Published in Image Processing On Line on 2022–10–07.
Submitted on 2022–09–28, accepted on 2022–09–28.
ISSN 2105–1232 © 2022 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2022.423

Phase Unwrapping using a Joint CNN and SQD-LSTM Network

Roland Akiki[1,3], Carlo de Franchis[1,3], Gabriele Facciolo[1], Jean-Michel Morel[1], Raphaël Grandin[2]

[1]Centre Borelli, Université Paris-Saclay, ENS Paris-Saclay, France
{roland.akiki, carlo.de-franchis, gabriele.facciolo, jean-michel.morel}@ens-paris-saclay.fr
[2]Institut de Physique du Globe de Paris, Université Paris VII, France
grandin@ipgp.fr
[3]Kayrros SAS, France

Communicated by Jean-Michel Morel *Demo edited by* Roland Akiki

Abstract

Phase unwrapping techniques are used in various applications, including Synthetic Aperture Radar (SAR) interferometry (InSAR). Deep learning methods have been recently proposed to tackle this problem. This work aims at explaining and evaluating the method proposed by Perera et al. in [A joint convolutional and spatial quad-directional LSTM network for phase unwrapping, ICASSP 2021]. Furthermore, we provide an online demo to simulate phase images and run them through the network. The network performance can be tested visually and through metrics such as the error standard deviation. The simulation can provide some out-of-distribution data, especially with the added atmospheric signal specific to the InSAR phase.

Source Code

The source code and documentation for this algorithm are available from the web page of this article[1]. Usage instructions are included in the `README.txt` file of the archive. Some pieces of the code were copied from the original author's repository[2].
This is an MLBriefs article, the source code has not been reviewed!

Keywords: phase unwrapping; CNN; LSTM; deep learning; InSAR; demo

[1]https://doi.org/10.5201/ipol.2022.423
[2]https://github.com/Laknath1996/DeepPhaseUnwrap, commit hash: 9316c1b1272c0457000fbbaaff850303d923df88

1 Introduction

In many signal processing applications, the quantity of interest is a phase, i.e., an angle that can only be observed in the $[-\pi, \pi]$ interval. Phase unwrapping refers to the process of retrieving the true unseen phase signal, knowing that it can span any arbitrary range. Phase unwrapping is an ill-posed problem since it can have an infinite number of solutions. Nevertheless, it is an essential processing step for many interferometric measurement techniques, such as Synthetic Aperture Radar (SAR) interferometry (InSAR) [4], magnetic resonnance imaging [11], and optical interferometry [8].

In order to solve the problem, it is necessary to make some assumptions about the structure of the solution, in particular stating that the actual phase varies slowly along the unwrapping dimension. This assumption is often referred to as the Itoh condition [3]. For a 2D image, according to the Itoh condition, the actual phase difference between neighboring pixels does not exceed π. Using this condition, the true phase gradients in the row and column direction are often approximated from the wrapped phase image gradients. However, in the presence of noise or of a fast varying signal (aliasing of the actual phase) or a discontinuity in the unwrapped signal (for instance, the InSAR phase of a tectonic fault rupturing the surface during an earthquake), the Itoh condition does not hold. Solving this problem is the primary motivation of phase unwrapping. Several approaches exist.

"Path following" methods try to identify the regions where the Itoh condition has potentially failed, and choose a reliable integration path by avoiding these regions. The branch cut method [4], and the quality guided methods such as [5] belong to that category. Most optimization-based techniques minimize the L^p norm of the difference between the true phase gradient and the rewrapped gradient of the wrapped image. If $p = 1$, this corresponds to the minimum cost flow method [2]. For p=2, the least-squares method can be solved efficiently with implementations based on the Fast Fourier Transform (FFT) [10]. Also under the umbrella of optimization techniques are statistics-based methods like SNAPHU [1].

