methods can be considered in order to determine a coherent noise map (if non-uniform), which approximately estimates the true image: an iterative method (or oracle method), and a local means method (using sliding windows).

The general approach for amplitude images can be summarized in Fig. 2. The training and testing steps are the following ones:

- **Step 1: Training** We take the set of raw images. Then, speckle noise is generated and added to the images. After normalization to $[0, 1]$ (as explained previously), the pair of images (X: noisy normalized image, Y : raw normalized image) is fed to the network. The noise map is uniform, or based on an iterative / local means method.

- **Step 2: Testing** Then, for the testing part, we take a raw image contaminated with generated speckle and feed it into the network. We also feed the network with a noise map of one of the methods mentionned.

2.2.2 Working in log

Then, another natural method is to move to a log-scale. Indeed, all classical denoising methods (BM3D, Non-local means, ...) are built to denoise additive noise. Therefore, it could be easier for FFDNet to denoise additive noise. Moving to a log-scale allows us to transform the multiplicative noise into an additive noise, as explained in introduction. Moreover, knowing the statistics of the speckle (Eq. 2, Eq. 4), it is easier to work in the log-space where the variances of the noisy pixels do not depend on the real value of the pixel. Therefore, the level of noise becomes uniform across the image and thus, we do not have to use a non-uniform noise map. The denoising steps are:

- **Step 1: Training** We take the raw image, normalize it and add speckle noise to it. Then, instead of sending (X,Y) to the network, we send the pair: ($\log(X)$, $\log(Y)$). The network therefore tries to map the log noisy image to the log true image.

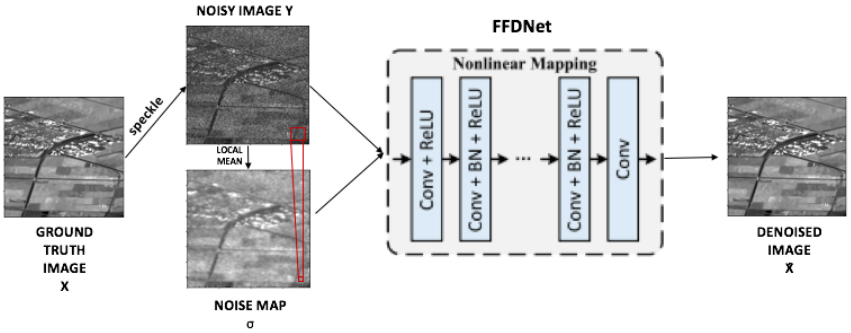


Figure 2: Experimental setup

- **Step 2: Testing** For the testing part, we feed the normalized log version of a raw image contaminated with speckle noise. Then, we get the output of the network (denoised image in the log space) and go back to the raw space with an exponential transformation.

In this project, we will focus on the work in amplitude because we thought more interesting to focus on the noise map, which is the specificity of FFDNet.

2.3. Training details

From a general point of view, our training was based on 7 Sentinel-1 images, cut in 1232 patches of size 128×128. Adam optimizer was used. Since training the network from scratch was too time consuming, we decided to use the weights of a pretrained FFDNet [1] as our network initialization, and considered approximately 20000 iterations during the training. The loss evolution can be seen in Fig. 3.

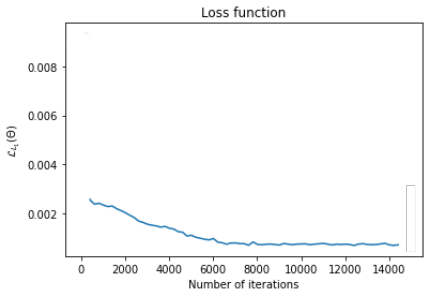


Figure 3: Evolution of the loss on the training set

Both ℓ_1 and ℓ_2 losses were considered in our experiments.

