Deep Learning and Quantum Annealing Methods in Synthetic Aperture Radar

by

Khaled Kelany
B.Eng., Cairo University, 2013
M.Sc., Cairo University, 2016

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

Deep Learning and Quantum Annealing Methods in Synthetic Aperture Radar

by

Khaled Kelany
B.Eng., Cairo University, 2013
M.Sc., Cairo University, 2016

Supervisory Committee

Dr. Nikitas Dimopoulos, Main Supervisor
(Department of Electrical and Computer Engineering)

Dr. Amirali Baniasadi, Co-Supervisor
(Department of Electrical and Computer Engineering)

Dr. Hausi A. Müller, Departmental Member
(Department of Electrical and Computer Engineering)

Dr. Clemens P. J. Adolphs, Outside Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Nikitas Dimopoulos, Main Supervisor
(Department of Electrical and Computer Engineering)

Dr. Amirali Baniasadi, Co-Supervisor
(Department of Electrical and Computer Engineering)

Dr. Hausi A. Müller, Departmental Member
(Department of Electrical and Computer Engineering)

Dr. Clemens P. J. Adolphs, Outside Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Mapping of earth resources, environmental monitoring, and many other systems require high-resolution wide-area imaging. Since images often have to be captured at night or in inclement weather conditions, a capability is provided by Synthetic Aperture Radar (SAR). SAR systems exploit radar signal's long-range propagation and utilize digital electronics to process complex information, all of which enables high-resolution imagery. This gives SAR systems advantages over optical imaging systems, since, unlike optical imaging, SAR is effective at any time of day and in any weather conditions. Moreover, advanced technology called Interferometric Synthetic Aperture Radar (InSAR), has the potential to apply phase information from SAR images and to measure ground surface deformation. However, given the current state of technology, the quality of InSAR data can be distorted by several factors, such as image co-registration, interferogram generation, phase unwrapping, and geocoding.

Image co-registration aligns two or more images so that the same pixel in each image corresponds to the same point of the target scene. Super-Resolution (SR),

on the other hand, is the process of generating high-resolution (HR) images from a low-resolution (LR) one. SR influences the co-registration quality and therefore could potentially be used to enhance later stages of SAR image processing. Our research resulted in two major contributions towards the enhancement of SAR processing. The first one is a new learning-based SR model that can be applied with SAR, and similar applications. A second major contribution is utilizing the devised model for improving SAR co-registration and InSAR interferogram generation, together with methods for evaluating the quality of the resulting images.

In the case of phase unwrapping, the process of recovering unambiguous phase values from a two-dimensional array of phase values known only modulo 2π rad, our research produced a third major contribution. This third major contribution is the finding that quantum annealers can resolve problems associated with phase unwrapping. Even though other potential solutions to this problem do currently exist - based on network programming for example - network programming techniques do not scale well to larger images. We were able to formulate the phase unwrapping problem as a quadratic unconstrained binary optimization (QUBO) problem, which can be solved using a quantum annealer. Since quantum annealers are limited in the number of qubits they can process, currently available quantum annealers do not have the capacity to process large SAR images. To resolve this limitation, we developed a novel method of recursively partitioning the image, then recursively unwrapping each partition, until the whole image becomes unwrapped. We tested our new approach with various software-based QUBO solvers and various images, both synthetic and real. We also experimented with a D-Wave Systems quantum annealer, the first and only commercial supplier of quantum annealers, and we developed an embedding method to map the problem to the D-Wave 2000Q_6, which improved the result images significantly. With our method, we were able to achieve high-quality solutions, comparable to state-of-the-art phase-unwrapping solvers.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

There are many people that I will always be indebted to. The least I can do is recognizing their efforts. First, I want to thank my family. Despite the physical distances between us during my fulfilment to my Ph.D., they were always there for me. I would like to thank my advisor, Dr. Nikitas Dimopoulos, for welcoming me to his lab and amongst his research group. Throughout the years of my Ph.D., he gave me all the guidance and the support I needed. I gained much experience from him on both the academic and personal levels. He truly treated me like a family. Also, I would like to thank my co-supervisor, Dr. Amirali Baniasadi, for giving me the opportunity to work under his guidance. Besides the academic knowledge, he was keen to provide me with life lessons. His wisdom shaped part of my personality. The list could go forever if I start mentioning my friends by name. To all my friends, thank you.

Khaled Kelany

DEDICATION

I dedicate this work to my family and many friends. A special feeling of gratitude to my loving parents, Abdullah and Awatef, whose words of encouragement and support ring in my ears. My brothers Hany and Tamer, I will always appreciate all they have done.

Chapter 1

Introduction

Mapping of earth resources, environmental monitoring, and many other systems require high-resolution wide-area imaging. Such images often have to be captured at night or in inclement weather conditions. Such a capability is provided by Synthetic Aperture Radar (SAR). SAR systems exploit the advantages of the characteristics of radar signal's long-range propagation and the advantages of the capability of modern digital electronics to process complex information to provide high-resolution imagery. SAR systems have advantages over optical imaging systems as they are not restricted by atmospheric conditions or time of day. Besides, SAR systems have unique responses to radar frequencies from terrain targets.

SAR is a microwave imaging system that uses a coherent side-looking radar. The coherent side-looking radar is the antenna being "flown" over the target area. The antenna is mounted on a platform; a satellite, airplane or ship. The received signal phase and amplitude carry extensive scatterer information with it (i.e., terrain). Signal processing algorithms are used to combine multiple received signals (SAR images) to extract relevant scattering terrain information. The information includes the elevation and the nature of the terrain (e.g., water vs. vegetation).

Two or more SAR images of the same terrain can be used to extract any changes (including elevation) of the terrain. This approach is called Interferometric Synthetic Aperture Radar (InSAR). Since the images are obtained from various antenna locations and at different times, they are usually processed as follows:

1. SAR image acquisition

2. Co-registration of two or more SAR images

3. Interferogram generation

4. Phase unwrapping

5. Geocoding of digital elevation models

Different critical factors affect the quality of products resulting from the InSAR images. These factors include co-registration and phase unwrapping. Co-registration is the alignment of two SAR images taken by the same antenna but at different repeat passes (usually several days later in a slightly different imaging position.) This alignment is essential to ensure the same pixel in the two SAR images reflects the same ground target. InSAR technology benefits from enhancing the accuracy of co-registration.

Two-dimensional phase unwrapping is the process of recovering unambiguous phase values from a two-dimensional array of phase values known only modulo 2π rad. The measured phase is also affected by random noise and systematic distortions. This problem arises when the phase is used as a proxy indicator of a physical quantity, which is the time delay between two signals in the case of interferometric synthetic aperture radar (InSAR) [17]. This time delay is significant, as it is affected by the height differences of the illuminated target. It can thus be used to extract accurate three-dimensional topography and reveal topographical changes that occur over time. As the phase is observable only on a circular space where all measured values are mapped to the range $(-\pi, \pi]$, the observed data must be mapped back to the full range of real phase values to be meaningful. We discuss co-registration and phase unwrapping in more detail in Chapter 2.

In this proposal, we study employing machine learning, using Convolutional Neural Networks (CNNs), and quantum computing to improve the InSAR processing.

1.1 Employing Machine Learning to Improve the InSAR Processing

InSAR is a measuring technology that uses the phase information in the SAR images. InSAR has been recognized as a potential method for digital elevation models (DEMs) generation and ground surface deformation measurement. Many critical factors influence the quality of InSAR data. Including image co-registration, interferogram generation, and phase unwrapping. Image co-registration aims to align two or more

images so that the same pixel in each image corresponds to the same point of the target scene. This study proposes a new algorithm for improving image co-registration and interferogram generation of SAR using learning-based images Super-resolution (SR).

SR is the process of generating high-resolution (HR) images from low-resolution (LR) ones. In learning-based SR algorithms, Artificial Neural Networks (ANN) are used. This is achieved by training the network using HR and LR image pairs and use this network later to create new HR images from LR ones.

However, in some applications (including SAR), HR ground truth images are missing. Additionally, training the model using HR images is computationally expensive. Our work postulates that the scaling process is invariant across scales. Thus, a model trained at lower scales can reconstruct higher resolution images when the ground truth is unavailable to train the model. We call this approach Scale-Invariant Super-Resolution (SINV). We evaluated SINV using different datasets and with different upscaling factors [1], and showed that it outperforms conventional approaches. We have applied SINV to processing InSAR images.

1.2 Employing Quantum Computing to Improve the InSAR Processing

This work explores the use of quantum annealing solvers for the problem of phase unwrapping of SAR images. Although solutions to this problem exist based on network programming, these techniques do not scale well to larger-sized images. Our approach involves formulating the problem as a quadratic unconstrained binary optimization (QUBO) problem, which can be solved on a quantum annealer. Given that present embodiments of quantum annealers remain limited in the number of qubits they possess, we decompose the problem into a set of subproblems that can be solved individually. These individual solutions are close to optimal up to an integer constant, with one constant per sub-image. In a second phase, these integer constants are determined as a solution to yet another QUBO problem. This basic idea is extended to several passes where each pass results in an image which is subsequently decomposed to yet another set of sub-problems until the annealer can accommodate the resulting image at hand.

[1] The upscaling factor is the factor by which the image resolution is increased.

We test our approach with various software-based QUBO solvers and various images, both synthetic and real. Additionally, we experimented using D-Wave Systems' quantum annealer, the D-Wave 2000Q_6 and developed an embedding method which, for our problem, yielded improved results. Our method resulted in high-quality solutions comparable to state-of-the-art phase unwrapping solvers.

The proposal is organized as follows. Chapter 2 presents background materials on SAR systems, Neural Networks, and quantum computing. In Chapter 3, we summarize our approach to tackle the problem of train super-resolution CNN for InSAR images. Chapter 4 summarizes the work accomplished in improving InSAR image quality through CNNs. Chapter 5 summarizes the methodology we developed in unwrapping the InSAR images phase through quantum annealers. Chapter 6 presents the conclusions and the future work. Appendix A provides detailed derivations of the equations presented in Chapter 2.

Chapter 2

Background

This chapter presents background materials on SAR, Neural Networks, and quantum computing.