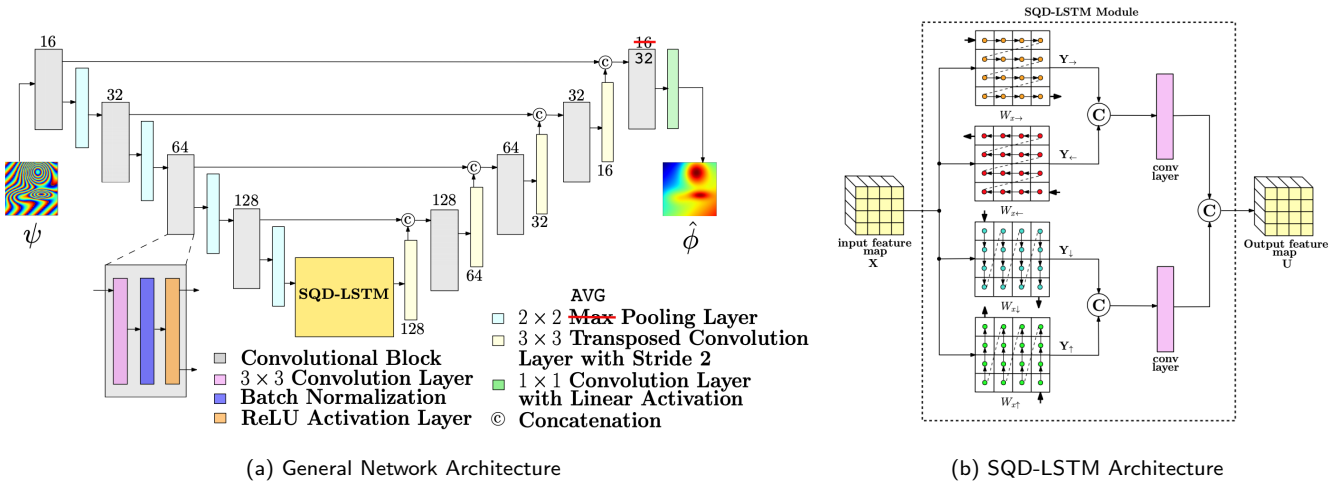
More recently, several works using deep learning have started to tackle the problem of phase unwrapping [14]. Some deep networks are trained to assist previous classical methods and improve some of their aspects. For instance, in [15], a deep neural network is used to predict the branch-cuts, i.e., the regions that the integration process should not cross. Therefore, the neural network improves and assists the method in [4]. Other networks [13] predict the phase gradients and are mainly followed by an L^p-norm unwrapping step. On the other hand, several deep learning works have attempted to reconstruct the unwrapped phase directly from the wrapped phase image, or to predict the ambiguity number, i.e., the integer multiple of 2π that needs to be added to the wrapped phase in order to retrieve the unwrapped solution. One of the best works that follow this methodology is Phasenet 2.0, detailed in [12].

The article [9] analyzed here tackles the problem of phase unwrapping using deep learning. In particular, its objective is to predict the unwrapped phase directly from the wrapped phase image as a regression problem. The proposed network is composed of a novel convolutional architecture, the Convolutional Neural Network (CNN) encoder-decoder architecture, that incorporates a Spatial-Quad Directional - Long short-term memory (SQD-LSTM) module. It is one of the most recent works on the subject and claims to have the best performance in terms of robustness to severe noise conditions and computational efficiency.

In the following sections, we present the details of the method, including network architecture, simulation of the training data, and the training process. We test the network performance and show some results in the experimental section. The final section provides a complete guide to the online demo.

2 Method

2.1 Network Architecture



(a) General Network Architecture

(b) SQD-LSTM Architecture

Figure 1: Figures taken from the original article [9] and slightly modified. The network architecture is shown. The network takes a wrapped image as input and outputs the unwrapped result. Feature size (number of convolutions) is indicated at the top/bottom of the blocks. Max Pooling has been replaced by Average pooling to match the source code. The last convolutional block feature number has been updated to match the source code. The SQD-LSTM module is also shown.

The network architecture used is shown in Figure 1a. It consists of a convolutional encoder-decoder structure. The output of the encoder is fed to the decoder through an SQD-LSTM module. Each convolutional block shown in grey consists of a 3×3 convolution layer followed by a batch normalization operation and Rectified Linear Unit (ReLU) activation. In the encoder section, the number of channels (features) is progressively increased while the spatial resolution is decreased through the use of 2×2 average pooling. In the decoder section, the inverse process occurs as the number of features is decreased, and the image is upsampled with 3×3 transposed convolutions. Skip connections are added to concatenate features from the early stages of the encoder with features of the latter decoder stages. The last convolutional block feature number has been updated in the figure to 32 to match the source code. Finally, the last layer consists of a single 1×1 convolution with linear activation (the identity). It simply combines the 32 features from the previous layer into a single pixel-wise scalar output (the unwrapped result).

As for the SQD-LSTM, the detailed structure of this module is shown in Figure 1b.