2.4. Evaluation methodology

To evaluate our results, we used 3 criteria:

- **Statistics of the noise** :We check that the residual noise estimated by the algorithm follows the right statistics (Rayleight distribution for the amplitude ; exponential distribution for the intensity). In addition, we quantify the mean and variance of the residual noise in order to check the two expected results:

$$\mu_s = 1$$
$$\sigma_s^2 = \frac{1}{L}$$

- **Qualitative approach:** Once the statistics is verified, we have to see whether the quality of the denoising is good. The first approach is merely to check by eye if the denoising is correct.

- **Quantitative approach:** Eventually, we will use the traditional metrics to evaluate the performances of our models: PSNR and SSIM (see Sec. 1.2)

3. Results and analysis

In this section, we are going to present the approach we followed. In preliminary experiments, we compared the results obtained with a model trained from scratch and the training based on the weights provided by FFDNet's team [2]. While the training from scratch is very time consuming and leads to weak results, performing transfer learning is quite quick ($\sim$ 20k iterations) and gives correct results. For such a project, where we want to get good results with a low computational power, we decided to work on pre-trained models. The main issue is that the pre-trained weights come from data with gaussian noise. We will therefore have to check that the network is able to interpret noise distribution change. Consequently we will first try to check that the statistics of the denoised image follows the speckle model.

3.1. From a gaussian pretrained model to speckle model

The first thing we have to check is that the use of weights that come from gaussian noise training does not bias the statistics of the denoised image. For this model, FFDNet has been trained with an additive gaussian noise of various standard deviation. For each noise level value, an uniform noise map equal to this value is fed to the network. In the end, the value of the noise map corresponds to the expected value of the noise level at a pixel.

Then, we first tried to merely apply the model. When we roughly use the weights without additional training, we see that the results are bad (see Fig. 4 and Appendix 1) and that the residual noise follows a gaussian distribution. In Fig. 5, we see that the distribution of the residual noise (noisy - denoised) follows a gaussian distribution with a standard deviation equal to the value of the applied uniform noise map.

Then, the natural question was to investigate whether the network was able to adapt the distribution of the residual noise to speckle distribution when performing transfer learning. Thus, we plotted the distribution of the residual noise ($\frac{noisy}{denoised}$) during the training to see the evolution of the statistics of the gaussian noise. We observed that very quickly the distribution of the residual noise was transported from a gaussian distribution to a Rayleigth distribution (ie. the right distribution of Speckle noise) (see 6 for the distribution of the residual noise after 5000 iterations and Appendix 2. for the distribution of the residual noise at the end of the training).

[1] https://github.com/cszn/FFDNet/tree/master/models

[2] https://github.com/cszn/FFDNet

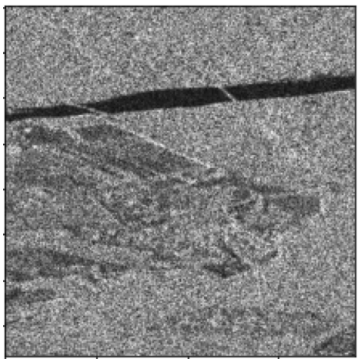


Figure 4: Results of denoising with the pre-trained weights (uniform noise map with $\sigma = 20$)

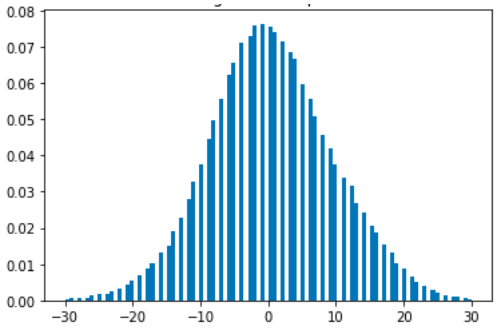


Figure 5: Distribution of the residual noise (uniform noise map with $\sigma = 20$)

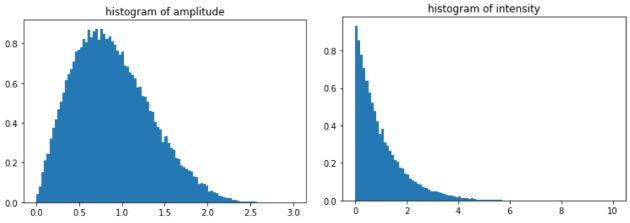


Figure 6: Distribution of the residual noise after 5000 iterations (a- amplitude ; b- intensity)

In the end, we see that performing transfer learning is efficient and FFDNet is able to adapt the statistics of the residual noise. For all our models, we obtain as expected the correct distribution (mean value almost equal to 1 and variance close to 1 for the residual noise in intensity).