2.1 Synthetic Aperture Radar

SAR is a microwave imaging system that uses a coherent side-looking radar [17]; the antenna being "flown" over the target area. Satellites, airplanes or ships are used as antenna platforms. SAR systems produce remote sensing imagery of high resolution. The phase and the amplitude of the received signal carry detailed information of the scatterer (i.e., terrain). Signal processing algorithms combine several received signals (SAR images) to extract pertinent information of the scattering terrain. Including elevation as well as information about the nature of the terrain (e.g., water vs. vegetation). A SAR imaging system is illustrated in figure 2.1.

SAR operates similarly to a phased array, but unlike the many parallel antenna components of a phased array system, SAR utilizes one time-multiplex antenna. The moving platform results in the distinct geometric positioning of the antenna elements. The direction along the track is called the azimuth direction (see figure 2.1). The direction in the across-track is called the ground range direction.

SAR systems are advantageous as compared to vision because [25]:

- The use of microwaves allows for cloud penetration.
- Being active systems, they can function daytime or night.

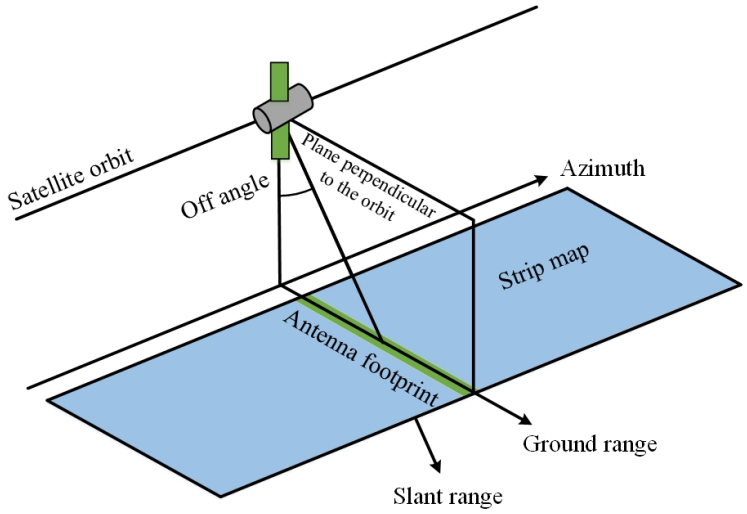


Figure 2.1: A SAR system from a satellite

- Being coherent, they can utilize interferometry (Interferometric SAR or InSAR) to measure the radiation travel path accurately.

The Interferometric Synthetic Aperture Radar (InSAR) is a Satellite imagery technology that uses the information contained in the phase of the SAR images. InSAR is utilized for mapping the deformation of the ground. InSAR uses two or more SAR images of the same target to extract any changes (including elevation).

Since the images are acquired at different times and from different antenna position, their processing follows a five-stage procedure as follows [116]:

1. SAR image acquisition;
2. Co-registration of two or more SAR images;
3. Interferogram generation;
4. Phase unwrapping;
5. Geocoding of digital elevation models.

The quality of InSAR images is influenced by various critical factors, including the co-registration of the SAR images and the phase unwrapping of the InSAR images. This section reviews some of the basic concepts of SAR image processing and super-resolution principles.

2.1.1 Complex SAR Image

A digital SAR image is a two-dimensional array formed by columns and rows of pixels. Each of these pixels is associated with a small area of the terrain's surface (called a resolution cell). Each pixel holds a complex number containing amplitude and phase information about the microwave field backscattered by all the scatterers on the ground within the corresponding resolution cell. Different columns of the image are associated with different slant range locations, while different rows indicate different azimuth.

2.1.2 The Amplitude SAR Image

This component of the SAR image comprises a measurement of the amplitude of the microwave backscattered by the scatterers in each SAR resolution cell. This amplitude is more dependent on the type of the scatterer. For example, trihedral scatters (a standard radar reflector that is widely used to calibrate radar systems [6]) produce the strongest return if aligned to look at the sensor. Surface roughness also affects the amplitude. Smooth flat surfaces have small amplitudes if the scatterer is positioned to deflect the signal away from the sensor.

2.1.3 The Phase SAR Image

To form the SAR image, the wave transferred from the radar must reach the scatterers on the ground and then return to the radar (two-way travel). The scatterers at different distances from the radar introduce different delays between transmission and reception of the wave. Because of the sinusoidal nature of the signal being transmitted, this delay is equal to a phase change φ between transmitted and received signals. Therefore, the phase change is proportional to the wave's two-way travel distance $2R$ divided by the transmitted wavelength λ, as below:

$$\varphi = \frac{2\pi}{\lambda} 2R = \frac{4\pi}{\lambda} R \tag{2.1}$$

However, only the residual part of the phase is recorded, i.e., $\phi = \varphi \ mod \ (2\pi)$. In other words, the SAR signal phase measures only the last part of the two-way travel distance, which is lower than the wavelength transferred. In practice, due to the large ratio of the resolution cell dimensions (of the order of a few meters) to λ (of the order of a few centimetres), the phase shift within a single SAR image looks

random. Phase unwrapping [50] is used to recover the integer part of the phase and estimate the phase and hence the distance R.

2.1.4 SAR Interferometry

InSAR is used in many applications, including classifying ground cover, measuring glacial motion, and infrastructure monitoring. However, one of the most prevalent applications for InSAR is mapping the deformation of the ground. A target area can be imaged several times by the same satellite. If the target has remained unchanged, then the two images should be identical. Otherwise, a change in the interferometric phase is recorded. The signal of interest is subsidence that evolves slowly or not at all as it is measurable by InSAR. However, images obtained at different times have been imaged from different satellite perturbed orbital positions (due to natural drift in orbits.) Co-registration is used to find correspondences of pixels between two SAR images. The two SAR images are conventionally referred to as master and slave. The InSAR interferogram is generated by cross-multiplying the master SAR image with the complex conjugate of the co-registered slave image (slave image resampled to the master's geometry, pixel by pixel) [7, 27, 76].

Thus, the amplitude of the interferogram is the amplitude of the first image multiplied by the amplitude of the second image, while its phase is the phase difference between the two images. Many factors contribute to generating a synthetic interferogram, including the precise sensor orbits, scene topography, and timing information [25, 90].

2.1.5 Coherence

The amplitude of the complex correlation coefficient between two SAR images (called coherence) has been shown to be an essential source of physical information in dual- or multidimensional complex SAR imagery.

Coherence is a measure of similarity between two sets of complex numbers (SAR image in our case). The coherence is defined as the cross-correlation coefficient of the two SAR images evaluated over a small window (a few pixels in azimuth and range). The coherence between two complex images x and y is defined as [104]:

$$Coherence\,(x,y) = \frac{E\left\{x \cdot y^*\right\}}{\sqrt{E\left\{|x|^2\right\} \cdot E\left\{|y|^2\right\}}}$$

where x and y are complex co-registered InSAR images, $*$ is the complex conjugate operator, and $E\{\cdot\}$ is the mathematical expectation. The coherence value ranges from 0 to 1.

The coherence image is used in the majority of InSAR studies to assess the efficacy of SAR image coregistration [74]. The coherence value of 1 indicates the high similarity between the two images and hence a successful co-registration. While the value 0 indicates uncorrelated SAR images and poor co-registration [68].

Because of the complex nature intrinsic to SAR systems, SAR images are affected by speckle. Speckle is a signal-dependent noise that degrades the appearance of images and is present in all active coherent imaging systems [5]. Speckle has the potential to degrade the performance of scene analysis and information extraction, as well as affect applications that require multiple SAR images (such as InSAR). As a result, speckles are seen as a stochastic term that corrupts useful information and is considered a noise component. When working with multidimensional SAR imaging, coherence is used to assess the similarity of speckle patterns. Hence, a strong coherence between two SAR images translates to a high-quality phase difference between them. On the other hand, a low coherence means a highly noisy phase difference. For InSAR, coherence is used to evaluate the quality of the topographic map obtained from the phase component [74].

2.1.6 Co-registration

SAR interferometry generation requires a pixel-to-pixel match between the features of two SAR images. The co-registration is the alignment of SAR images captured from the same antenna at different acquisitions in time. This alignment is necessary to ensure the same pixel in the master and the slave image reflects the same ground target. Therefore, co-registration is a fundamental step for the precise determination of interferometric phase difference and reducing noise [68]. The alignment between the master and the slave is performed on a pixel-by-pixel basis. To achive this alignment, the slave image is tranformed (shifted and rotated) to match the master image. The most common approach to estimate the tranformation is to calculate the amplitude cross-correlation between the master and the slave images and fit the tranformation function co-efficients [67].

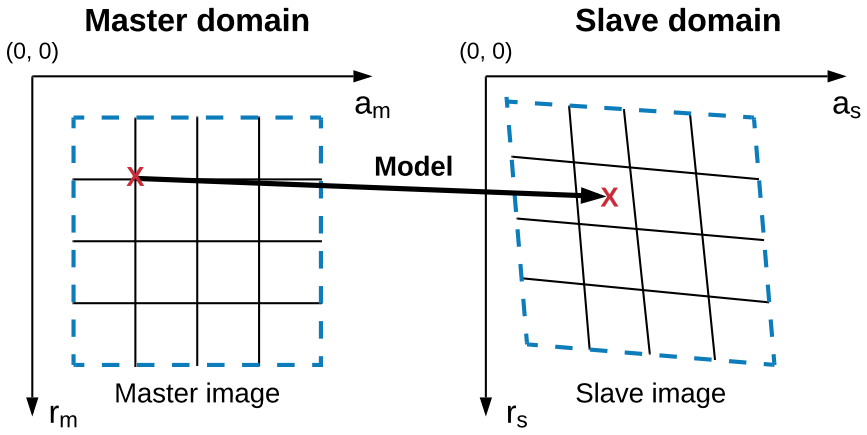


Figure 2.2: The transformation between the mater and the slave image

2.1.6.1 Transformation Coefficients

The master and the slave images capture the same terrain area but from two different antenna positions. Therefore, there is a deformation due to dilation and different orientation of the coordinate systems of the two images. The transformation that describes the deformation between the master and the slave can be well approximated by the following polynomial [69]:

$$\begin{cases} r_s = a \cdot r_M^2 + b \cdot r_M + c \cdot a_M + d \\ a_s = e \cdot r_M^2 + f \cdot r_M + g \cdot a_M + h \end{cases} \tag{2.2}$$

where (a_M, r_M) are the azimuth and range coordinates of the master image and (a_S, r_S) are the azimuth and range coordinates of the slave image corresponding. Figure 2.2 illustrates the transformation. After estimating the transformation function between the master and the slave, interpolation methods are used to approximate the subpixel values. Usually, Sinc interpolation gives the best performance [35, 68].