Let $\mathbf{X} = x_{ij} \in \mathbb{R}^{w \times h \times c}$ be the input feature map where w, h, c refer to width, height, channel number respectively. Let $x_{\rightarrow}$, $x_{\leftarrow}$, $x_{\downarrow}$, $x_{\uparrow}$ be the sequences obtained when one traverses X from left to right, right to left, downwards and upwards respectively as shown in Figure 1b. Each element of the sequence is a feature vector $x^{(s)} \in \mathbb{R}^c$, where $s \in [1, \cdots, w \times h]$. These four sequences obtained from $\mathbf{X}$ are passed through 4 different LSTM modules separately. The LSTM output is given by,

$$y^{(s)} = \mathbb{LSTM}(x^{(s)}, y^{(s-1)}; W_x, u) \tag{1}$$

where $y^{(s)} \in \mathbb{R}^u$ is the LSTM output at s, W_x the unified weights and u the number of units. Therefore, the four LSTM have weights $W_{x_{\rightarrow}}$, $W_{x_{\leftarrow}}$, $W_{x_{\downarrow}}$, $W_{x_{\uparrow}}$ respectively. The output sequences $y_{\rightarrow}$, $y_{\leftarrow}$, $y_{\downarrow}$, $y_{\uparrow}$ are reordered into feature blocs $\mathbf{Y}_{\rightarrow}$, $\mathbf{Y}_{\leftarrow}$, $\mathbf{Y}_{\downarrow}$, $\mathbf{Y}_{\uparrow}$. The horizontal ($\mathbf{Y}_{\rightarrow}$, $\mathbf{Y}_{\leftarrow}$) and vertical ($\mathbf{Y}_{\downarrow}$, $\mathbf{Y}_{\uparrow}$) directions are concatenated and passed through two different convolutional layers containing d filters each. The final $\mathbb{R}^{w \times h \times 2d}$ is obtained by concatenating the two results from the convolutional layers. For the model at hand, u was set to 32 and d to 64.

2.2 Data generation

The training is done in a supervised manner and requires wrapped phase images with their unwrapped ground truth. Therefore, a simulation is a simple way to get a training dataset. The authors of [9] generated unwrapped images of dimension 256×256 by combining a mixture of Gaussians with a phase ramp and additive centered Gaussian noise. The wrapped phase image $\psi(x, y)$ of the synthetic unwrapped phase image $\phi(x, y)$ can be simply obtained by

$$\psi(x, y) = \angle \exp(j\phi(x, y)). \tag{2}$$

By examining the code of the simulation provided by the authors, we can look more closely into the details of the training data. The number of Gaussians is chosen randomly as an integer in $[2, 5]$. Each Gaussian amplitude is selected randomly as an integer in $[50, 1000]$ radians. The location of the Gaussians is chosen randomly in the image (but a 20 pixels margin on the borders is taken so that the center of the Gaussians is always more than 20 pixels away from the border). Each Gaussian's standard deviation (which controls the spatial extent) is taken randomly from $[10, 45]$. Then the Gaussians are summed and multiplied by 0.1. On the other hand, the ramp's slopes in the x and y direction are drawn from a uniform distribution $\mathcal{U}(0, 0.5)$ and the ramp's constant is chosen as a random integer in $[1, 10]$. The ramp is added to the Gaussian mixture.

It is beneficial at this point to stop and examine the interval values chosen by the authors. Each Gaussian in the mixture should yield a phase with a range of at most ≈ 100 radians (because it is later multiplied by 0.1). Similarly, for the maximum slope for the ramp of 0.5 rad/px, considering the image dimension, the ramp should yield at most 128 rad of phase variation in the image. Therefore, the intervals of values are chosen to balance the Gaussians and the ramp. Then the combination of the Gaussian and the ramp is rescaled to $[-2\pi a, 2\pi b]$, where a and b are chosen random integers in $[1, \text{max\_lower\_bound} + 1]$ and $[1, \text{max\_upper\_bound} + 1]$.

According to the article, the training dataset consists in 6000 images with 5000-1000 train/test split. As for max_lower_bound and max_upper_bound, they might have been fixed to 6, since the article says that the unwrapped phase range is $[-44, 44]$. However, from the naming of the model in the open source code, we suspect that the given weights correspond to a model trained on 1000 image pairs with max_lower_bound = max_upper_bound = 4.

Finally, the standard deviation was determined for the noise simulation from fixed Signal to Noise Ratio (SNR) levels, assuming that the signal power was equal to 1 dB. The SNR levels were chosen randomly from $[0, 5, 10, 20, 60]$ which is equivalent to saying the noise standard deviation was chosen randomly from $[1.12, 0.63, 0.35, 0.11, 0.0011]$ radians.

Finally, all the components are added, which gives us the unwrapped phase $\phi(x, y)$. The wrapped phase $\psi(x, y)$ is obtained following Equation (2).

2.3 Training

As mentioned in the introduction, the phase unwrapping problem does not have a unique solution, i.e. many $\phi(x, y)$ can correspond to the same wrapped phase $\psi(x, y)$. Therefore, the network should learn that multiple solutions are possible. Furthermore, it should not be penalized if its output is shifted by a constant w.r.t. the ground truth. Therefore, the loss reflects this last statement,

$$\mathcal{L}_c = \lambda_1 \mathcal{L}_{var} + \lambda_2 \mathcal{L}_{tv}, \tag{3}$$

where

$$\mathcal{L}_{\text{var}} = \mathbb{E}\left[(\hat{\phi} - \phi)^2\right] - (\mathbb{E}[(\hat{\phi} - \phi)])^2, \tag{4}$$

$$\mathcal{L}_{tv} = \mathbb{E}\left[\left|\hat{\phi}_x - \phi_x\right| + \left|\hat{\phi}_y - \phi_y\right|\right], \tag{5}$$

and $\lambda_1 = 1$, $\lambda_2 = 0.1$. $\hat{\phi}$ is the predicted phase and ϕ is the ground truth phase.

