école normale supérieure paris-saclay

MVA Master

Remote Sensing
Final project report

FFDNet for SAR despeckling

Authors:
Lucas Elbert
Bjoern Michele

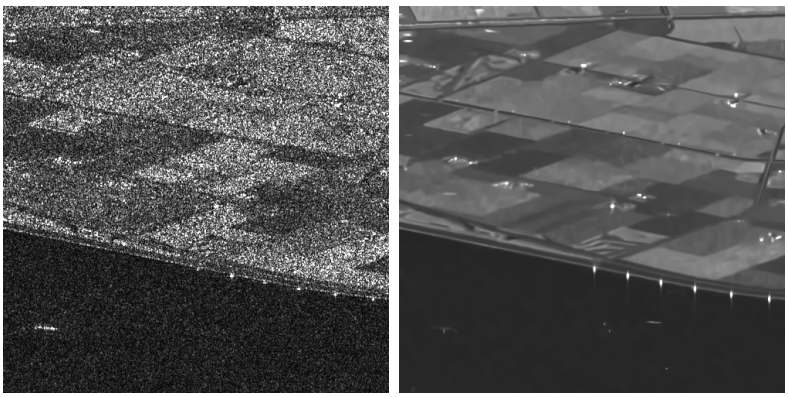
Supervisors:
Florence Tupin
Emanuele Dalsasso

April 3, 2020

1 Introduction

Synthetic-aperture radar (SAR) imaging is an imaging technology intensively used in earth observation as it allows high quality imaging equally at day and night and is furthermore more robust to weather conditions than optical imaging. However it also has its difficulties one of the major ones being the *speckle phenomenon*, which has an appearance similar to noise and poses problems to the image interpretation. In this project we evaluate different approaches of using the FFDNet [3] for a despeckling of SAR images.

The report is organized as follows: Section 2 gives a brief introduction to SAR images and the speckle phenomenon, section 3 introduces the FFDNet with its specificities and is followed by section 4 which explores an alternative training approach when no ground truth data is available. Section 5 describes experiments and shows and discusses results whereupon section 6 shows some caveats when operating on real SAR images. Finally section 7 summarizes the findings of this project and gives indications on further work.



(a) real image (b) desired image

Figure 1: Visualization of the speckle phenomenon

2 SAR images and speckle

To acquire a SAR image a radar emits electromagnetic radiation and measures the from the target reflected waves. Typically it measures the intensity of the backscattered wave. As a radar has a finite resolution, the measured wave in fact is a combination of interfering waves backscattered from everything inside a resolution cell. As the ground usually is extremely rough compared to the wavelength, the measured wave is heavily influenced by the cell's internal structure. Because the internal structure varies from cell to cell and there is no knowledge about it, the measured wave is subject to *seemingly* random influences. For this reason some cells/pixels appear brighter and others darker. This is the so called speckle.

2.1 Distributions

To model the speckle phenomenon the Goodman model describes the measured intensity I as a random variable with expectation R. R is called the reflectivity and represents the hidden desired value. Another important measure is the amplitude $A = \sqrt{I}$. Motivated by the probability distributions we call images with intensity or amplitude values to be in the *multiplicative domain*, whereas logarithmic intensity and logarithmic amplitude are called the *additive domain*. To retain a maximum of resolution in the image this document always works in the single look case.

2.1.1 Intensity

In the single look case the intensity (I) follows an exponential distribution with expectation R.

$$I|R \sim Exp(1/R), \quad p(I|R) = \frac{1}{R}e^{-1/R}, \quad \mathbb{E}[I|R] = R \tag{1}$$

This can also be written as

$$I = R \cdot S, \quad S \sim Exp(1) \tag{2}$$

With a constant reflectivity R and a random variable S to represent the speckle. This equation shows the speckle's multiplicative nature.

2.1.2 log-intensity

To transform the multiplicative speckle into an additive phenomenon a logarithm can be applied. The log-intensity follows a Fisher-Tippet distribution.

$$\ln(I) = \ln(R) + \ln(S) \tag{3}$$

$$p(\ln(I)|\ln(R)) = e^{\ln(I)-\ln(R)}e^{-e^{\ln(I)-\ln(R)}} \tag{4}$$

$$\mathbb{E}[\ln(I)|\ln(R)] = \ln(R) + \psi(1) \tag{5}$$

It should be noted that the log-intensity is biased.

2.1.3 Amplitude

The single look amplitude ($A = \sqrt{I}$) follows a Rayleigh distribution.

$$p(A|R) = \frac{2A}{R}e^{-A^2/R} \tag{6}$$

$$\mathbb{E}[A|R] = \sqrt{\frac{\pi R}{4}} \tag{7}$$

Once again the amplitude's bias should be noted.

3 FFDNet

The FFDNet of [3] is a neural network approach to remove spatial variant additive white Gaussian noise in optical images. Additive white Gaussian noise is added to the original signal image u as shown in formula 8. The variance σ^2 of the normal distribution modelling the noise, can obviously vary depending on several factors.

$$v = u + \epsilon \quad \text{with } \epsilon = N(0, \sigma^2) \tag{8}$$

One limitation of existing denoising approaches is that they are only able to denoise images with the same noise level or in a small range of noise levels. One important feature of the FFDNet is therefore,that given the noise level of an image, the network should be able to denoise images with a large noise level range and it can be adaptively controlled the trade-off between details preservation and noise-reduction. The balance between these two points can be varied between images but also on a spatial level in one image.