2.1.7 Phase Unwrapping

Two-dimensional phase unwrapping is the process of recovering unambiguous phase values from a two-dimensional array of phase values known only modulo 2π rad. This problem arises when the phase is used as a proxy indicator of a physical quantity, which is the time delay between two signals in the case of SAR. This time delay is significant, as it is affected by the height differences of the illuminated target.

It can thus be used to extract accurate three-dimensional topography and reveal topographical changes that occur over time.

As the phase is observable only on a circular space where all measured values are mapped to the range $(-\pi, \pi]$, the observed data must be mapped back to the full range of real phase values to be meaningful.

2.1.7.1 The Cost Function

Strictly speaking, phase unwrapping is an ill-posed problem as the unwrapped phase array contains information that is not available in the wrapped array. Therefore, to perform correctly, all phase unwrapping methods rely on regularizing assumptions. The most common of these assumptions is that the Nyquist criterion is met throughout most (but not necessarily all) of the scene; that is, the spatial sampling rate is assumed to be high enough that aliasing is avoided [13].

The Nyquist criterion implies that the difference between the phases of two neighbouring pixels is less than π. Therefore, the key to phase unwrapping lies not on directly calculating the unwrapped phase values themselves but in estimating these values given that the differences of the wrapped phases are the same as those of the unwrapped phases dictated by the Nyquist assumption.

Let ϕ, φ, and k denote the unwrapped phase, the wrapped phase, and an integer label to be estimated, respectively. For the phase of pixel i, we have,

$$\phi_i = \varphi_i + 2\pi k_i \tag{2.3}$$

The unwrapping problem can then be expressed as an optimization problem of the cost function,

$$E\left(k\right) = \sum_{(s,t)\in A} W_{st}|k_t - k_s - a_{st}| \tag{2.4}$$

that is,

$$\arg\max_k E(k) \tag{2.5}$$

where k_i are the labels that will determine the original phase as per equation 2.3, A is the set of pixels in the SAR image, W_{st} are weights defining the neighbourhood

structure, and a_{ij} are constants obtained from the image as per the equation:

$$a_{ij} \stackrel{\text{def}}{=} \frac{\text{wrap}\left(\phi_i - \phi_j\right) - \left(\phi_i - \phi_j\right)}{2\pi} \tag{2.6}$$

where

$$\text{wrap}\left(\theta\right) = \arg\left(e^{i\theta}\right) = \theta - \lfloor \frac{\theta}{2\pi} \rfloor \tag{2.7}$$

Appendix A details the derivation of the cost function shown in (2.4).

The optimization problem as defined in (2.6) above admits infinite solutions since only the difference of the labels is used in the cost function. Labels can be increased or decreased by the same amount and still result in the same minimal cost. A way to further regularise the solution is to insist that the desirable solution involves minimal possible labels. Therefore, the cost function is augmented with an extra term that depends on the labels themselves as follows,

$$E = \sum_{(s,t)\in A} W_{st}|k_t - k_s - a_{st}| + \sum_{s\in A} \omega_s|k_s - a_s| \tag{2.8}$$

The weights W_{st}, ω_s and the bias a_s are chosen heuristically and represent ad-hoc information one may have on the scene represented in the image. Most often, $a_s = 0$.

Without loss of generality, one can also consider cost functions involving quadratic expressions of the labels instead of the more challenging absolute value ones,

$$E = \sum_{(s,t)\in A} W_{st}\left(k_t - k_s - a_{st}\right)^2 + \sum_{s\in A} \omega_s\left(k_s - a_s\right)^2 \tag{2.9}$$

and in case that $a_s = 0$, then a similar cost function is:

$$E = \sum_{(s,t)\in A} W_{st}\left(k_t - k_s - a_{st}\right)^2 + \sum_{s\in A} \omega_s k_s^2 \tag{2.10}$$

Although the $L1$-norm is preferred over the $L2$-norm in the continuous case—as the $L2$-norm tends to spread the error and does not result in good solutions [15]—this is not a factor in the integer case.

The most commonly used method of solving the phase unwrapping problem, Tree-Reweighted Message Passing (TRWS), is attributed to V. Kolmogorov [57]. The TRWS algorithm is used for discrete energy minimization where the energy function can be formulated as follows:

$$E\left(x|\theta\right)=\theta_{\text{const}}+\sum_{s\in\nu}\theta_{s}\left(x_{s}\right)+\sum_{(s,t)\in\varepsilon}\theta_{st}\left(x_{s},x_{t}\right) \tag{2.11}$$

where ν corresponds to the set of pixels; x_s indicates the label of pixel $s \in \nu$, ε corresponds to the set of edges (each edge connects two related pixels), $\theta_s(\cdot)$ is the penalty function (i.e., a term of an unconstrained objective function added to add some constraint to it) of unary data, and $\theta_{st}(\cdot,\cdot)$ is the penalty function of the pairwise terms. This energy function is usually derived in the context of Markov random fields [29].

2.2 Machine Learning and Convolutional Neural Networks

The quality of products resulting from InSAR images is influenced by various critical factors including co-registration, interferogram generation, and phase unwrapping. Co-registration is the alignment of two SAR images taken by the same antenna, but a different repeat passes (usually several days later in a slightly different imaging position.) This alignment is essential to ensure the same pixel in the two SAR images reflects the same ground target. InSAR technology benefits from enhancing the accuracy of co-registration [116]. Machine learning models, specifically Convolutional Neural Networks (CNNs), have achieved great success in learning-based super-resolution (SR) algorithms.

In the first part of this thesis, we employ CNNs in interferometric SAR processing to improve the accuracy of the obtained results. In our approach, we use CNNs to generate super-resolution images which we then use to achieve superior co-registration and interferogram generation. Many papers in the literature proposed to improve the co-registration [33,72,80,99,108,113]. However, none of these papers employed super-resolution nor machine learning to address this problem. Hence, our thesis, to the best of our knowledge, is the first to pursue this approach.

This section provides a brief introduction to the progress made in the Machine Learning area, with an emphasis on CNNs.

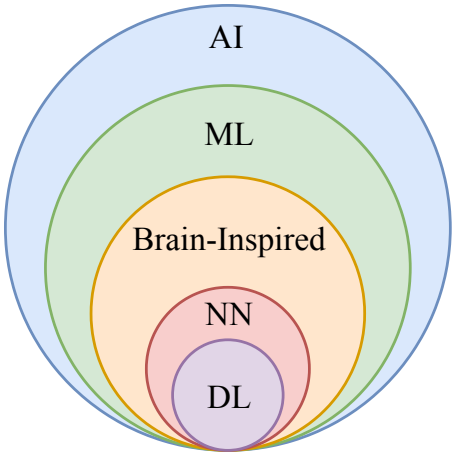


Figure 2.3: The taxonomy of AI. AI: Artificial Intelligence; ML: Machine Learning; NN: Neural Networks; DL: Deep Learning [3]

2.2.1 Machine Learning (ML)

ML algorithms are a branch of the specialized area of Artificial Intelligence (AI) research. Figure 2.3 shows the taxonomy of AI. ML provides intelligence to computer systems by making decisions without being explicitly programmed, which is achieved by learning the underlying relationships amongst the data [56]. ML is based on the premise that, with minimal human interference, systems can learn from information, recognize patterns and create decisions. ML is, therefore, a data analysis technique that automates the construction of analytical models.

In constructing mathematical models, ML utilizes the theory of statistics because the objective is to make inferences from a sample. Using example data or previous experience, the ML model is developed by programming computers to optimize a performance criterion. The ML model is defined through some parameters. During the learning process, an algorithm is run to optimize the model parameters using the training data [4].

ML is applied in a wide range of applications. These applications include computational learning, natural language processing, object detection, pattern recognition, visual object recognition, and speech recognition [23, 40, 59, 78, 88, 97, 100].

In the next subsection, we are going to present briefly different ML approaches.

2.2.1.1 Classification

Predictive modelling is the problem of developing a model that uses historical data to predict new data. Classification is a predictive model. Sometimes classes are referred to as targets/labels or categories. It is the process by which the class of data points is predicted. Hence, the task of Classification predictive modelling is to approximate a mapping function that maps from input variables to discrete output variables. Each output variable represents a class [4].

2.2.1.2 Regression

Regression is also a predictive model. The task of Regression predictive modelling is to approximate a mapping function that maps from input variables to a continuous output variable.

2.2.1.3 Supervised Learning

Regression and classification are both supervised learning problems where input, X, output, Y are present, and the objective is to learn a mapping from input to output. ML models are parameterized so that for a given problem, their behaviour can be changed. Such models can have many parameters and can be viewed as an optimization problem to find the best combination of parameters.

The machine learning algorithm optimizes the parameters to minimize the approximation error, that is, the error between the module output and the desired output Y [4].

2.2.1.4 Unsupervised Learning

In supervised learning, the goal is to learn a mapping from input to output for which a supervisor provides the right values. There is no such supervisor in unsupervised learning, and we only have input data. The purpose is to discover the regularities in the input [4]. Input space has a structure such that some patterns happen more frequently than others, and we need to see what generally occurs and what generally does not. This is called the density estimation in statistics. One technique for estimating density is clustering, where the objective is to discover input clusters or groupings [41, 89].

2.2.1.5 Reinforcement Learning

For some applications, the output is a sequence of actions. A single action is not important. In such a case, what is important is the policy which leads to the sequence of corrective actions to achieve the goal [4]. In any intermediate state, there is no such thing as the best action; an action is correct if it is part of a good policy. In this case, in order to generate a policy, the machine learning program should be able to evaluate the goodness of policies and learn from past sequences of good action. Such techniques of learning are called reinforcement learning.