Nevertheless, we observe that there was still an influence of the pre-training statistics on the distribution of the residual noise. Indeed, when we analyze the results for various values of noise levels (from $\sigma = 0$ to $\sigma = 50$), the statistical results changed:

- the expectation was unchanged
- the error of variance increases with σ

The absence of additional bias but the change in variance leads us to modelise the statistics of the residual noise as a mixture of the target Rayleight distribution $\mathcal{R}$ and the gaussian distribution $\mathcal{R}$ (not exactly a gaussian noise since we consider here the multiplicative residual noise) by:

$$S = \mathcal{R} + \alpha\mathcal{G} \tag{6}$$

where α is the coefficient of the mixture that decreases with the number of epoch. This mixture can explain why the use of pre-trained weights leads to a short error in variance while not adding bias. We started to investigate whether it was possible to get a quantitative estimation of this parameter α but the researcher does not lead to conclusive results, only a global tendency was observed.

3.2. Results of FFDNet without using the noise map

Once we verified that using pre-trained weights has no impact on the statistics of the residual noise (or a small impact), we start to evaluate the performance of FFDNet. Throughout the section, we are going qualitatively compare the results and present the researching path that leads us to the best results.

Firstly, we started by evaluating the results of FFDNet after training the pre-trained weights with the SAR dataset without using the noise map. Therefore, speckle is added to images and FFDNet is fed with a uniform noise map equal to 0. All the results with a model trained with an ℓ_2-norm are presented in Appendix 2. and an example is presented in Fig. 7

Figure 7: Result for a SAR image (loss: ℓ_2 / no noise map)

We see that the results are good and that we reach the performance of a lot a deep learning algorithm for despeckling. Here are some comments about the results:

- the edges are well reconstructed since that we recognize the main shapes of the images.
- The flat areas are well denoised and there is not a lot of artefacts (compared for example with BM3D where oscillations appear due to the DCT basis)
- The first main drawback is that the model tends to average too much the image. The effect is that we loose some textures in the flat parts (see the fourth image of Appendix 2 for example) and some details. Moreover, the model does not manage to discriminate close shades of gray.
- Overall, there is an undesired blurry effect that we want to remove

3.3. Influence of the loss (ℓ_1 VS. ℓ_2)

The previous results come from a model that has been trained with a ℓ_2-norm as it is the default norm proposed with FFDNet. Nevertheless, a natural test we have to do is to compare the results obtained with an ℓ_2-norm and an ℓ_1-norm. As it is described in Sec. 1.2, training ℓ_1-norm is often said to lead to better results. In our cases, we report the comparative results between the two losses in Appendix 3 and an example in Fig. 9.

We see that the results obtained are really close. Nevertheless, we can notice some small improvments provided by the ℓ_1-norm. Indeed:

- We can see that the shades of gray are better treated. On the example, we see that the edges in the fields are better defined at the interface between close shades of gray.
- In addition, the blurry effect that we observed with the ℓ_2-norm is attenuated with the ℓ_1-norm. We can see it in the examples 2-3-4 of the Appendix 3.

Overall, we can consider that the results obtained with the ℓ_1-norm are slightly better than those obtained with the ℓ_2-norm.

3.4. Working with a uniform noise map

From now, we did not use the noise map. The results are already quite good but we can expect to get even better results by using the noise map. As described in Eq. 6, the effect of the noise map is equivalent to add a uniform smoothing of the image. Qualitative results are shown in

Figure 8: Result for a SAR image (loss: ℓ_2 / no noise map)

Figure 9: Result for a SAR image (loss: ℓ_1 / no noise map)

Appendix 4. As we can see, the greater σ, the smoother the result is. Indeed, a uniform noise map is useful to smooth the flat areas and remove additional artefacts that can be added during the denoising. From this point of view, the noise map seems interesting.