In opposite to other denoising methods, the FFDNet should learn a direct mapping function F from the noisy image y and the per pixel noise level map M to the denoised output $\hat{x}$ as it is

shown in formula 9. This direct mapping function F should be learned by a CNN architecture with the learnable weights Θ. The complete architecture of the FFDNet is shown in figure 2.

$$\hat{x} = F(y, M, ; \Theta) \tag{9}$$

The loss function used for the learning with a stochastic gradient descent for a batch of size N is given in formula 10.

$$L(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} ||F(y_i, \sigma, ; \Theta) - x_i||^2 \tag{10}$$

Besides the additional input of the noise level map the FFDNet also uses directly in the beginning a reversible downsampling operator such that instead of the single input image 4 downsampled subimages together with the noise level map are used. Important about this downsampling is, that no information is lost as the input image with the size $W \times H \times C$ is only rearranged in four smaller subimages such that all four subimages together have a size of $\frac{W}{2} \times \frac{H}{2} \times 4C$. Each of the 4 subimages has 1 out of 4 of the original pixels, which are sampled, such that each of the subimages appears as a downsampled version of the original input image. The authors of the FFDNet paper state, that this downsampling should reduce the computation time and memory costs, as through the downsampling the convolutional kernels have a larger receptive field (in the original image) but don't need so many learnable parameters. This allows an efficient model with a large receptive field. As additional regularization for the convolutional layer, orthogonal regularization is used for the convolution kernels which should improve generalization ability of the network. The architecture of the FFDNet network consists of 12 (color image) or 15 (gray scale image) convolutional layer which use besides of the output layer the ReLU activation function. The convolutional layer in the middle of the network are also completed with a batch normalization layer which should accelerate the learning process [1]. As the FFDNet is a fully convolutional network it can be trained and used with images of different sizes.

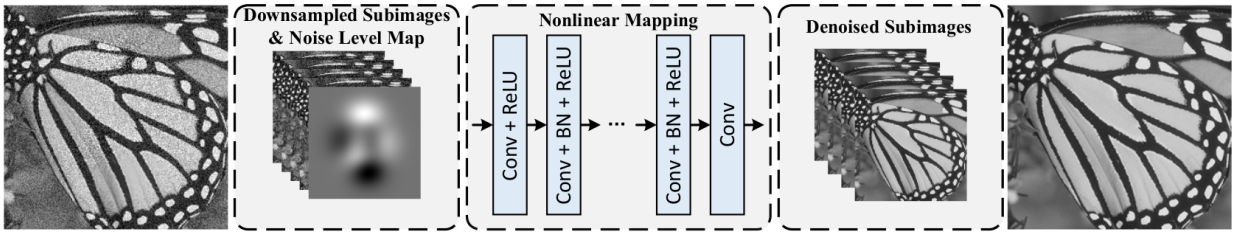


Figure 2: FFDNet architecture.

4 Noise2Noise

In some applications there might be a lack of denoised ground truth images but the possibility to acquire several noised images of the same ground truth. For these cases the Noise2Noise [2] offers a framework to train a network based on noised images only.

When having a θ-parameterized network f_θ and a loss function l, pairs (x, y) of ground truth images (x) and a noised version (y) the "standard" neural network approach is to optimize:

$$\underset{\theta}{argmin} E_{(y,x)}[l(f_\theta(y), x)] = \underset{\theta}{argmin} E_y[E_{x|y}[l(f_\theta(y), x)]] \tag{11}$$

The Noise2Noise setting now basically replaces x by another noisy version $\tilde{y}$ of x. And optimizes

$$\underset{\theta}{argmin} E_y[E_{\tilde{y}|y}[l(f_\theta(y), \tilde{y})]] \tag{12}$$

Where only the noisy data $y, \tilde{y}$ are known and x is unknown but existing. With some additional assumptions to the distributions and loss function the optimization problems (11), (12) are equivalent.

4.1 L_2-loss

When optimizing (12) with l being the L_2-norm an optimal solution would be $f_{\theta^*}(y) = \mathbb{E}[\tilde{y}|y]$. With the subtle additional assumption that all $\tilde{y}$ in the training set are generated from the same ground truth x this simplifies to

$$f_{\theta^*}(y) = \mathbb{E}[\tilde{y}|x] \tag{13}$$

The L_2-loss can therefore directly be applied if the noise distribution is unbiased. Otherwise an debiasing step is needed.

4.1.1 Amplitude

The single-look speckle amplitude expectation is $\mathbb{E}[A|R] = \sqrt{\pi R/4}$ and therefore a meaningful L_2-loss function with incorporated debiasing step is:

$$l(f_\theta(A), \tilde{A}) = ||f_\theta(A) - \frac{4\tilde{A}^2}{\pi}||_2^2 \tag{14}$$

4.1.2 log-intensity

The single-look speckle log-intensity expectation is $\mathbb{E}[\ln(I)|\ln(R)] = \ln(R)+\psi(1)$. A meaningful L_2-loss function with incorporated debiasing step is:

$$l(f_\theta(\ln(I)), \ln(\tilde{I})) = ||f_\theta(\ln(I)) - \ln(\tilde{I}) + \psi(1)||_2^2 \tag{15}$$

4.2 Fisher-Tippet-loss

To be closer to the speckle phenomenon one can also try to optimize (12) with the negative log-likelihood of the observations corresponding to the distributions given in section 2.1 as a loss function. For the Fisher-Tippet distribution this is:

$$\begin{aligned} l(f_\theta(y), \tilde{y}) &= -\ln(p(\ln(I) = \tilde{y}|\ln(R) = f_\theta(y))) && (16)\\ &= -\tilde{y} + f_\theta(y) + e^{\tilde{y} - f_\theta(y)} && (17) \end{aligned}$$

In fact when optimizing (12) with that loss as Emanuelle Dalsasso has shown, the optimum is achieved at $f_{\theta^*}(y) = \ln(\mathbb{E}[e^{\tilde{y}}|y])$.