$\mathcal{L}_{\text{var}}$ minimizes the variance of the difference between the predicted and the ground truth phase (i.e., the difference should be ideally constant). $\mathcal{L}_{tv}$ is the total variation loss that minimizes the L^1-norm of the difference between the gradients of the predicted and ground truth phase.

We did not retrain the model and will only explain the training procedure of [9]. The model was implemented in Keras and trained using the ADAM optimizer with a 0.001 learning rate. It is worth mentioning that the input phase is the wrapped noisy phase, and the ground truth output is the unwrapped noiseless phase. So the network also learns to denoise the phase during unwrapping. The weights of the convolutional layers were initialized with the he_normal method [6], i.e. the weights are drawn as samples from a truncated normal distribution centered on 0 with $\sigma = \sqrt{\frac{2}{\text{fan\_in}}}$ where fan_in is the number of input units in the weight tensor.

2.4 Evaluation

The authors of [9] evaluated their method against others, such as Phasenet2.0 [12], on 1000 test samples generated with the same simulation procedure as the one used to get the training data. The other networks were retrained on the same training data for a fair comparison. The metric that was used is the Normalized Root Mean Square Error (NRMSE - normalized by the max-min range of the corresponding true unwrapped phase image). Each prediction was rescaled by an affinity to match the range of the ground truth before NRMSE computation. Using this evaluation procedure, the authors showed that their method had the lowest NRMSE (0.9%) with the lowest average run time per output (0.054 seconds).

Since the ground truth is not available in an actual phase unwrapping scenario, the rescaling procedure adopted is not really adapted prior to evaluation. In theory, we should only be allowed to add a constant to the result, but scaling is inappropriate. Furthermore, NRMSE might hide some method failures if the data range is significant. However, the authors' statement should still hold since visual comparison in their article shows that the method gives good results.

For our demo, we used the standard deviation of the error as an evaluation metric as seen in Equation (6) and (7) (similarly to Equation (4)), and we also look at its normalization w.r.t. ground truth data range, for completeness, as seen in Equation (8)

$$err = \hat{\phi} - \phi, \tag{6}$$

$$\sigma_{err} = \sqrt{\mathcal{L}_{var}}, \tag{7}$$

$$\text{NRMSE} = \frac{\sigma_{err}}{max(\text{gt}) - min(\text{gt})}, \tag{8}$$

where gt is the ground truth image.

3 Experiments

In order to evaluate the performance of the method, we developed our own data simulation. The idea is to generate samples from the training data distribution by wisely choosing specific parameters, and to be able to get out-of-distribution data as well. This will enable us to examine the effect of the chosen parameters on the performance, by visual inspection and by looking at the evaluation metrics.

3.1 Simulation

The simulation for the demo is inspired by the previous simulation detailed in Section 2.2. For brevity, only the differences will be explained here. For the Gaussian mixture, the amplitude is taken this time from $\mathcal{U}(0.2, 1)$. There is also an option to invert (multiply by -1) the amplitude of half the Gaussians. This option is added because the original simulation only had extruding Gaussians. The Gaussians are rescaled to a given interval $[-\frac{\text{gauss\_scale}}{2}, \frac{\text{gauss\_scale}}{2}]$, where gauss_scale is given by the user. The ramp's slopes and intercept are drawn from $\mathcal{U}(0, 1)$. The ramp is rescaled similarly to the Gaussian with a user input ramp_scale. For the noise, the standard deviation can be specified by the user.

An interesting feature that was added to the simulation is the ability to generate atmospheric phase images. The images occur in the context of InSAR due to the signal propagation delay in the atmosphere. It was shown that a simple way to generate the turbulent component of the atmospheric phase is to use fractal surfaces, typically with a dimension of 2.67, and the code was adapted from the DORIS software Matlab simulation files (insarfractal) [7]. This feature allows us to test the network with out-of-distribution data. The atmospheric signal will be rescaled with a user input atmo_scale.