Nevertheless, as we can see in the previous subsection, FFDNet tends to smooth the images too much even with a 0 noise map. The result is that we loose some textures and details. Therefore, the main challenge at this point is to find a way to recover those details. However, by using noise maps with non-zero noise level, we add some smoothing, which is not something we really want. This is why, we tried another approach: we trained a model where the intensity of the added speckle is less important (typicaly L=5). Therefore, the models learns to "less" denoise the images. With

such a model we expect to have a default model that keeps details but which is not able to well denoise flat areas. Then by adjusting the noise level σ, we can play with the trade-off between smoothness and details preservation.

In Appendix 4b, we show the results for various values of σ (from 0 to 30). As we can see, the result with $\sigma = 0$ is still noisy while the result for $\sigma = 30$ is really close to the result of the previous model without the use of noise map (see Appendix 4a with $\sigma = 0$). Thus, there is a range of values (from 0 to 30) of σ we can play with to adjust the trade-off between smoothness and detail preservation. To finally select the right value, we used both a qualitative evaluation and a quantitative evaluation using the SSIM coefficient. For our 7 images, the SSIM evaluation is shown in Fig. 10.

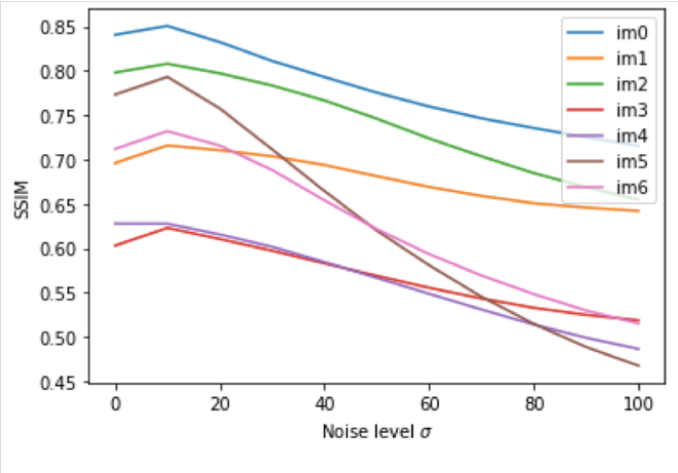


Figure 10: SSIM evaluation for different noise level σ

As we can see for all our images, the best results have been obtained using values of σ between 10 and 20. This evaluation is confirmed by eye. In the end, we choose an "optimal" value equal to 20 for all the images. The results on all the images are shown in Appendix 5 and an example is shown in Fig. 11.

In the appendix, we show the comparison with and without the use of the noise adaptive noise map. Then, here are some comments we can make:

- Let's remind that the aim of the use of the noise map is to recover textures and details and refine the edges. As we can see on all the examples, the use of the noise map appears as a powerful tool to address this challenge. We get a better granularity with the use of noise map while smoothing the flat part of the images. Both details and textures are reintroduced in the denoised images (see especially images 2-4-5 of Appendix 5).

Figure 11: Result with a uniform noise map (loss: ℓ_1 / $\sigma = 20$)

In addition, we see that the edges at the interface between close shades of gray are better defined. We can observe this effect in the images of fields (1 and 3) where the model is now able to better distinguish the limits of the fields.

- The main drawback is that there are still some artifacts and effects that we may want to remove. This could be due to the fact that we are applying a uniform noise map while the actual intensity of the noise depends on the value of the pixel (speckle model). A natural improvement is therefore to investigate how we can build a non-uniform noise map.

3.5. Working with non-uniform noise map

With the previous section, we found a criterion to play with the trade-off between smoothness and details preservation. Nevertheless, we only considered uniform noise maps. However, the specificity of SAR images is that the intensity of the noise on a pixel depends on the true value of the pixel. Therefore, it seems better and useful to use non-uniform noise maps.