4.2.1 log-intensity

Again with the subtle assumption that each $\tilde{y}$ is only generated from a single ground truth image and knowing that the intensity is unbiased ($\mathbb{E}[I|R] = R$) a meaningful loss now simply is:

$$l(f_\theta(\ln(I)), \ln(\tilde{I})) = -\ln(\tilde{I}) + f_\theta(\ln(I)) + e^{\ln(\tilde{I}) - f_\theta(\ln(I))} \tag{18}$$

5 Experiments

5.1 Dataset

Our basis dataset consists of 7 large SAR amplitude images (size: lely 1536×1024, limagne 4096×1024, marais1 4096×1024, marais 4096×1024, ramb 8192×512, risoul 2048×512, saclay 768×1024). These 7 images are already denoised. Additionally, we also have all images in a not denoised version, with real speckle noise. For the training we use the denoised SAR images, and add artificial speckle noise as described in formula 19 in order to generate several possible realisations. $rand_a$ and $rand_b$ are independent drawings of the standard normal distributions. During the training randomly 250×250 patches are cropped out of the large images, and the artificial speckle noise is added. In one "epoch" of the training 30 patches are randomly cropped out of every image. The random cropping ensures, that many different crops are used for the training. Our test data set consists of 7 smaller images, each with a size of 500×500. The test scenes are not part of the training scenes, but are similar to them. During the evaluation we always crop the same 100 subimages out of these test images and add artificial speckle noise. So, in each evaluation process we consider the same image crops.

$$R = A \times \sqrt{S} \text{ with } \sqrt{S} = \sqrt{\frac{|(rand_a + rand_b\ i)|^2}{2}} \tag{19}$$

5.2 Evaluation metrics

In terms of quantifiable image quality we evaluate our results with the Peak-Signal-to-Noise ratio (PSNR) and the ratio between the noisy images and the predicted denoised images. The PSNR is calculated with the formula:

$$\text{PSNR} = 10 \log_{10}(\frac{\text{MAX}_R^2}{\text{MSE}}) \tag{20}$$

with the maximal value MAX of the denoised ground truth image and the mean squared error MSE between the predicted denoised image and the denoised ground truth image. The PSNR is a metric which measures the image quality of a denoised (or noisy) image in relation to the correctly denoised image. The higher its value the better. As it is often criticised that it is not given that the PSNR values closely correlate with the human visual perception, we additionally also do a manual inspection of the denoised images in order to qualitatively evaluate the denoising.

Another quantitative metric is the ratio between the noisy input image and the predicted denoised image (= predicted reflectivity):

$$S_{ratio} = \text{ratio} = \frac{\text{noisy input}}{\text{predicted reflectivity}} \tag{21}$$

The ratio can also be interpreted as the speckle noise in the multiplicative domain. With this ratio, we want to see, if the predicted reflectivity, given the noisy input, is consistent with our model of the speckle noise. This especially means: $E[S_{ratio}] = 1$ and $\sigma_S = 1$ and an exponential distribution of the speckle noise. Therefore, we compare the calculated ratio with these expected characteristics. The results of the additive domain are transformed back to the multiplicative domain for the evaluation metrics, such that the results can be compared between those two domains.

5.3 Training configurations

In our experiments we investigate different training configurations in order to see which combinations are efficient and which advantages or disadvantages each of the configurations has. The overview over the different possible settings for each of the parameters can be seen in figure 3. Each of them will also be explained in detail in this section.

We consider two large approaches, the first one is to use the FFDNet and train the denoising

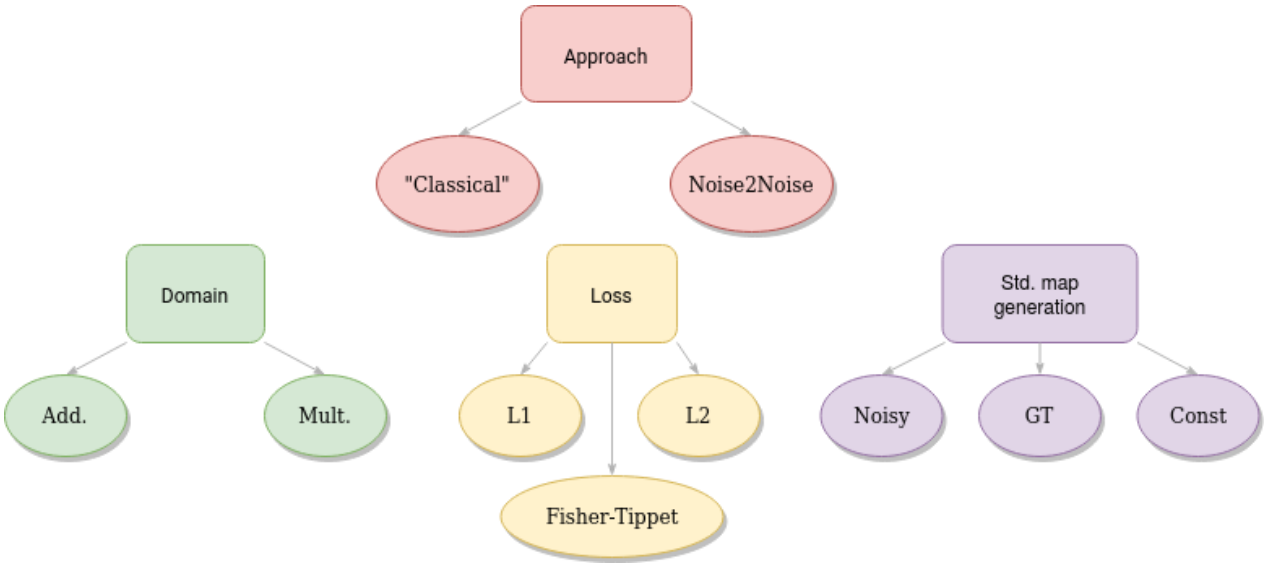


Figure 3: Overview training configurations.

similar to the denoising of the optical images, by comparing the output of the network with the denoised ground truth image. For this approach we experiment with different combinations of the way we treat the noise (multiplicative or additive in the logarithmic domain), the loss functions and the calculation of the noise level map that will be explained later in detail. Our second approach is to use the Noise2Noise framework together with the FFDNet. With this approach the denoised output of the FFDNet will not be compared with the denoised ground truth data, but to another independent noisy image of the same scene. For the Noise2Noise framework we also investigate different combinations of the working domain, loss functions and the calculation of the noise level map that will be described later.