2.2.2 Multilayer Perceptrons (MLP)

Many ML applications use feedforward Neural Network architectures, also known as MLP, which learn how to map a fixed-size input to a fixed-size output. To move from one layer to the next layer, a set of neurons, perceptron with activation function, calculate a weighted sum of their inputs coming from the previous layer and transfer the resulted sum through a nonlinear function. Neurons that are not in the input layer or the output layer are conventionally referred to as hidden layers. The hidden layers can be viewed as nonlinear, distorting the input so that the last layer can linearly separate categories [21].

The perceptron inputs may come from the outputs of the previous layer or from the environment. With each input, $x_i \in , i = 1, \ldots, d$, where d is the number of the perceptron inputs, there is a connection weight associated, w_i. In the simplest case, the output of the perceptron, y , is a weighted sum of the inputs:

$$y = \sum_{i=1}^{d} w_i x_i + w_0 \tag{2.12}$$

where w_0 is the intercept value for a more general model. It is modelled as the weight of an additional bias unit x_0. The perceptron, as described in (2.12), defines a hyperplane that can be used to split the input space into two classes: the half-space where the output is positive and the half-space where the output is negative [22].

A perceptron with a single layer of weights can only approximate linear functions of the input and cannot solve problems where the discriminant, to be estimated, is nonlinear. Similarly, a perceptron cannot be used for nonlinear regression. This limitation does not apply to feedforward networks with intermediate or hidden layers between the input and the output layers which include a nonlinear activation

function. Also, the activation function should be differentiable as we use gradient descent to train these networks. Such feedforward networks of perceptrons that include a nonlinear activation function at each perceptron in the hidden layer are termed multilayer perceptrons (MLP). MLP are trained using backpropagation (as we will explain shortly). The output is then a linear combination of the nonlinear functions computed through the hidden units. If used for classification, such MLP can implement nonlinear discriminants and, if used for regression, can approximate nonlinear functions of the input. If the outputs of the hidden units were linear, the hidden layer would be of no use because the linear combination of linear combinations is a linear combination.

It can be said that the hidden units transform nonlinearly from the d-dimensional input space into the H-dimensional space spanned by the hidden units. One is not restricted to one hidden layer. After the first hidden layer, more hidden layers can be put, thus implementing more complex functions of the input (figure 2.4).

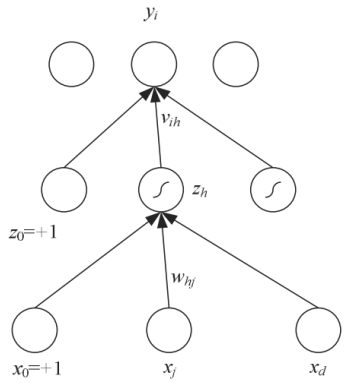


Figure 2.4: The structure of a multilayer perceptron. $x_j, j = 0, ..., d$ are the inputs and $z_h, h = 1, ..., H$ are the hidden units where H is the dimensionality of this hidden space. z_0 is the bias of the hidden layer. $y_i, i = 1, ..., K$ are the output units. w_{hj} are weights in the first layer, and v_{ih} are the weights in the second layer [21]

2.2.2.1 MLP as a Universal Approximator

A multilayer perceptron can approximate any arbitrary feature with continuous inputs and outputs. By increasing the number of hidden units, MLP accuracy can be increased to the desired value. It has been proven that an MLP with a single hidden layer (with an arbitrary number of hidden units) can learn any nonlinear function [44]. However, even if it is very wide with many neurons, using a very shallow network is good at memorization but not good at generalization. On the other hand, using

multiple layers provide the advantage of learning features at many levels of abstraction. Hence, multiple layers networks are much better at generalization as they learn intermediate features between the raw input data and the high-level output.

2.2.2.2 MLP Training

MLPs are mostly used for supervised machine learning. A MLP is trained using a set of examples, a training set. Each example r contains a pair of input X_r and output Y_r. To understand how MLP is trained, first, let's consider a multi-perceptron network as depicted in figure 2.5.

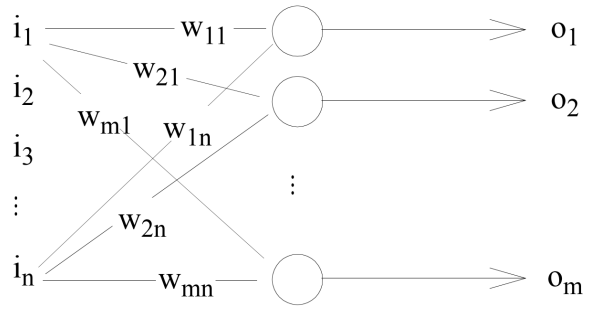


Figure 2.5: A layer of multi-perceptron network [21]

Assuming we have input X_r with n units I_i, where $i = 1 : n$, and m perceptrons O_j, where $j = 1 : m$. Thus $O_j = \sum w_{ji} I_i$, where w_{ji} is the weight connecting the unit i with the perceptron j. The objective is to tune the weights in order for the output to match the target response T_j (T_j equals to Y_r for the last layer). Therefore, a general learning rule can be written as:

$$\Delta w_{ji} = \eta \left(T_j - O_j\right) I_i = \eta \delta_j I_i \tag{2.13}$$

where η is a constant related to the speed of learning, T_j is the target response, O_j is the perceptron output, δ_j is the difference between the target response T_j and perceptron output O_j, and Δw_{ji} is the weight change due to X_r. This rule is called the delta rule. The delta rule provides a gradient descent search in the error space.

Now consider a multilayer network where one layer's output becomes another layer's input. The objective is to derive a rule of training for such networks. Assuming for a given layer:

$$e = \frac{1}{2} \sum_j \left(T_j - O_j\right)^2$$

$$Oj = f_j\left(\sum_k w_{jk} \cdot I_k\right)$$

$$S_j = \sum_k w_{jk} \cdot I_k$$

where f_j is the activation function. Note that T_j for the last layer is the output Y_r [21]. Then when we apply the chain rule of differentiation $\frac{\partial e}{\partial w_{ji}} = \frac{\partial e}{\partial S_j} \cdot \frac{\partial S_j}{\partial w_{ji}}$, the update rule for the weights for the layer l, where $l = 0 : L$ and L is the number of the layers, is given by:

$$\Delta w_{ji}^l = \eta \cdot f_j^{l'}\left(S_j^l\right) \cdot \left(\sum_{k_j} \delta_j^{l+1} \cdot w_{jk}^{l+1}\right) O_i^{l-1} \tag{2.14}$$

$f^{l'}$is the derivative of the activation function for layer l. The derivative exists since the activation function is assumed semilinear.

The training starts by first propagating the input X_r to the output of the MLP O, forward propagation. Then, we proceed to find the part of the error that influences the previous layer's error (back-propagate the error) δ_j^{l+1}. Hence comes the name backpropagation training.

The backpropagation method can be used to calculate gradients as long as the modules represent smooth functions between the inputs and the internal weights. Several research groups during the 1970s and 1980s discovered the idea that this could work [87].

2.2.2.3 Training, Validation, and Test Sets

Generally, the data used to create the final model is subdivided into several datasets. Mainly, three datasets are widely used in creating the model; training, validation, and test dataset. The training set is a set of examples used to fit the model's parameters, weights. Hence, the model is initially fit, through a training algorithm, on the training set. The training data set often consists of pairs of input and the corresponding output vectors.

During the training of the model, the model is used to predict the response of a second dataset called the validation set. While tuning the model parameters, the validation dataset presents an unbiased evaluation of the model fit to the training

dataset. Validation data sets can be used to early stopping the model training for regularization and avoid the common problem of overfitting, to be explained shortly. By early stop, we mean to stop the training when the validation error increases while the training error decreases.

Finally, to provide an unbiased evaluation of the trained model, a test dataset is used. The data in the test dataset should never be seen by the model during the training process to make sure the evaluation is unbiased.

2.2.2.4 Overfitting and Regularization

Overfitting is a modelling error that arises when a function matches a limited set of data points too closely. This means the model has a small error only when the training set is applied; however, testing it on any other datasets results in high prediction error. In other words, we want the model to generalize to different datasets other than the training dataset. Therefore, the model should fit the signal but not the noise in it to be able to avoid overfitting. Figure 2.6 depicts the concept of overfitting.

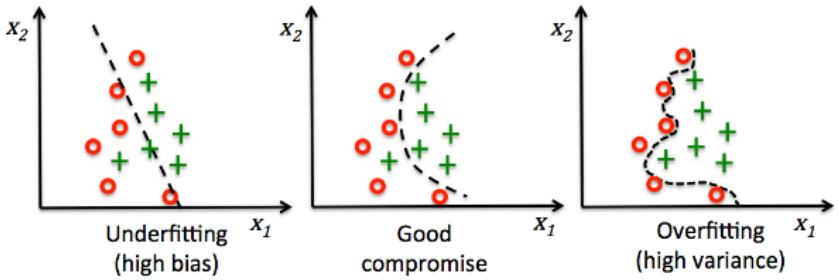


Figure 2.6: An example of the model overfitting. Assuming the model is trained to classify two sets, a set of circles and a set of crosses. The dashed line represents the model in three cases; underfitting, well-fitting and overfitting [1]

Regularization is the process of adding information to the model in order to solve or mitigate the problem of overfitting. One way to regulate the model is to add a term to the cost function that penalizes overly complex models.

2.2.3 Going Deeper

In conventional machine-learning techniques, a carefully engineered feature extractor is required. This feature extractor transforms the raw data into a proper internal representation such that the learning subsystem could detect patterns in the input [60].

In 2001, Csáji [16] represented a universal approximation theorem. The universal approximation theorem states that a single hidden layer is sufficient to approximate any function, but this is at the expense of exponentially many neurons, making it often computationally impractical. In this aspect, Bengio and Delalleau [18] proposed that deeper networks have the ability to preserve the network's expressive power at a decreased cost [101]. However, going deeper introduces the problem of vanishing gradient.

2.2.3.1 The Vanishing Gradient Problem

As more layers with conventional activation functions, such as sigmoid, are added to neural networks, the gradients of the loss function approach zero, making the network difficult to train.