In the speckle model, the noise level is proportional to the real image. Thus, the truth image seems to be right candidate as noise map. Nevertheless, we do not have access to the truth image. To face this issue, we tried two methods to have a quick estimate of the truth image: a local means method an oracle method:

- **Local means**: The principle of the local means method is to use the simplest denoising algorithm we

can think of. We apply a kernel on the image an estimate the truth value of a pixel by averaging the pixels that surrounds it in the kernel.

- **Oracle**: The oracle is an iterative method. The idea is that, at this point, we already have models that are able to provide a good estimate of the truth image. Then, we can use the output of such a model as input (noise map) of a new model.

We tried these two methods in two situations:

- **Application during training time**: We trained the models with non-uniform noise map estimated with one of the two methods. We realized that training the models with non-uniform noise maps has a very bad effect. When using the local means method, the outputs were very noisy. When using the oracle method, the outputs of the models were very close to the noise map provided since that it is equivalent to train it with a uniform noise map: in brief, FFDNet takes the noise map as additional information and overfits it. The effect is particularly stricking. If we give the real image as noise map during the training time to force the algorithm to use the right noise level, we see that the model actually simply outputs the noise map.... with is not useful when we work on a real case.

- **Application during testing time**: The second approach is to use the models trained during the previous sections on uniform noise maps and create noise maps as described by the two methods. We observed that the fact that FFDNet has been trained only on uniform noise maps does not lead to bad effects due to the use of non uniform noise maps.

Consequently, we tried the two methods during the testing time. The results are shown in Appendix 6 and an example is shown in Fig. 12

In Appendix 6, we compared the results obtained with the uniform noise map, the local mean method and the oracle method. Here are some comments about the results we got:

- The results of the three methods are overall very similar. Nevertheless, we get some interesting improvements by using the non uniform noise map. On the images where there is a lot of textures, we see that the granularity is improved. Moreover, we see that there were some flat areas that contain artefacts when we denoised with the uniform noise map. Thanks to the non uniform noise map (especially the oracle method), some of these artefacts disappear.

Figure 12: Result with a uniform noise map (loss: ℓ_1 / $\sigma = 20$)

- The results obtained with the local means method and the oracle method are close. Nevertheless, we can consider that the results obtained with the oracle method contains less artefacts.

3.6. Conclusion on the study

In the end, we see that FFDNet allows us to get very good results for the despeckling task. The evolution of the results throughout the study are shown in Appendix 8. As we can see, just performing transfer learning and training with the ℓ_1-norm led to correct results (that can be compared to current deep learning algorithms for despeckling). However, the main improvement is made by adapting the noise map. If we cannot clearly quantitatively quantify the value of the noise map (since the noise level σ belongs to a gaussian model), it is very easy to calibrate it. The method we propose is merely to train the model on a speckle noise that is slightly less intense ($\sim L = 5$) not to denoise too much (since FFDNet tends to average a little bit more the image) with a 0-noise map and then to adapt the noise level in order to smooth the image. If the results are very correct with a uniform noise map (actually, during the training, FFDNet learns the non-uniformity of speckle noise), we can improve the results by estimating a non-uniform noise map. For now on, the best estimation we found was to iterate the denoising process twice and feeding the result of the first iteration as noise map of the second iteration. One major force of the noise map is that we can adapt the results to the user, depending on whether the aim is to preserve details or mainly smooth the image to avoid having artefacts.

If we can still try to improve our results (there are still some artefacts), they seem globally satisfying.

4. Extension to real images

Now that we built models to denoise generated speckle noise, we want to see whether we can generalize our model to real images. In our trainings, the generated speckle was independent for all the pixels. Nevertheless, it is not the case in real life where the pixels are correlated. Therefore, we want to see whether the presence of this correlation has bad effects on the denoising. For other deep learning approaches (Sar-CNN), it leads to artefact.

4.1. Results

For our models we report the results for testing images (Sentinel-1 and TerraSAR-X) in Appendix 7 when using a uniform noise map and the oracle method. An example of denoised real image is also shown in Fig. 13b .