3.2 Results

In this section, we test the network with simulated data using different parameters. The parameters of the first configuration are shown in the first line of Table 1, for experiment *good*. They should yield an image that looks like the training data. The results of the simulation, as well as the network unwrapping, can be seen in the first line of Figure 2. We can see that the network successfully retrieved a continuous unwrapped image that has a similar aspect to the ground truth. The error shows some patterns and is not constant. The error standard deviation is 0.276 rad, which can be acceptable or not depending on the application. Normalizing the error by the ground truth image range, we get NRMSE = 1.04%, where gt is the ground truth image. This result is comparable with the NRMSE values shown by the authors.

exper_id	n_g	inv_g	g_scl	ramp_scl	σ_{noise}	atmo_scl	σ_{err}	NRMSE
good	4	no	15	15	0.5	0	0.276	1.04 %
trivial	1	no	2	1	0	0	0.376	14.33 %
invert_gauss	4	yes	15	15	0.5	0	0.341	1.81 %
atmo	4	no	15	15	0.5	8	0.643	2.58 %
fast_varying	4	no	50	15	0.5	0	1.157	1.99 %

Table 1: Experiment parameters and results. From left to right: the column abbreviation refers to experiment id, number of Gaussians, invert Gaussians, Gaussian scale, ramp scale, noise standard deviation, atmospheric scale, error standard deviation, normalized root mean squared error, i.e. the normalization of the error standard deviation w.r.t. ground truth range.

For the *trivial* experiment, the parameters in Table 1 have been chosen to get a small range for the ground truth phase. This way, there would be no fringes (discontinuities) in the wrapped phase, as can be seen in the second line of Figure 2. Furthermore, no noise was added, so any traditional phase unwrapping technique would be able to perform the task with very high accuracy. In this case, the output should equal the input shifted by an arbitrary constant. However, the network modifies the input. The error standard deviation of 0.376 might be considered high because of the trivial nature of the problem. The NRMSE reaches the highest value of all our experiments because of the small range of the ground truth. In terms of the visual aspect, we can see clearly in this example,