Certain activation functions, such as the sigmoid activation function, squishes a large input space into a small output space between 0 and 1. A significant change in the sigmoid function input will, therefore, induce a slight change in the output. Hence, the derivatives become small. Figure 2.7 shows the sigmoid function and its derivate.

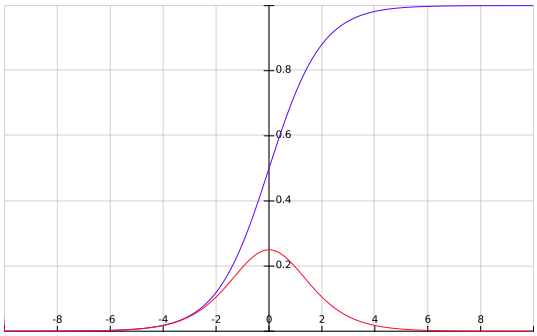


Figure 2.7: The sigmoid function, in blue, and its derivate, in red

This is not a significant problem for shallow networks with just a few layers. Nevertheless, it can cause the gradient to be too low for learning to work effectively when more layers are used.

MLPs find gradients using backpropagation. Backpropagation finds the network derivatives by moving from the final layer to the initial layer. By the chain rule, each layer's derivatives are multiplied through the network from the last layer to the initial layer to calculate the initial layer derivatives.

Small gradients mean that each training session will not efficiently change the

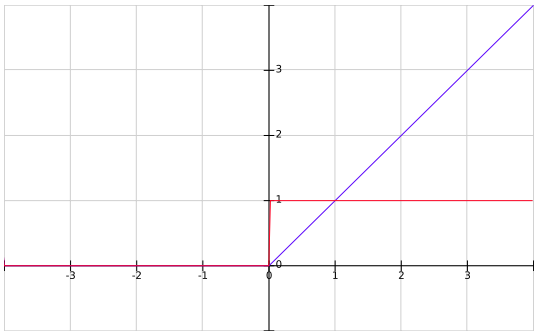


Figure 2.8: The ReLU function, in blue, and its derivative, in red

weights and biases of the initial layers. Since these initial layers are often essential to identifying the fundamental features of the input data, this can affect the entire network's inaccuracy.

2.2.3.2 ReLU Activation Function

The straightforward solution is to use other activation functions, like ReLU, which does not result in a small derivative. The ReLU activation function is defined as:

$$f(x) = \max(0, x)$$

where x is the input to the activation function. Figure 2.8 shows the ReLU function and its derivative.

The advantage of ReLU is that there will be no degradation of the error signal when the derivative is backpropagated. Nevertheless, the ReLU activation retains a characteristic of non-linearity.

However, there is a problem associated with ReLU in some cases. The problem is the derivative of ReLU is zero when the input is less than zero. Therefore, the back-propagated error can be cancelled out when there is a negative input to the activation function, which may affect the learning abilities. To overcome this problem, Leaky ReLU activation is introduced.

2.2.3.3 Leaky ReLU Activation Function

The Leaky ReLU activation is defined as:

$$f(x) = \max(\alpha x, x)$$

Where α is a small factor between 0 and 1. When x is below 0, the output will be αx. Hence, the derivative when x is below zero is α. This gives a chance for the gradient to propagate, even if the input is negative, which could potentially improve learning performance.

2.2.3.4 Residual Networks

Residual networks are another solution for the vanishing gradient problem. Residual networks provide a residual block as defined below:

$$y = (f(x) + x)$$

where f is the activation function, and x and y are the input and the output of the residual block, respectively. The residual block directly adds the input of the block to the output of the block, residual connection, in parallel to the activation function. This residual connection doesn't go through the activation function; hence, it gives a path to the gradient to back-propagate. We discuss the residual block in the next section in more detail.

2.2.3.5 Batch Normalization

Batch normalization also helps to mitigate the vanishing gradient problem. Batch normalization normalizes the input to the range that does not reach the outer edges of the sigmoid function. Hence the derivatives do not get relatively small, and the gradients can back-propagate. This approach is discussed in more detail in the next sections.

2.2.4 Fully Connected Networks

In regular Neural Networks, fully connected Neural Networks, each neuron in a layer is connected to all the neurons in the previous layer through different weights. The fully connected network has general-purpose connection patterns that make no assumption about the features in the input data. This makes the fully connected networks not to scale well with many common data structures, such as images, and also make them very expensive in terms of the required memory and computational power. For example, in CIFAR-10 [58], one of the most common image datasets, the images have a size of $32 \times 23 \times 3$, which is a relatively small size image. Hence, the first

fully connected layer will have $32 * 32 * 3 = 3072$ weights. Now when we scale to $200 \times 200 \times 3$ images, the number of the weights will be 120000 weights for only the first layer. Adding more layers will make the weights add up quickly. Such networks with a massive number of parameters (weights) face problems of slow training and chances of overfitting; the module tightly fits the training set and doesn't perform well with any other sets, besides the issues of excessive memory use and high processing power required. Other network structures emerged from the fully connected networks and achieved great success in a wide range of applications. These networks account for the features in the input data and employ them to reduce the number of parameters in the module. The most prevalent types of these networks are Convolutional Neural Networks and Recurrent Neural Networks.

2.2.4.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [60] consist mainly of convolutional layers. In a convolutional layer, each neuron is connected only to the local neurons in the previous layer. Besides, the set of local connection weights are replicated across all the neurons within the layer. This connection pattern is inspired by the visual cortex and proves most beneficial when the data has a spatial structure and the features being spatially local and equally likely to happen at any spatial position. The typical case for using CNNs is for images where the features are local and equally likely to exist at any position. The fewer number of weights make CNNs more efficient in terms of memory and compute power needed compared to fully connected networks. We discuss CNNs in more detail in the next sections.

2.2.4.2 Recurrent Neural Network

Recurrent Neural Networks (RNNs) [87] are a family of NNs that model dynamic systems, i.e., they are functions of the previous state, the current state, and the input. Hence, they can be used to model sequential data. In a recurrent network, units have self-connections or connections to units in the previous layers in addition to feedforward connections. This recurrency functions as short-term memory and allows the network to remember what happened previously. Since the units have self-connections (the output is connected to the input), the weights are shared across different time steps. Weight-sharing enables the model to be extended and applied to examples of distinct lengths. If we used a model with a separate parameter for

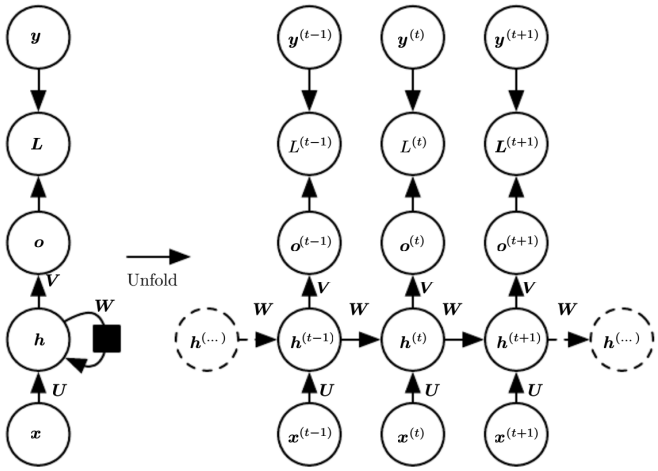


Figure 2.9: The computational graph to compute the training loss of a recurrent network that maps an input sequence of x values to a corresponding sequence of output o values. A loss L measures how far each o is from the corresponding training target y [31]

each time stamp, we will not be able to generalize this module to sequence with an arbitrary length. Besides, weight-sharing makes RNN more efficient in terms of memory and compute power needed compared to fully connected networks.

2.2.5 Convolutional Neural Networks

Machine Vision's challenging nature creates a specific class of Neural Networks that mimics Visual Cortex processing and is called CNNs [61, 62, 64]. CNNs are regarded as one of the best methods for understanding image content and showed state-of-the-art outcomes linked to image recognition, segmentation, detection and retrieval [14,56]. The main advantages of CNNs are hierarchical learning, automatic extraction of features, weight sharing, and multitasking [34, 73]. CNNs take advantage of natural signal properties: local connections, weight-sharing, pooling, and the use of many layers. With the capacity to extract automatic features, CNNs decrease the need to synthesize a distinct feature extractor. Therefore, with diminutive processing, CNNs can learn good internal representation from raw pixels.

2.2.5.1 CNN Architecture

Units in a convolutional layer are arranged in feature maps. Each unit within a feature map is connected through a set of weights, called a filter bank, to local patches in

the previous layer feature maps. All units within a feature map share the same filter bank. There are two reasons for this architecture. First, local value groups are often strongly correlated in array data, such as images, creating unique local motifs that can be identified easily. Second, the images local statistics are invariant to location.

CNN topology is split into various learning stages consisting of a combination of different components. In the next subsection, we explain the different basic components of CNNs.

2.2.5.1.1 Convolutional Layer The convolutional layer is composed of a set of convolutional kernels (each neuron acts as a kernel). It operates by splitting the image into small blocks known as receptive fields. The receptive fields are convolved with the filter weights (multiplying filter elements with appropriate receptive field elements) [10]. The operation of Convolution is described in (2.15).

$$F_k^l = I\left(x, y\right) * K_k^l \tag{2.15}$$

Where $I\left(x, y\right)$ is the input image, x and y represent the spatial locality, K_k^l represents the k^{th} convolutional kernel of the l^{th} layer, and F_k^l represents the output layer, feature map, of the k^{th} kernel.

Different kernels generate different feature maps per a convolutional layer. The feature maps are stacked to form a three-dimensional layer to be processed by the next convolutional layer, as shown in figure 2.10. Image division into small blocks allows extracting locally correlated pixel values. This information aggregated locally is referred to as feature motifs. A different set of features are obtained from the image with the same set of weights by sliding the convolutional kernel. Convolution operation can also be classified into distinct types based on filter size, padding type and convolution direction (strides) [60].