In both approaches we tested the denoising in the two domains: the multiplicative domain, with the speckle as a multiplicative noise, and the additive domain, with the speckle as an additive noise. The motivation for the transformation in the additive domain is that the FFDNet has been designed and tested only for additive white Gaussian noise. We expect that by transforming the speckle noise from the multiplicative domain to the additive domain it is more similar to the additive white Gaussian noise.

The loss function that should be minimized during the training is an important factor for this optimization problem. The standard loss function that is used for the FFDNet is the L2-loss. Therefore, we also use this loss function for our case. Additionally, we also try out the L1-loss as we expect sharper and more homogeneous surface results as the L1-loss does not over-weight large differences but instead provide even for small differences the same gradient. In the paper [4] they also state that the L1-loss provides a better convergence for tasks as image denoising and get less likely stuck in local minima. They additionally provide some empirical results, that the L1-loss produces less artifacts in image restoring tasks than the L2-loss. Besides these two loss functions, we also used the described Fisher-Tippet-loss for the Noise2Noise framework, which should better model the speckle noise distribution.

The last configuration parameter is the calculation of the noise level map. The noise level map

is an additional input for the FFDNet. In the additive domain the variance of the speckle noise is constant and can be calculated independently of the noisy image. In the multiplicative domain the standard deviation of the noise depends on the denoised values of the amplitude images. Due to the Goodman model we know that in the theory the expectation and the standard deviation for a physically homogeneous area are connected with the formula $\sigma_A = 0.53 \cdot E(A)$. During the training the standard deviation, which is used for the noise level map, can be approximated with this formula. The expectation for the homogeneous area is replaced by the mean of a local window. We experiment with different combinations of the calculation of this mean value. Important is, on which image the mean of the local window is calculated. One option to do this, is to calculate it on the noisy image. Of course this would not give the correct value, but it could be a good approximation that is easy to calculate as during the real inference only the noisy image exists. Another option, at least for the training or to develop the model, would be to calculate the standard deviation based on the denoised ground truth image. This would lead to better or (for a window size of 1) even the correct standard deviation value. We also do experiments with this approach to see if it has positive effects to use this only during the training, and in order to see if a better approximation of the noise level map could be helpful. However, the results of using the denoised ground truth for the noise image should be always interpreted with the knowledge that it could be also the data leakage (for window size 1 we are even sure that it is data leakage) that is responsible for the result, as parts of the ground truth are given with the noise level map as input.

5.4 Results and discussion

In this section we show and discuss the results of the experiments. We begin with a detailed description of individual results and conclude with a broader summary.

Table 1 shows quantitative results of our previously described experiments as average PSNR values and a mean ratio. The input to the network, which was the artificially noised ground truth, consistently had a PSNR value of 12.0. The average PSNR on the test set was between 20.7 and 24.61 for the different training approaches. We note here that this shows an average PSNR value over a test set of 100 images whilst the PSNR-performance showed high variance over the different ground truth images. The variance over different speckle realizations was small. Nevertheless, all networks always significantly increased the image quality and we observed good learning curves for all practical relevant configurations. They showed a smooth network convergence and no sign of overfitting. Furthermore, as in most cases the mean ratio is close to 1.0, most networks seem to be unbiased.

In experiments one and two (rows and and two of the table) we observe an exceptional training behavior as they always continued to learn and were only stopped by a time limit. This can be explained as in these experiments the ground truth data was used for the std-map estimation which can be considered as a data leakage. Though useless for practical work on the first sight, these experiments could motivate an iterative denoising approach of using a first denoised prediction just for std-map estimation. Experiments three and four performed worse and showed a bias that is also visible in the mean ratio. This can be explained by the different ways of estimating the std-map (from ground truth during training and from noisy data whilst testing). Experiments five and six showed unbiased good performances and are also practically relevant configurations as the std-map was only estimated from noisy data. These are among our best results. In experiments seven and eight we worked in additive domain where the speckle is additive. These experiments showed an unbiased denoising behaviour but surprisingly gave lower PSNR values than the multiplicative configurations. The last three rows of the table (experiments nine, ten, eleven) show the Noise2Noise experiments. The mean ratio

in the table shows no bias and the PSNR performance is comparable to the networks trained with ground truth data. With the usage of Fisher-Tippet-loss as described in section 4 it was in terms of PSNR value the best configuration from these experiments.