Here are some comments about the results we get:

- As we can see in Appendix. 7, the results we get are very satisfying. In particular, we do not observe recurrent artifacts as expected. It seems that the model easily generalize to real images.
- We show in the appendix. 7 the comparison between the result obtain by merely applying a constant noise map and the oracle method. As we can see, the edges are better defined with the oracle and there is some interesting textural effects appearing. Nevertheless, we also get additional artifacts. Thus, depending on what the user want, one method or the other can be said to be better.

4.2. Discussion

Those very good results, on true images, are quite surprising. However, we could think that the subsampling operations realized at the beginning of the FFDNet network allow to decorrelate the pixels. Indeed, subsampling operations are known to decorrelate the pixels. Nevertheless, we want to be careful with the results we get. Indeed, the network has been trained on patches/images that are very close to the target patches of the real we test (in particular, they come from the same satellite). From this point of view, it could be consired as a kind of overfitting.

However, we also tested our model on an image that comes from terraSAR-X. The result (with oracle noise map) is shown in the Appendix. 9 (we also put the result of Sar-CNN as a source of comparison). As we can see, the result is very good and some bad effects that are observed with Sar-CNN (blurry effects) are not present in the denoising. Therefore, the example allows us to think that the model may be strongly adapted to more types of SAR images.

5. Conclusion and ackowledgement

In this project, we proposed an adaptation of FFDNet to the despeckling of SAR-images. A lot of approaches are present in the literature and throughout the project we tried some methods (amplitude approach, homomorphic approach, noise2noise...). Nevertheless, since it is quite a short project, we decided to mainly focus on one aspect of FFDNet, the work in amplitude and the adaptation of the noise map because it is the main specificity of FFDNet. In the end, the approach we proposed allows to get strong and good results that are quite promising. This work lacks a strong quantitative analysis. Nevertheless, we thought that it was more interesting for such a project to deepen the qualitative analysis rather than spending a lot of time with artificial metrics.

Finally, we want to **acknowledge** Florence Tupin and Emanuele Dalsasso for the wise advices they gave us.

References

[1] J. W. Goodman. “Some fundamental properties of speckle∗”. In: *J. Opt. Soc. Am.* 66.11 (Nov. 1976), pp. 1145–1150. DOI: 10.1364/JOSA.66.001145. URL: http://www.osapublishing.org/abstract.cfm?URI=josa-66-11-1145.

[2] Paul A. Thompson et al. “Spotlight-Mode Synthetic Aperture Radar: A Signal Processing Approach”. In: 1996.

[3] A. Baraldi and F. Parmiggiani. “A refined gamma MAP SAR speckle filter with improved geometrical adaptivity”. In: *IEEE Transactions on Geoscience and Remote Sensing* 33.5 (1995), pp. 1245–1257.

[4] A. Achim, P. Tsakalides, and A. Bezerianos. “SAR image denoising via Bayesian wavelet shrinkage based on heavy-tailed modeling”. In: *IEEE Transactions on Geoscience and Remote Sensing* 41.8 (2003), pp. 1773–1784.

[5] Kostadin Dabov et al. “Image denoising with block-matching and 3D filtering”. In: *Proceedings of SPIE - The International Society for Optical Engineering* 6064 (Feb. 2006), pp. 354–365. DOI: 10.1117/12.643267.

[6] J. M. Bioucas-Dias and M. A. T. Figueiredo. “Multiplicative Noise Removal Using Variable Splitting and Constrained Optimization”. In: *IEEE Transactions on Image Processing* 19.7 (2010), pp. 1720–1730.

[7] P. Wang, H. Zhang, and V. M. Patel. “SAR Image Despeckling Using a Convolutional Neural Network”. In: *IEEE Signal Processing Letters* 24.12 (2017), pp. 1763–1767.

[8] Francesco Lattari et al. “Deep Learning for SAR Image Despeckling”. In: *Remote Sensing* 11 (June 2019), p. 1532. DOI: 10.3390/rs11131532.

[9] G. Chierchia et al. *SAR image despeckling through convolutional neural networks*. 2017. arXiv: 1704.00275 [cs.CV].