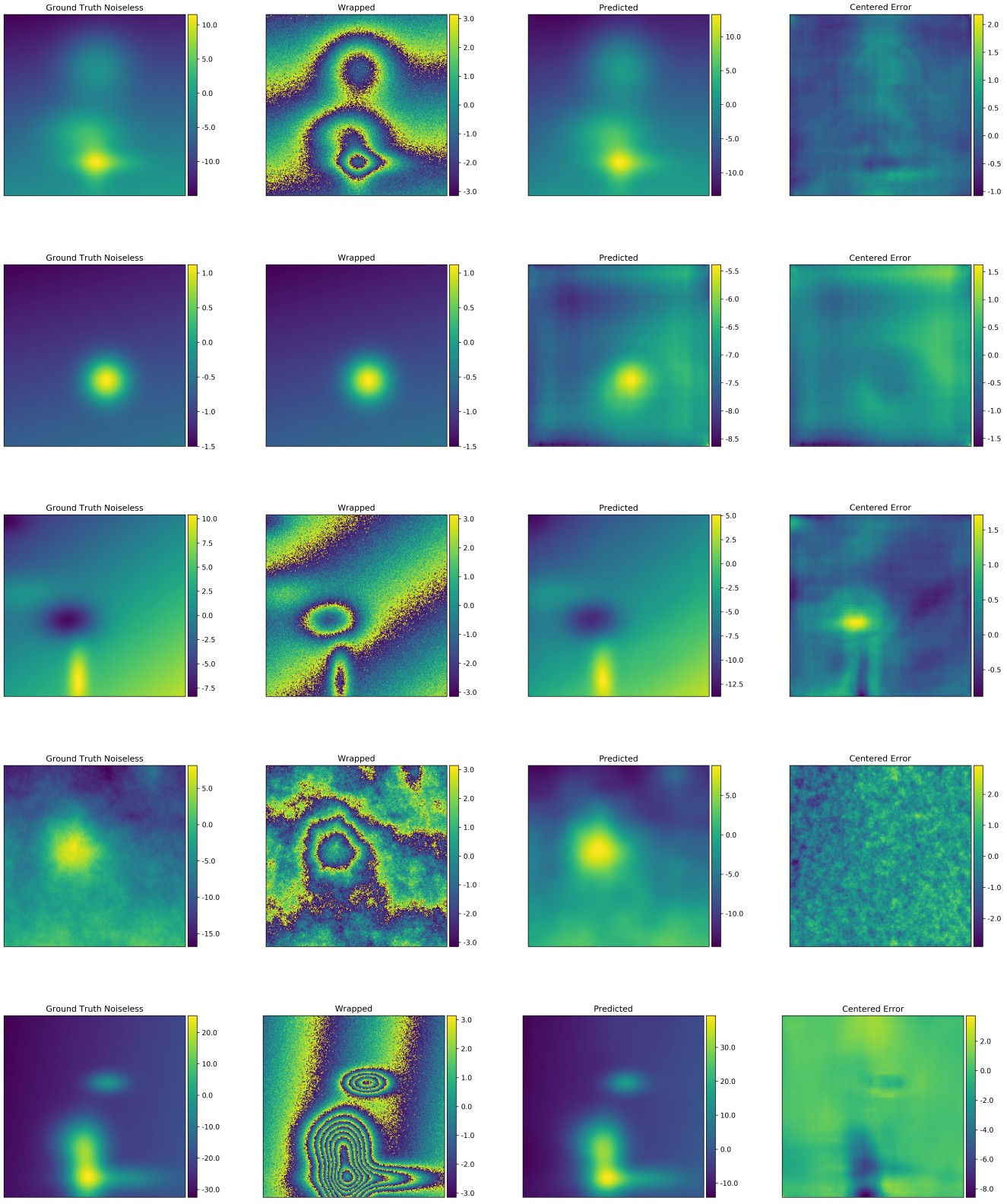


Figure 2: Results on the different sets of experiments in the same order as shown in Table 1 from top to bottom. From left to right: the ground truth noiseless image, the wrapped noisy image, the predicted unwrapped image, and the centered error.

especially in the flat areas, how the network has a tendency to generate stripes and gridded structures

in the output.

For the *invert_gauss* experiment, all the parameters of the *good* experiment are kept, but half the Gaussians will be inverted. It is clear that the error metrics are higher as seen in Table 1. Furthermore, by looking at the error plots, we can see that the error is structured and is higher around the Gaussian peaks. The error is positive for the negative peaks and negative otherwise. Therefore, the prediction underestimates the peaks in both the rising and decreasing cases.

For the *atmo* example, an atmospheric signal was added with a scale of 8 rad. Here, the ground truth is obtained by adding the Gaussians, the ramp, and the atmospheric phase. Since it is preferable to restrict the responsibility of the network, the network should only unwrap and denoise. When looking at the error plots in Figure 2, we can see that the network over-smoothed the result and eliminated some components of the atmospheric phase. The error is dominated by the atmospheric phase since it has been lost in the prediction. This phenomenon explains why the error shows such high values in Table 1.

For the *fast_varying* experiment, the Gaussian scale is increased in a way that would yield a range of values in the ground truth phase near the limits of the ranges seen during training. The resulting wrapped image has many fringes that are close to each other. The network's prediction has a similar aspect to the input image, but the prediction underestimates the peak of the Gaussians. The standard deviation of the error is the biggest in this case, but the NRMSE of 1.99% is reasonable.

We can observe from the previous experiments that the network performed reasonably well on in-domain data. The performance deteriorates on out-of-domain data. This behavior is natural and can be circumvented by simply re-training the network on this type of data. However, the question of network responsibility remains relevant. Even though the association between phase denoising and phase unwrapping might seem reasonable, this complicates the evaluation procedure. Indeed, in this case, part of the error might be due to the denoising task (over/under smoothing). For future studies, predicting the noisy unwrapped image might be interesting, i.e., the network would only try to unwrap the image.

4 Demo

In this section, the demo developed to showcase the method's performance is illustrated.

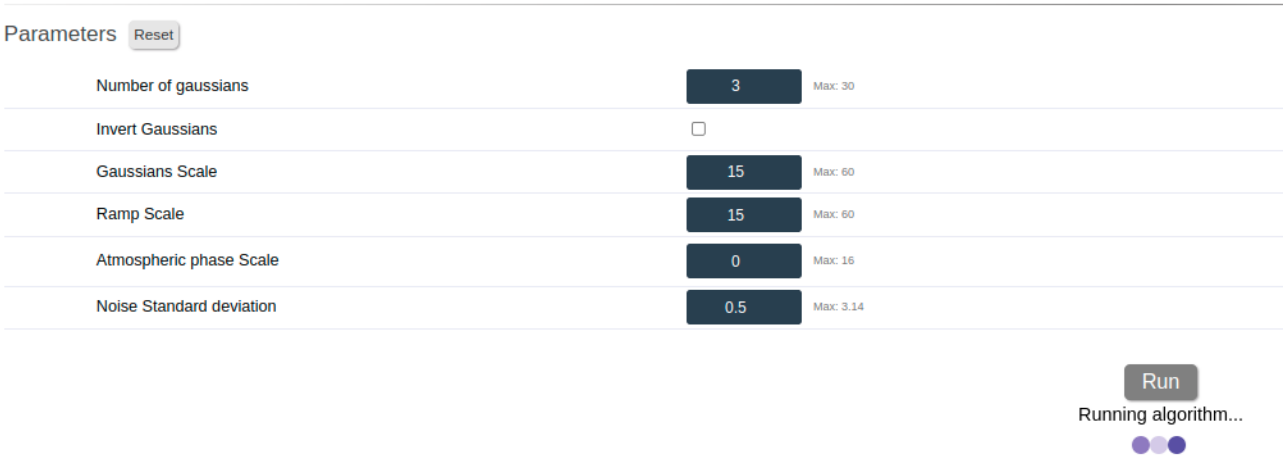


Figure 3: Interface of the demo. Each input in a blue box is of numeric type. The maximum value is shown on the right. The number of Gaussians should be an integer, while others can be floating-point scalars. Invert Gaussians is a Boolean represented by a checkbox on the interface. The default values that are shown yield images similar to the training data.