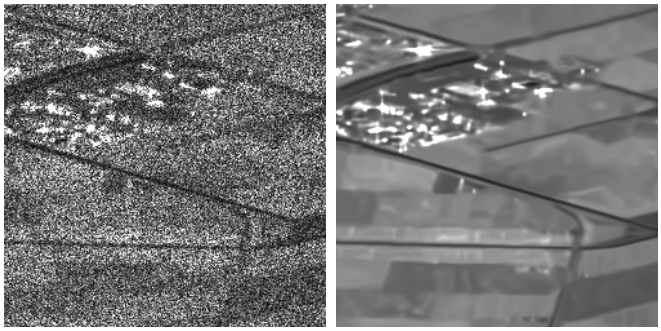
	Configuration	Loss	PSNR (pred)	PSNR (noisy)	Mean ratio
1	Multiplicative Std. (Train: GT, Test: GT), Ws: 7	L1	24.5	12	0.97
2	Multiplicative Std. (Train: GT, Test: GT),Ws: 7	L2	25.0	12	0.99
3	Multiplicative Std. (Train: GT, Test: Noisy), Ws: 7	L1	20.7	12	1.25
4	Multiplicative Std. (Train: GT, Test: Noisy),Ws: 7	L2	22.00	12	1.22
5	Multiplicative Std. (Train: Noisy, Test: Noisy), Ws: 7	L1	**24.26**	12	0.99
6	Multiplicative Std. (Train: Noisy, Test: Noisy), Ws: 7	L2	23.44	12	0.92
7	Additive Std. (const.)	L1	21.85	12	1.03
8	Additive Std. (const.)	L2	22.97	12	0.98
9	Multiplicative Noise2Noise Std. (Train: Noisy, Test: Noisy)	L2	22.76	12	1.04
10	Additive Noise2Noise Std. (const)	L2	22.77	12	0.98
11	Additive Noise2Noise Std. (const)	Fisher-Tippet	**24.61**	12	0.96

Table 1: Results of different training configurations.

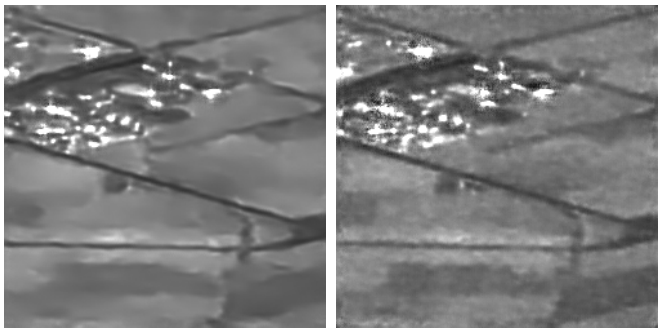
Figure 4 shows qualitative results of experiments 1-8 on an example image from the test set. Figure 5 shows the corresponding outputs from Noise2Noise trainings. We can say that all networks produced a meaningful output but the denoised images look significantly blurred compared to the ground truth. The images denoised with a network trained on L1-loss generally show smoother outcomes on homogeneous areas like the crop fields which was not visible from the PSNR values. We can compare the multiplicative vs. additive domain by looking at the two bottom rows ((g)-(j) corresponding to experiments 5-8). It seems to us that working in logarithmic/additive domain softens the difference of L1 and L2-loss. Also when working in additive domain with L2-loss ((h)) we observe artefacts around high response areas that might be due to the std-map estimation with sliding windows. Working in logarithmic domain helped suppressing those.

Figure 5 shows qualitative results of the Noise2Noise experiments for the same test image. Overall the results look similar to what was produced in the 'classical' approaches in figure 4. We can see that similar to the 'classical' approach, working in additive domain has a better behavior regarding artefacts around high response areas. Furthermore the Fisher-Tippet-loss produces an output that is slightly more homogeneous over constant areas like the crop fields.

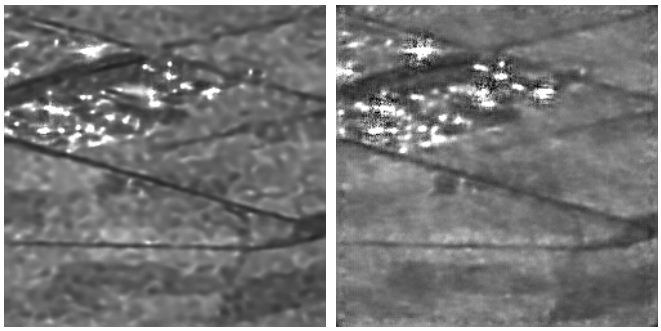
Figure 6 shows exemplarily the ratio of the in figure 4 denoised image for experiment (8). It shows the quotient 'input'/'output' in intensity. We see that the ratio's histogram matches



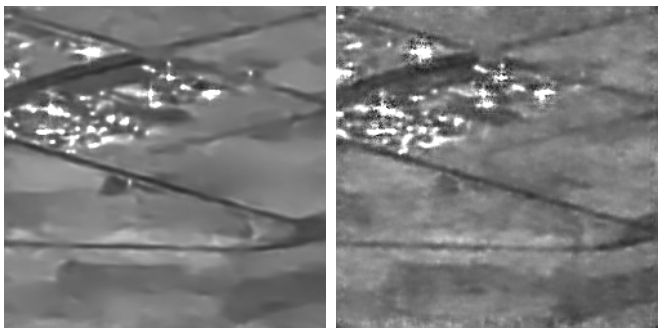
(a) Input (b) Ground Truth



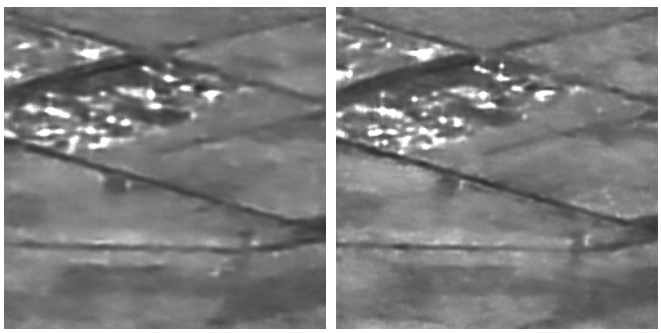
(c) Mult., L1, GT, GT (d) Mult., L2, GT, GT