2.2.5.1.2 Activation Function Selecting a suitable activation function can speed up the learning process. Activation function is defined in (2.16).

$$T_k^l\left(x, y\right) = \mathrm{f}_A\left(F_k^l\left(x, y\right)\right) \tag{2.16}$$

where F_k^l is a convolution operation output, and $\mathrm{f}_A\left(\cdot\right)$ is an activation function. $\mathrm{f}_A\left(\cdot\right)$ adds non-linearity and returns an output T_k^l for the l^{th} layer and k^{th} feature map.

Different activation functions can be used for different layers. Various activation

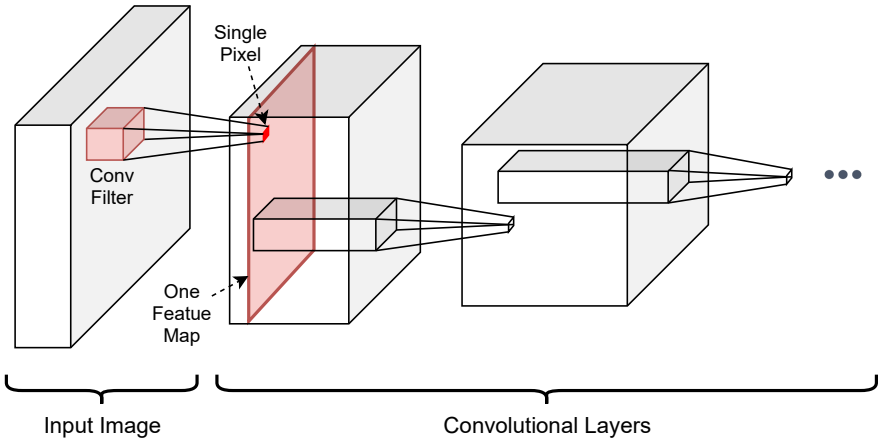


Figure 2.10: The operation of the convolutional layer

functions are used in literature to produce a nonlinear combination of features. This includes Sigmoid, Maxout, Tanh, ReLU, in addition to variations of ReLU such as PReLU, leaky ReLU, and ELU [32, 107]. ReLU and its variants are preferred over the rest of activations because it helps in overcoming the problem of vanishing gradient, as discussed earlier [43].

2.2.5.1.3 Pooling Layer Once features are extracted, their precise location becomes less fundamental as long as their approximate position is maintained relative to others [56]. Pooling (also known as downsampling) is a local operation like convolution. It basically sums up information in the receptive area and produces the dominant response in this local area [66]. The forward propagation of the pooling layer results in reducing an N×N pooling block to a single value (the winning unit). The backpropagation then computes the error produced by the winning unit. To keep track of the index of the winning unit, the index is recorded during the forward propagation to be used for gradient calculation during the backpropagation. The pooling operation is shown in (2.17).

$$Z_k^l = \mathrm{f}_p\left(F_k^l\left(x, y\right)\right) \tag{2.17}$$

where Z_k^l is the k^{th} output feature map, $F_{x,y}^l$ represents the l^{th} input feature map, and $\mathrm{f}_p\left(\cdot\right)$ defines pooling operation type. Different types of pooling functions are used to extract translational invariant features such as max, average, L2, and overlapping pooling [9, 102].

The pooling layer results in reducing the size of feature maps. Reducing the size

of feature maps regulates the network's complexity.

2.2.5.1.4 Fully Connected Layer The fully connected layer is usually added for classification purposes at the end of the network [56]. It takes input from the previous layer and globally analyzes output from all previous layers [71]. It creates a nonlinear combination of chosen features used to classify data. The fully connected layer is a global operation, unlike pooling and convolution [85].

2.2.5.1.5 Dropout One of the significant issues in training a relatively large network is the co-adaptive problem. In the co-adaptive problem, some of the weights tend to learn more than the other weights (some of the weights are updated more fequently than the others). These weights become dominant during the model evaluation, while other weights have less effect on the prediction. The co-adaptive problem can not be prevented using the traditional regularization discussed earlier.

Dropout is a regularization method that mitigates the co-adaptive problem and helps regularize the model [94]. During training, some of the layer outputs are randomly disregarded or dropped out. This makes the layer act as a layer with a different number of neurons and connectivity to the prior layer, which may help to prevent some weights from becoming dominant. This conceptualization suggests that dropout might break up situations where network layers co-adapt to the previous layers, and hence making the model more robust.

2.2.5.1.6 Batch Normalization Batch normalization is used to approach the problems related to covariance shift within feature maps. The internal shift of covariance is a change in the values of hidden units distribution. This shift slows the convergence (forcing the learning rate to a low value) and needs careful parameter initialization. Batch normalization is implemented during the training by measuring each layer input variable's mean and standard deviation and using these statistics to perform the normalization. The equation (2.18) shows batch normalization process.

$$N_k^l = \frac{T_k^l}{\sigma^2 + \sum_i T_i^l} \tag{2.18}$$

where T_k^l is the input feature map, N_k^l is the normalized feature map, and σ represent variation in the feature map.

Batch normalization unifies the feature map values distribution by moving them

to zero mean and unit variance [48]. It also smoothes the gradient flow and acts as a regulating factor that enhances network generalization without depending on dropout.

2.2.5.1.7 Residual Layer Residual blocks or skip connections were introduced in 2016 [37] as a building block for ResNet. The main idea of the residual block is to introduce an identity shortcut connection that can skip one or more layers. Figure 2.11 shows the residual block architecture.

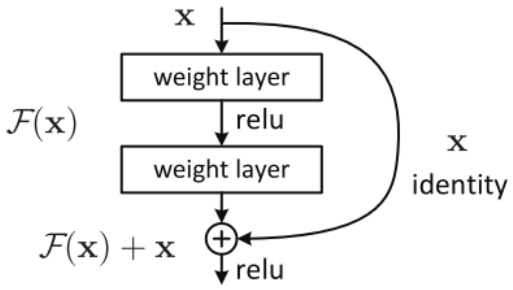


Figure 2.11: Residual block architecture [37]

Typical ResNet models consist of double or triple layer skips containing ReLU activation function and batch normalization in between the layers [105]. Additional weights may be employed to learn the skip; these models are called HighwayNets [95]. Also, several skips may be used in parallel; these models are known as DenseNets [46]. The motivation for using skip connection is to mitigate the problem of vanishing gradients. This is achieved by reusing activations from a previous layer until the skipped layer learns its weights. Skipping simplifies the network efficiently, using fewer layers in the initial stages of training. It speeds training by mitigating the effects of vanishing gradients since there are fewer layers for the gradient to propagate through. The network then progressively restores the layers that have been skipped as they learn the feature space.

Stacking residual blocks does not degrade the network performance because we are stacking identity mapping. This indicates that constructing a deeper model should not result in a training error higher than its shallower counterparts.

2.2.5.2 CNN Examples

CNNs have been used in many applications achieving high performance in both classification and regression problems. The biggest boom in using CNN for image classi-

fication and segmentation happened after it was found that CNN's representational capability could be improved when increasing the depth [59]. Research in CNNs has been accelerated by the availability of a significant amount of data and improvements in hardware processing units. Many CNN architectures are introduced in the literature by varying and tweaking different CNN parameters and blocks. Hence, the architectures differ in the number of the layers, the number of filter maps per layer, the size of the filters, the residual connections, etc.

In table 2.1, we highlight some of the remarkable architecture in the literature.

Table 2.1: The architectural evolution of CNN

Architecture Name	Year	Main contribution	Depth	Parameters
ConvNet	1989	LeCuN et al. introduced ConvNet as the first multilayered CNN [61]. Used supervised training using the Backpropagation algorithm compared to its predecessor unsupervised reinforcement learning scheme [28]. Showed good results only for problems related to handwritten digits and zip code recognition [115].	-	-
LeNet	1998	LeCuN enhanced ConvNet and used it to classify characters in document recognition [63]. This enhanced architecture was named LeNet. Consisting of five alternating convolutional and pooling layers, followed by two fully connected layers. Used the average-pooling.	7	60 K

		This architecture has become the standard 'template': stacking convolutions and pooling layers, and ending the network with one or more fully connected layers.		
AlexNet	2012	AlexNet (figure 2.12) was introduced by Krizhevesky et al. [59]. Deeper and wider (more filters per layer) than the LeNet. AlexNet has 8 layers; 5 convolutional and 3 fully connected. Were the first to implement Rectified Linear Units (ReLUs) as activation functions. In the initial layers, AlexNet used larger filters (11×11 and 5×5).	8	60 M
VGG	2015	Simonyan and Zisserman introduced VGG in [92]. Homogeneous topology. VGG used a stack of 3×3 filters layer to replace the 11×11 and 5×5 filters. VGG experimentally proved that the successive small filters did not affect the modelling ability of the CNN. The use of small size filters reduced the number of parameters, hence give an additional advantage of low computational complexity. Because of its simplicity, homogeneous topology and increased depth, VGG became famous. VGG set a new research trend for working with CNN with lower size filters.	19	138 M

		High computational costs were the major limitation associated with VGG.		
GoogLeNet (Inception-V1)	2015	GoogLeNet [97] introduced Inception modules. The inception module uses three branches with different filters (1×1, 3×3 and 5×5) to capture different features. The outputs of the three branches are concatenated. To reduce the dimensionality, a 1×1 filter is used to output fewer dimensions (fewer feature maps). The idea of using multi-path and then concatenate them become known as the split, transform and merge blocks. GoogLeNet architecture's main objective was to obtain high accuracy with lower computational costs. The number of the parameters in each branch can be chosen in such a way that the total number of the parameters is less than using a single filter, yet the network can perform better because we are capturing spatial information at multiple spatial resolutions through different filters. GoogLeNet's main drawback was its heterogeneous topology, which must be customized from module to module.	22	4 M
		Srivastava et al. suggested a deep CNN called the Highway Network [95].		
Highway Networks	2015		19	2.3 M

Highway Network's aim is to utilize the depth for learning enhanced feature representation by presenting cross-layer connectivity.
Highway Network enables the unrestrained flow of information across layers by adding two gating units within a layer which were inspired by Long Short Term Memory (LSTM).
Highway Network introduced the following transformation to replace the plain network:
$y = H(x, W_H) \cdot T(x, W_T) + x \cdot (1 - T(x, W_T))$
where x is input, H is the direct path transform function, T is the Transform Gate, and W_H and W_T are the weight matrices associated with H and T respectively.
T formally is the sigmoid function.