Figure 3 displays the demo's user interface. The parameters that are shown are needed to simulate

the ground truth unwrapped image ϕ and deduce the noisy wrapped image ψ. The noisy wrapped image is then fed to the network, and the prediction $\hat{\phi}$ is used to estimate the error err according to (6).

The simulation is performed with a set of random variables that control the Gaussian positions, standard deviation, relative scale, and the ramp slopes, atmospheric phase pattern, and noise pattern. Therefore, re-running the demo with the same parameters will give different realizations of the random variables, different inputs to the network, and different results.

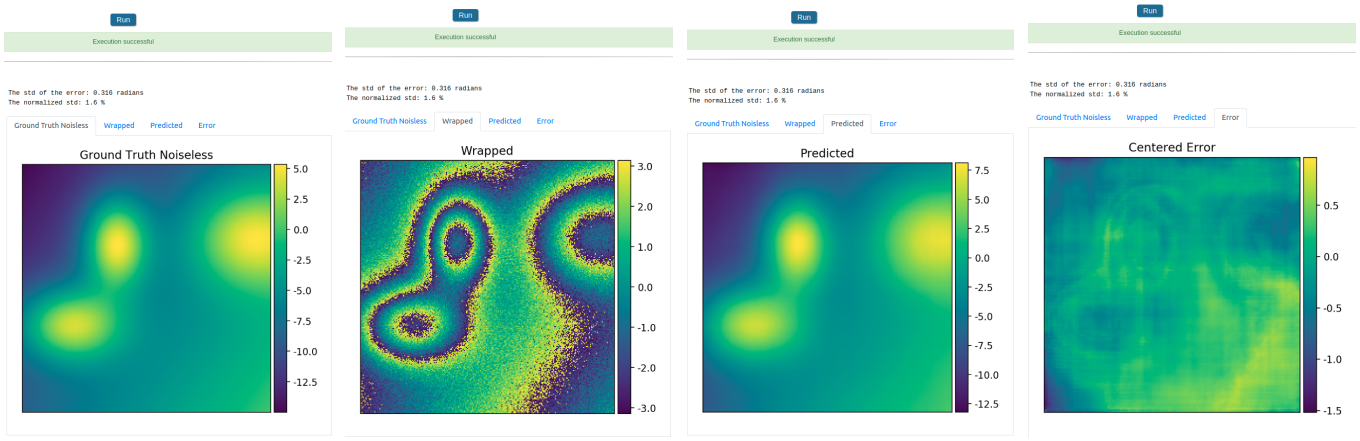


Figure 4: The output of the demo. The metrics are displayed on top of a tabset. Four tabs are available for the ground truth, wrapped noisy, predicted and error image.

When the user presses the run button, three moving dots are displayed, like at the bottom right of Figure 3. When the execution is finished, the user gets a result in the form shown in Figure 4. The error standard deviation and its normalization are printed out. A tab set is displayed with the visualized results. Here, each tab was clicked in succession, and the successively displayed images are shown in a single row in Figure 4. From left to right, we see the ground truth unwrapped image, the wrapped noisy image, the predicted unwrapping by the network, and the prediction error.

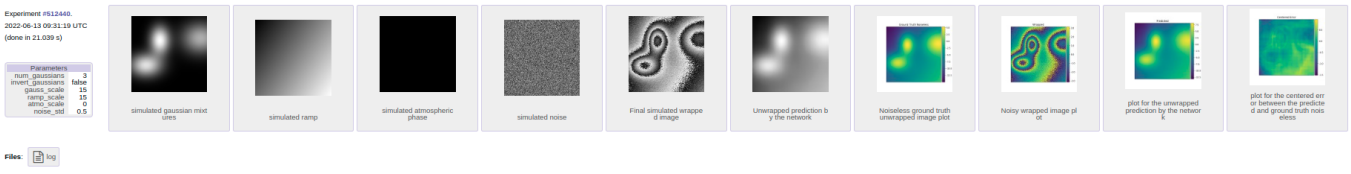


Figure 5: Archive

```
num_gaussians,invert_gaussians,gauss_scale,ramp_scale,noise_std,atmo_scale,error_std,error_std_norm
3,False,15,15,0.5,0,0.31597552,0.015534976
```

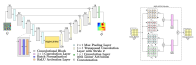
Figure 6: Log text file containing input parameters and output metrics in csv format.