(e) Mult., L1, GT, Noisy (f) Mult., L2, GT, Noisy

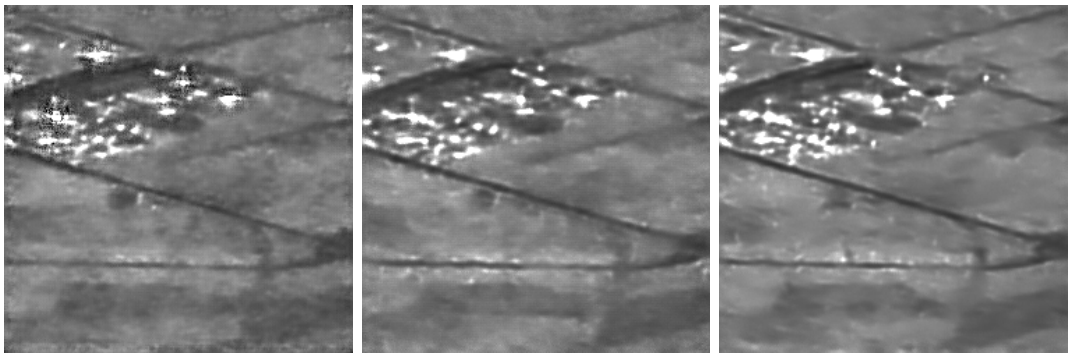


(g) Mult., L1, Noisy, Noisy (h) Mult., L2, Noisy, Noisy



(i) Add., L1, const. (j) Add., L2, const.

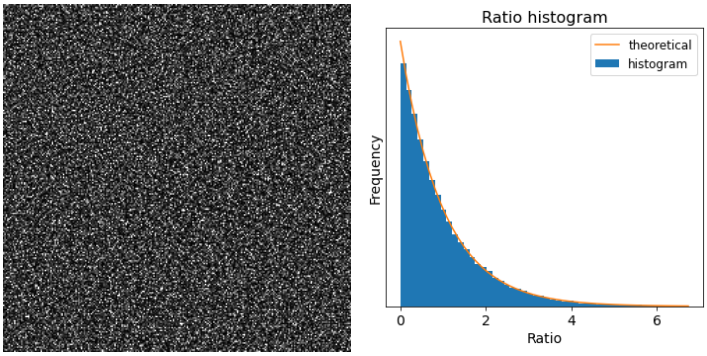
Figure 4: Qualitative comparison of different configurations.



(a) Mult. L2, Noisy, Noisy (b) Add. L2, const. (c) Add. Fisher-Tippet, const.

Figure 5: Qualitative comparison of the Noise2Noise configurations (also compare figure 4).

the theoretical ratio distribution very well. We manually checked many histograms and found that in all experiments where the mean ratio was close to 1.0 the histograms matched well with the theoretical distribution.



(a) Ratio (b) Histogram

Figure 6: Ratio of the image from figure 4, denoised with experiment 8/(j).

To summarize the results we can say that all performed experiments produced meaningful denoised outputs and were mostly unbiased as the ratio histograms showed. On the downside the outputs were still blurry compared to the ground truth. We observed the L1-loss to produce more homogeneous output over constant areas than the L2-loss. Working in the multiplicative domain mostly gave better PSNR values than in additive domain, though working in additive domain damped artefacts around high response areas. Furthermore we found the denoising performance in terms of PSNR values to be highly dependent on the ground truth image (more than on the concrete speckle noise). Lastly in our experiments the Noise2Noise training was able to produce equally good results as the 'classical' approach and the use of a speckle related loss function ('Fisher-Tippet-loss') increased the PSNR performance.

5.5 Comparison to SAR-CNN and U-Net

In order to see how our best performing configurations perform compared to other methods, we run our methods also on a special evaluation data set. For this evaluation data set we have the results generated by the SAR-CNN and a U-Net with a Noise2Noise framework. We compare the performance of our configurations quantitatively as well as qualitatively with this reference solution. The evaluation data set consists of 7 different SAR amplitude images of

Method	PSNR (denoised)	PSNR(noisy)	ratio
SAR-CNN	**25.59**	12.19	0.988
U-Net (Noise2Noise)	25.13	12.19	0.998
Mul. L1 (FFDNet)	24.94	12.19	0.99
Noise2Noise Add. Fisher-Tippet (FFDNet)	22.25	12.19	0.956

Table 2: Comparison between different denoising methods on SAR amplitude images.