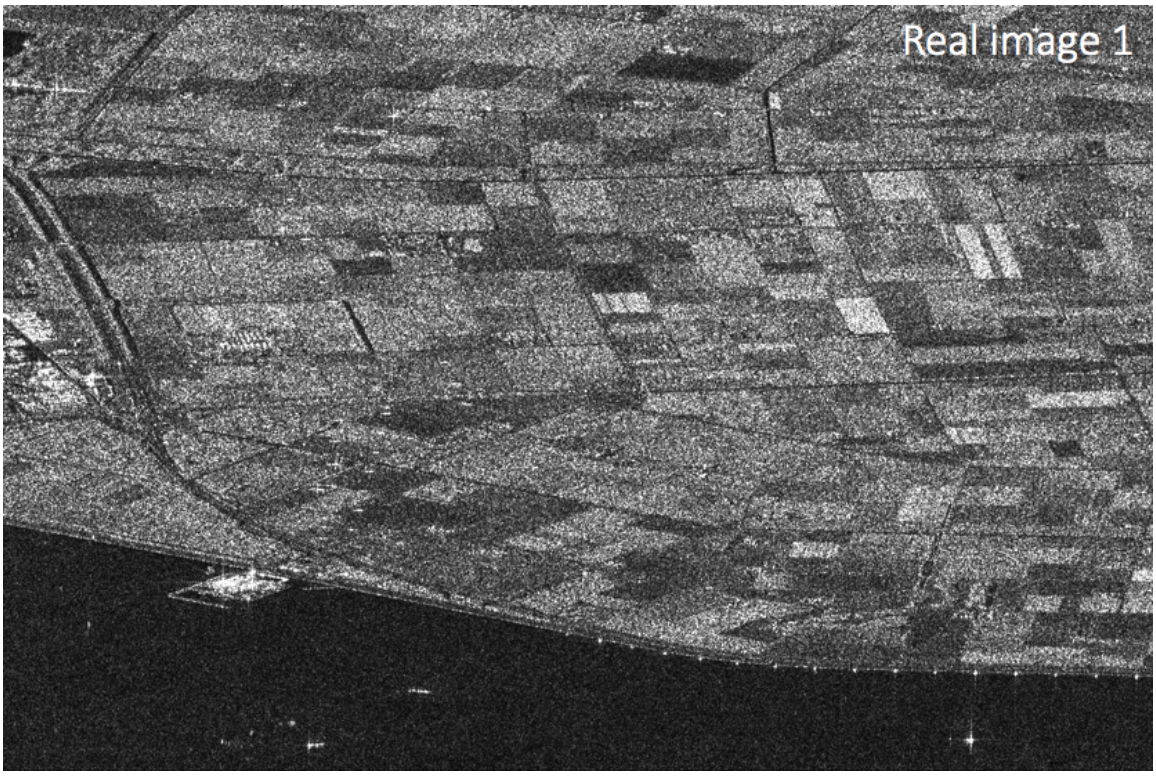
[10] Qiang Zhang et al. “Learning a Dilated Residual Network for SAR Image Despeckling”. In: *Remote Sensing* 10 (Feb. 2018), pp. 1–18. DOI: 10.3390/rs10020196.

[11] H. Zhao et al. “Loss Functions for Image Restoration With Neural Networks”. In: *IEEE Transactions on Computational Imaging* 3.1 (2017), pp. 47–57.

[12] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612.

[13] L. Zhang et al. “A comprehensive evaluation of full reference image quality assessment algorithms”. In: *2012 19th IEEE International Conference on Image Processing*. 2012, pp. 1477–1480.

[14] Kai Zhang, Wangmeng Zuo, and Lei Zhang. “FFDNet: Toward a Fast and Flexible Solution for CNN-Based Image Denoising”. In: *IEEE Transactions on Image Processing* 27.9 (Sept. 2018), pp. 4608–4622. ISSN: 1941-0042. DOI: 10.1109/tip.2018.2839891. URL: http://dx.doi.org/10.1109/TIP.2018.2839891.



(a) Real, noisy image

(b) Denoised image, with uniform noise map

Figure 13: Example of our result on a real image