The results are stored in the archive of the paper, and can be revisited later and downloaded. As can be seen in Figure 5, the archived results do not only correspond to the output plots, but also to the individual simulations in the tif format, the parameters used for the simulation, and a log text file. The log file contains the parameters of the simulation as well as the output metrics in csv format, as shown in Figure 6.

Acknowledgment

- Original article in [9], and source code at github[3].
- Doris software for the simulation of the atmospheric phase [7].

Image Credits

 Network architecture from [9]

References

[1] C.W. Chen and H.A. Zebker, *Two-dimensional phase unwrapping with use of statistical models for cost functions in nonlinear optimization*, Journal of the Optical Society of America (JOSA) A, 18 (2001), pp. 338–351. https://doi.org/10.1364/JOSAA.18.000338.

[2] M. Costantini, *A novel phase unwrapping method based on network programming*, IEEE Transactions on Geoscience and Remote Sensing, 36 (1998), pp. 813–821. https://doi.org/10.1109/36.673674.

[3] D.C. Ghiglia and M.D. Pritt, *Two-dimensional phase unwrapping: theory, algorithms, and software*, A Wiley Interscience Publication, (1998). ISBN 978-0-471-24935-1.

[4] R.M. Goldstein, H.A. Zebker, and C.L. Werner, *Satellite radar interferometry: Two–dimensional phase unwrapping*, Radio Science, 23 (1988), pp. 713–720. https://doi.org/10.1029/RS023i004p00713.

[5] R. Grandin, M-P. Doin, L. Bollinger, B. Pinel-Puysségur, G. Ducret, R. Jolivet, and S.N. Sapkota, *Long-term growth of the Himalaya inferred from interseismic InSAR measurement*, Geology, 40 (2012), pp. 1059–1062. https://doi.org/10.1130/G33154.1.

[6] K. He, X. Zhang, S. Ren, and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification*, in IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034. http://dx.doi.org/10.1109/ICCV.2015.123.

[7] B.M. Kampes, R.F. Hanssen, and Z. Perski, *Radar interferometry with public domain tools*, in FRINGE 2003 Workshop, vol. 550, 2004.

[8] S.M. Pandit, N. Jordache, and G.A. Joshi, *Data-dependent systems methodology for noise-insensitive phase unwrapping in laser interferometric surface characterization*, Journal of the Optical Society of America (JOSA) A, 11 (1994), pp. 2584–2592. https://doi.org/10.1364/JOSAA.11.002584.

[9] M.V. Perera and A. De Silva, *A Joint Convolutional and Spatial Quad-Directional LSTM Network for Phase Unwrapping*, in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 4055–4059. http://dx.doi.org/10.1109/ICASSP39728.2021.9414748.

[3] https://github.com/Laknath1996/DeepPhaseUnwrap

[10] M.D. Pritt and J.S. Shipman, *Least-squares two-dimensional phase unwrapping using FFT's*, IEEE Transactions on Geoscience and Remote Sensing, 32 (1994), pp. 706–708. https://doi.org/10.1109/36.297989.

[11] A. Rauscher, M. Barth, J.R. Reichenbach, R. Stollberger, and E. Moser, *Automated unwrapping of MR phase images applied to BOLD MR-venography at 3 Tesla*, Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine, 18 (2003), pp. 175–180. https://doi.org/10.1002/jmri.10346.

[12] G.E. Spoorthi, R.K.S.S. Gorthi, and S. Gorthi, *PhaseNet 2.0: Phase unwrapping of noisy data based on deep learning approach*, IEEE Transactions on Image Processing, 29 (2020), pp. 4862–4872. https://doi.org/10.1109/TIP.2020.2977213.

[13] L. Zhou, H. Yu, and Y. Lan, *Deep convolutional neural network-based robust phase gradient estimation for two-dimensional phase unwrapping using SAR interferograms*, IEEE Transactions on Geoscience and Remote Sensing, 58 (2020), pp. 4653–4665. https://doi.org/10.1109/TGRS.2020.2965918.

[14] L. Zhou, H. Yu, Y. Lan, and M. Xing, *Artificial Intelligence in Interferometric Synthetic Aperture Radar Phase Unwrapping: A Review*, IEEE Geoscience and Remote Sensing Magazine, 9 (2021), pp. 10–28. https://doi.org/10.1109/MGRS.2021.3065811.

[15] ——, *Deep Learning-Based Branch-Cut Method for InSAR Two-Dimensional Phase Unwrapping*, IEEE Transactions on Geoscience and Remote Sensing, 60 (2021), pp. 1–15. https://doi.org/10.1109/TGRS.2021.3099997.