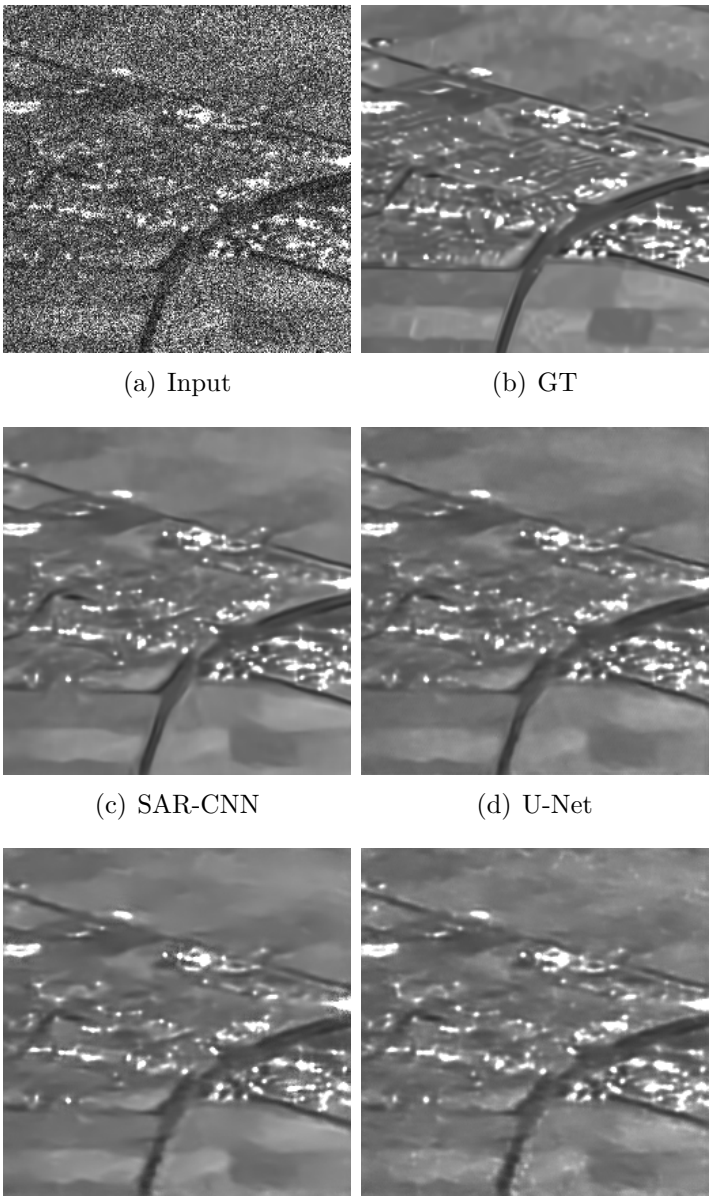
size 256×256. For each of the different images we have 20 different realizations of the speckle noise. The PSNR results as well as the ratio mean values for the different methods can be seen in table 2. The two other methods outperform our configurations slightly in the PSNR values. However, it can be seen that the results with the FFDNet are of the same magnitude than the other methods and it therefore could be worth to further investigate the denosing of SAR images with the FFDNet. Looking at the denoised outputs in figure 7 or in figure 9 in the appendix, it can be also seen qualitatively, that the FFDNet methods generate similar solutions, but which are slightly worse than the other methods.

While the Noise2Noise approach in the additive domain achieves a PSNR value of 24.61 on our own test data set, which is also close to the performance of the best classical approach, it surprises a bit that it achieves on this evaluation set with very similar SAR images, a 2 dB worse result. An explanation for this could be that, as already mentioned in the section 5.4, the achieved PSNR values are very dependent on the ground truth image. As this data set consists of only 7 ground truth images (with each 20 realizations of the speckle noise) it is perhaps just bad luck that on these seven images our Noise2Noise approach perform worse in average, compared to our own test data set where we average over 100 different image crops.

6 Real images

As described in section 5.1, the training and testing performed so far were always done on images with synthetically inserted speckle noise. The artificial speckle noise followed distributions seen in section 2.1 and were based on the Goodman model. One of the assumptions of the Goodman model is pixel wise independence of the speckle. In real conditions however this assumption does not hold and the theoretical speckle can only be seen as an approximation. It is therefore an interesting question how the on artificial data trained FFDNet behaves on real SAR images.

Table 3 shows numerical results of experiments where we applied two nets that were trained on artificial data on real SAR images. Figure 8 shows visual results on the real images for the network trained in multiplicative domain. The top row of figure 8 and lines one and three of table 3 show that the networks had problems adapting to real images. The PSNR values are much lower than on artificial data and on the image we can see left over artefacts in form of a shimmer. One possible technique in tackling this problem lies in subsampling the input before processing as the subsampling is expected to sufficiently decorrelate adjacent pixels. The second row of figure 8 shows such a processing where the input image was subsampled by a factor two before being fed to the network. The output is much more appealing but the subsampling comes with the big disadvantage of loss in spatial resolution. Although the PSNR values increased by the subsampling they are still lower than on artificial data. It would be interesting to investigate more into the nature of the correlation in SAR images and possibly directly train a network on real SAR images or independently denoise the disjoint subsampled images and recombine them afterwards.

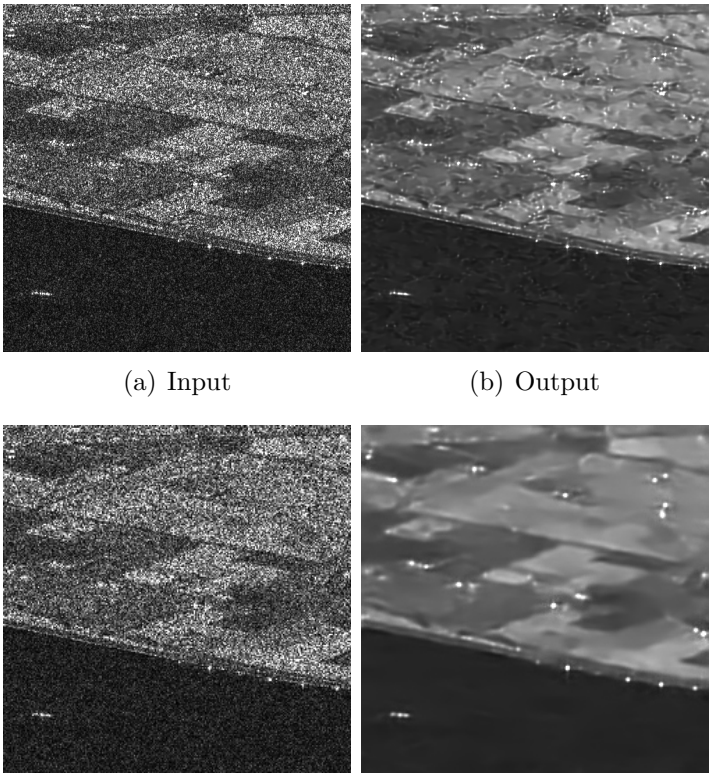


(a) Input (b) GT

(c) SAR-CNN (d) U-Net

(e) Mul. L1(FFDNet) (f) N2N. Fisher-Tippet (FFDNet)

Figure 7: Qualitative comparison of the different methods.