- When $T = 0$, the input passes to the output directly which creates an information highway.

- When $T = 1$, the non-linear activated transformed input passes to the output.

By learning W_T, the network can adaptively pass H or just pass x to the next layer.

		Highway network obtained similar accuracy but with a much fewer number of parameters compared to other networks presented in that time on MNIST and CIFAR datasets [56].		
ResNet	2016	ResNet [105] is based on Residual blocks (figure 2.11), discussed earlier. Residual block is cross-channel connectivity that facilitates learning by regulating information flow across blocks. ResNet Architecture used this concept to train 152 layers network. This depth is achieved because ResNet can avoid the problem of vanishing gradient.	152	6.8 M
		ResNet concept is close to highway networks. However, the gates are data-independent and parameter-free. ResNet has reduced computational complexity compared to VGG, even with increased depth. ResNet won the 2015-ILSVRC competition using the proposed 152-layers deep CNN. ResNet was 20 times deeper than AlexNet and 8 times deeper than VGG [56]. ResNet is among the first to use batch normalization.	110	1.7 M
WideResNet	2016	WideResNet proposed by Zagoruyko and Komodakis [111]. WideResNet used the residual block to make ResNet wide rather than deep.	28	36.5 M

		WideResNet expanded width by adding an extra factor k to control network width (figure 2.13). Even though deep residual networks have enhanced the network representational capability, they have certain defects such as inactivation of many feature maps, gradient vanishing and exploding problems, and time-intensive training. An empirical study by Zagoruyko and Komodakis showed that WideResNet has twice as many parameters as ResNet but that WideResNet could train better. In WideResNet, adding a dropout in-between the convolutional layers instead of inside a residual block made learning more effective.		
ResNeXt	2016	Xie et al. proposed ResNeXt [106], sometimes known as Aggregated Residual Transform Network, as an improvement over the Inception Network. Xie et al. exploited the split, transform and merge topology by introducing cardinality. ResNeXt used multiple transformations within a split, transform and merge block. These transformations are defined in terms of cardinality (figure 2.14).	101	68.1 M

		WideResNet used the concept of residual blocks. However, ResNeXt contribution is the adding of parallel branches within each module. Xie et al. demonstrated that increasing cardinality enhances performance.		
Squeeze and Excitation Networks	2017	In order to select feature maps relevant to object discrimination, Hu et al. proposed a new architecture, Squeeze and Excitation Network (SE-Network) [45]. SE-Network is based on a new block named SE-block. SE-block suppresses less essential feature maps and adds high weight to feature maps specifying the class. This block's operation is divided into two tasks; squeeze and excitement (figure 2.15). Convolution kernel captures information locally but ignores contextual relationships of features outside of this receptive field. To get a global perspective of feature maps, the squeeze block produces channel-wise statistics by suppressing spatial information of the input. Squeeze operation uses global average pooling to produce feature map wise statistics. On the other hand, the excitation procedure uses two-layer feedforward NN to assign weights to feature maps.	152	27.5 M

		The operation is described in the equation: $V_M = \sigma\left(w_2, \delta\left(w_1 D_M\right)\right)$ where V_M is the weight for each feature map, D_M is the global average pooling of the feature map, δ refers to the ReLU function, σ refers to the sigmoid function, and w_1 and w_2 are regulating factors used to aid the generalization and limit the model complexity. SE-block exploited the gating mechanism using the sigmoid activation function, which models interdependencies between the feature maps. SE-Network achieved a record error decrease on the ImageNet dataset.		
DenseNet	2017	Another attempt to solve the vanishing gradient problem is DenseNet [46]. DenseNet linked each layer in a feedforward fashion to each other layer. Thus, structure maps of all previous layers were used in all subsequent layers as inputs (figure 2.16).	190	25.6 M
		DenseNet concatenates previous layers' features instead of adding them. Therefore, DenseNet has the potential to differentiate between information added to the network and information that is preserved.	250	15.3 M

Directly connecting each layer to the gradients of the loss function augments the flow of information throughout the network.

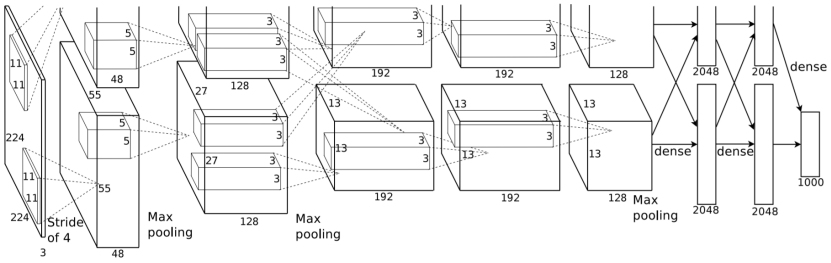
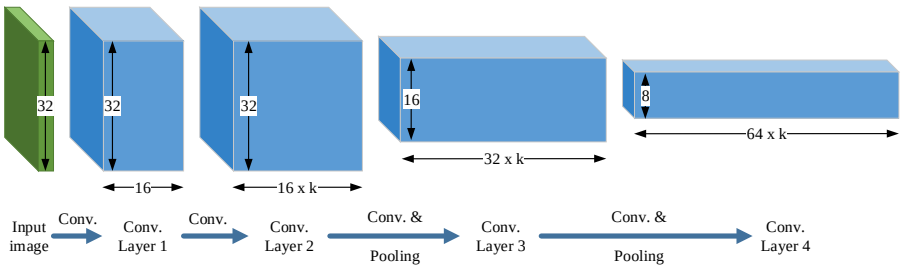


Figure 2.12: Basic layout of AlexNet architecture [59].

2.2.6 Super-resolution

Super-resolution concepts offer the potential of resolution beyond the classical limit [20]. Hence, super-resolution is the retrieval of spectral information outside the system bandpass. However, it is an ill posed problem. It is presumed that the system bandpass is determined by a hard limit beyond which there is no spectral information; either the frequency response is identically zero, or noise masking imposes an effective band limit. Unfortunately, significant super-resolution improvements are not practical in an image using analytical approaches [20]. The limit is set by the super-resolution problem's high sensitivity to noise and system errors. However, learning based approaches (e.g., CNNs) attempt to solve the inverse problem and have shown excellent results .



[109]

Figure 2.13: Variations residual blocks used in WideResNet

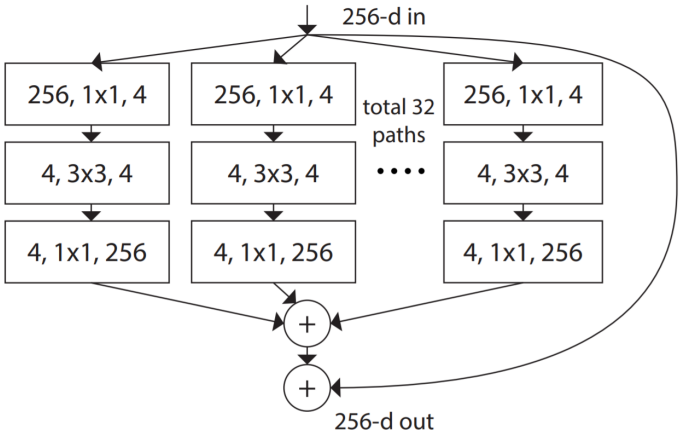


Figure 2.14: ResNext building block [106]

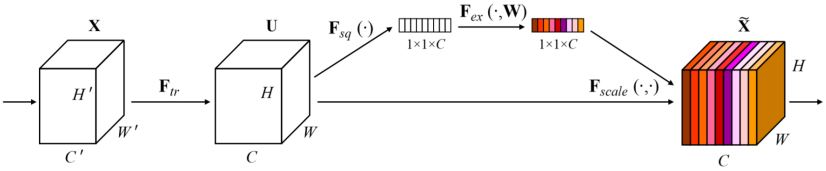


Figure 2.15: Squeeze and Excitation block [45]

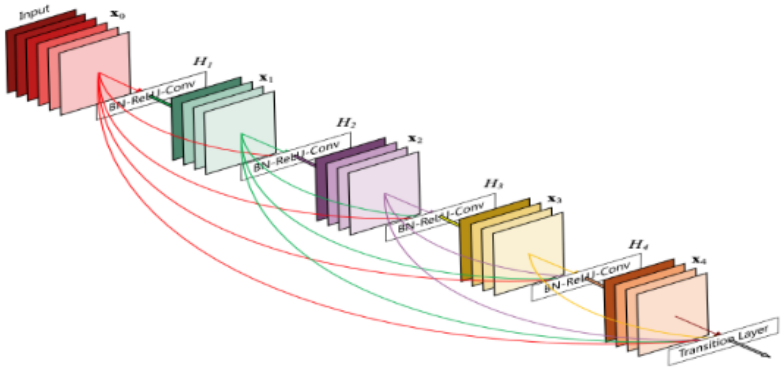


Figure 2.16: DenseNet architecture [46]

2.3 Quantum Optimization

Quantum computing methods are potential solvers for the optimization problem of the phase unwrapping explained previously. Hence, this section introduces the use of quantum computing for optimization. First, we introduce the concept of optimization. Then, we briefly summarize how quantum techniques are used for optimization (quantum annealing).

2.3.1 Optimization

Optimization is the process of finding the minimum value, or the maximum value, of a function of several variables. This function could be subject to a set of constraints that need to be satisfied.

Even though a function has a minimum value, known as a global minimum, most functions are subject to several local minima. A function's local minimum is a point where the value of the function is smaller than or equal to the value at neighbouring points, but possibly this value is greater than a distant point. A function's global minimum is a point where the value of the function is smaller than or equal to the value at all other possible points (see figure 2.17).