(a) Input (b) Output

(c) Subsampled input (d) Output of subsampled input

Figure 8: FFDNet on real SAR images. Top row: FFDNet application on an unmodified real SAR image (corresponds to line one in table 3), bottom row: the same with a factor two subsampling of the input (corresponds to line two in table 3). These images are crops of a larger image that can be found in the appendix.

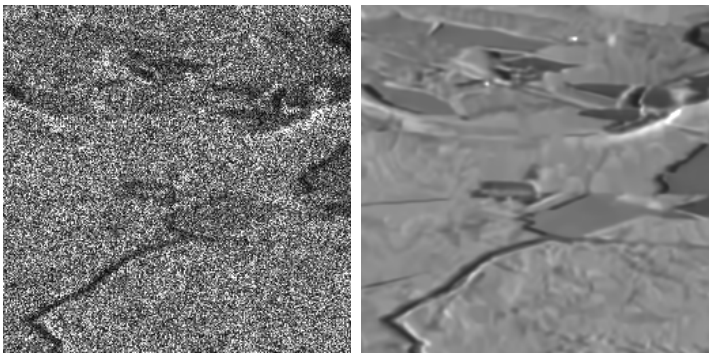
Method	PSNR (denoised)	PSNR(noisy)	ratio
Mult., L1, std. Noisy, Noisy	16.80	9.66	0.99
Mult., L1, std. Noisy, Noisy Downsampled	17.33	9.66	0.99
Add., Noise2Noise Fisher-Tippet loss	14.84	9.66	0.97
Add., Noise2Noise Fisher-Tippet loss Downsampled	16.66	9.66	1.02

Table 3: Results of the on artificial data trained FFDNet applied onto real SAR images.

7 Conclusion and further work

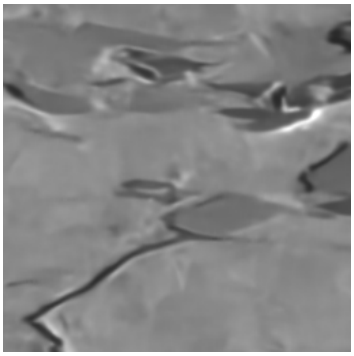
We have shown that the FFDNet can not only be used to denoise optical images, but can also be used to denoise SAR amplitude images. In a supervised manner the best results with the FFDNet can be achieved in the multiplicative domain. Especially interesting for cases where no ground truth data exists is also, that at least comparable denoising results can also be obtained in an unsupervised manner with the Noise2Noise framework. Furthermore, our experiments with different training configurations could give an overview of how the different configuration decisions could influence the result. We also have shown that with our approach it is not possible without a large drop in the performance to use the trained FFDNet to denoise real speckle noise. Therefore, a logical next step would be to train our approaches on real speckle noise data. We are confident, that our approaches will also perform very well on the real speckle data as, even if it is not as good as on the artificial data, it is possible to denoise the real speckle images with the models trained on the artificial speckle. For the Noise2Noise approach not even a denoised ground truth data would be necessary but only a second independent aligned image of exactly the same scene. Another interesting approach could be to calculate the noise level map in an iterative manner. Therefore, the noise level map would be calculated on an already from the FFDNet model denoised image. The intuition would be that a more precise noise level map would be generated which could help for a better prediction. Through the flexible noise level map it is also possible to easily extend our approaches to multi-look images. Therefore, it would be interesting to investigate the performance of our approaches on multi-look images. Perhaps it could be even beneficial to train with different multi as well as single look images.

Appendix

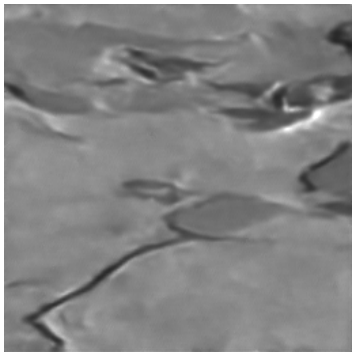


(a) Input

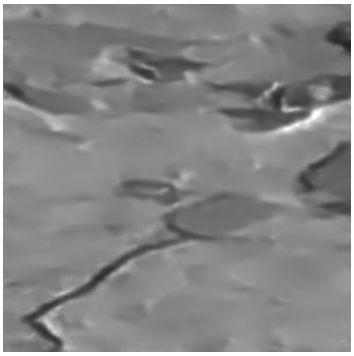
(b) GT



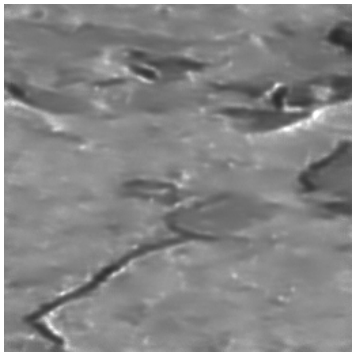
(c) SAR-CNN



(d) U-Net



(e) Mul. L1(FFDNet)



(f) N2N. Fisher-Tippet (FFDNet)

Figure 9: Qualitative comparison of the different methods.

(a) Input

(b) Output

(c) Output of FFDNet on downsampled input

Figure 10: Despeckling of a real SAR image (see section 6).

References

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[2] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. *CoRR*, abs/1803.04189, 2018.

[3] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[4] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017.