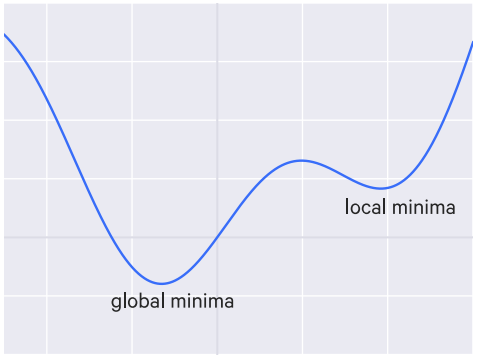


Figure 2.17: A local minimum vs. the global minimum of a function

The function to be optimized is called the objective function. The objective function may also be referred to as the cost function or energy function if the optimization goal is to minimize the function value.

Optimization problems can be considered as searching a topology for the globally

highest point (if the objective is to maximize) or the globally lowest point (if the objective is to minimize).

To better understand the optimization problem, consider the native approach for optimization, the greedy random optimization algorithm, with the objective of the optimization to minimize. This optimization algorithm starts at some initial position in the topology, then randomly tries to switch into a new position, usually a close position, and checks whether it is better (has lower value) or not. If it is lower than the current value, it sets the new position as the current solution. Otherwise, the algorithm will try another random position. This process goes on until the exit condition is satisfied, usually because a minimum has been reached. A simple greedy-descent strategy that constantly steps to the least-cost neighbour works well if the landscape is smooth and generally trending downhill toward the optimal solution. However, if the landscape has a large number of local minima, the greedy strategy may get stuck in one of these local minima, go in circles, and never find the best solution. Hence, the position found is not necessarily the global minimum; instead, it may be a local minimum.

This algorithm is greedy since a position with a lower value is always chosen, even if it is only insignificantly lower. Therefore, it might be easy for this algorithm to get stuck in local minima, as any step away from the found point seems worse, even if the global minima is right around the corner.

2.3.2 Simulated Annealing

Simulated annealing is similar to the greedy random algorithm; however, it has a probabilistic step that can move the algorithm out of the local optima (i.e., local minima or local maxima).

Annealing is a metalworking process in which the metal starts at a very high temperature and slowly cools down. Simulated annealing utilizes a similar approach to control its probabilistic step. A simulated annealing algorithm begins with a high "temperature" that "cools" down as progress is made.

The algorithm makes a random move, much like the greedy random method. If the new position is better than the current value (has lower energy in case of energy minimization), it is accepted as the new position. If the change is worse than the current value, then it can still be accepted; the likelihood of this is dependent on the current temperature T, the current energy e, and the preceding energy e':

$$P(e, e^{'}, T) = \exp\left(\frac{-\left(e^{'} - e\right)}{T}\right)$$

This probability of accepting a worse position gives the algorithm a chance to escape local minima.

Each random move is called an iteration, whether accepted or not. The temperature decreases after each iteration according to a cooling schedule. There are various cooling schedules; this includes linear cooling schedules, logarithmic cooling schedules, geometrical cooling schedules, and arithmetic-geometric cooling schedules [75]. The cooling schedule depends on the current temperature, the current iteration, the initial temperature, and how the new random state is chosen. For instance, if the random step distribution function is a Cauchy function, a semi-local search with occasional long jumps, the cooling schedule is inversely linear. This is faster than the cooling schedule that is inversely proportional to the logarithmic function of time, suitable for the Boltzmann machine [98]. The Boltzmann machine is strictly a local search.

2.3.3 Quantum Computing

Quantum computing exploits the laws of quantum mechanics to process information [81]. In contrast with classical computers, which use bits to process information, quantum computers use quantum bits, or qubits. Qubit is the basic unit of quantum information that encodes the zero and the one into two distinct quantum states. This is realized by the quantum phenomena of superposition and entanglement.

Superposition is the ability of a quantum system to exist in multiple states at the same time. Entanglement is a strong correlation that exists between quantum bits. This correlation is strong so that two or more quantum bits can be linked in unity, even if separated by large distances. Because of superposition and entanglement, a quantum computer can process an enormous number of calculations concurrently.

A qubit holds the probability of the states one and zero. Therefore, the results generated by a quantum computer are not deterministic. This means the measurement process is essentially probabilistic and requires special algorithms to compensate.

There are three basic approaches for quantum computing: Universal Quantum Gate Model, Adiabatic Quantum Computing, and Quantum Annealing. All three models give different perspectives on quantum computing's practical applications. Before addressing the different approaches for quantum computing, we introduce

the basics of quantum computation by making the connection between the Boolean Algebra used in classical machines and quantum mechanics used in quantum circuits.

2.3.3.1 Basics of Quantum Computation

In this section, we introduce the mathematical framework of quantum computation and highlight the fundamental contrasts between quantum and classical computation models [77].

2.3.3.1.1 Qubits The concept of state is the most prominent distinction between classical and quantum computation. A classical algorithm's state is represented by a register R of bits, each of which has a value of 0 or 1. A quantum algorithm's state is represented by a register Q of qubits, each of which has unusual properties.

Superposition is one of these properties, which means a qubit can be in both states 0 and 1 at the same time. We may define a superposition state $|\phi\rangle$ that corresponds to a qubit state vector comprise of two complex numbers α and β. The state vector is a linear combination of two basis states, which are conventionally $|0\rangle$ and $|1\rangle$ basis: that is, $|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.

In matrix notation, the state vector and the basis become:

$$|\phi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha & \beta \end{pmatrix}^T$$

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix}^T$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \end{pmatrix}^T$$

According to quantum mechanics, superposition states are impossible to observe directly. Instead, we imagine that when a qubit is measured with respect to a given basis, it instantly collapses to one of the basis states; to state $|0\rangle$ with probability $|\alpha|^2$ and to state $|1\rangle$ with probability $|\beta|^2$, where $|\alpha|^2 + |\beta|^2 = 1$. Hence, the basis states are called the observable states of a qubit.

A state vector can be viewed as a point on the three-dimensional unit sphere, known as the Bloch sphere (see Figure 2.18). The north and south poles, by convention, correspond to the basis states of $|0\rangle$ and $|1\rangle$, respectively. In addition to

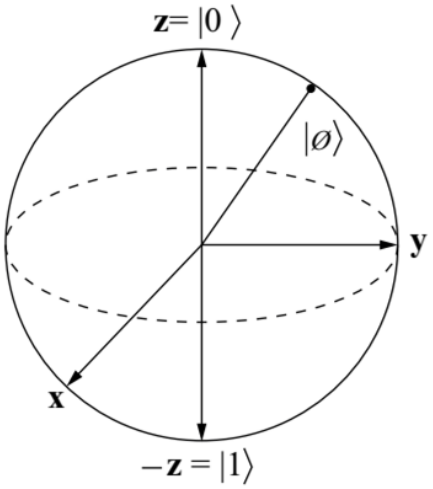


Figure 2.18: Figure 1: A Bloch sphere [77]

the standard basis $|0\rangle$ and $|1\rangle$, there are two alternative pair of bases; $|x_+\rangle$ and $|x_-\rangle$ that lie along the positive and negative x-axis, and $|y_+\rangle$ and $|y_-\rangle$ that lie along the positive and negative y-axis.

2.3.3.1.2 Quantum Gates and Operators A classical logic gate operates on classical bits. Similarly, a quantum logic gate operates on qubits, causing the state vector of each qubit to transform, rotate, to a different location on the Bloch sphere. The Pauli matrices, $\sigma^x\sigma^y\sigma^z$, represent the three fundamental rotation operators that can be performed on a qubit. These are represented in matrix notation, along with the identity transformation I, by:

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\sigma^y = \begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix}$$

$$\sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

For instance, applying the Pauli gate σ^x to a given qubit vector $|\phi\rangle = (\alpha, \beta)^T$

induces a reflection about the $x-y$ plane, causing α and β to switch places. Therefore, we have:

$$\sigma^x |1\rangle \rightarrow |0\rangle$$

$$\sigma^x |0\rangle \rightarrow |1\rangle$$

which is equivalent to the classical NOT gate. One fundamental difference between the quantum gates and the classical gates is that the quantum gate must be ***reversible***. That is, any quantum logic operator's input qubits must be possible to identify by reading the output qubits. This is not the case of classical gates, except for the NOT gate.

2.3.3.1.3 Eigenvectors and Eigenvalues. An eigenvector v for a given square matrix A is a non-zero vector that, if multiplied by A, results in a constant λ multiple of v. That is, $Av = \lambda v$. The constant λ is called the eigenvalue of the matrix A corresponding to the eigenvector v.

Each Pauli matrix has two eigenvalues: $+1$ and -1. That is:

$$\sigma^z |0\rangle = +1 |0\rangle$$

$$\sigma^z |1\rangle = -1 |1\rangle$$

$$\sigma^x |x_+\rangle = +1 |x_+\rangle$$

$$\sigma^x |x_-\rangle = -1 |x_-\rangle$$

$$\sigma^y |y_+\rangle = +1 |y_+\rangle$$

$$\sigma^y |y_-\rangle = -1 |y_-\rangle$$

2.3.3.1.4 Collections of Qubits We can combine two qubit states, $|\phi_1\rangle$ and $|\phi_2\rangle$, to form a new state $|\phi\rangle = |\phi_1\phi_2\rangle$ by applying the tensor product $\otimes$ on the two

state vectors, $(\alpha_1, \beta_1)^T$ and $(\alpha_2, \beta_2)^T$, of the two states, $|\phi_1\rangle$ and $|\phi_2\rangle$. That is:

$$\begin{aligned}(\alpha_1, \beta_1)^T \otimes (\alpha_2, \beta_2)^T &= (\alpha_1\alpha_2, \alpha_1\beta_2, \beta_1\alpha_2, \beta_1\beta_2)^T \\ &= \alpha_1\alpha_2 |00\rangle + \alpha_1\beta_2 |01\rangle + \beta_1\alpha_2 |10\rangle + \beta_1\beta_2 |11\rangle \end{aligned} \tag{2.19}$$

where

$$|\alpha_1\alpha_2|^2 + |\alpha_1\beta_2|^2 + |\beta_1\alpha_2|^2 + |\beta_1\beta_2|^2 = 1 \tag{2.20}$$

$|\phi\rangle$ is a combination state of individual states. It can be represented by a vector of coefficients of length $N = 2^n$. For example, in (2.19), we have two independent states $|\phi_1\rangle$ and $|\phi_1\rangle$. $|\phi\rangle$ can represent the combination of the two states with the four coefficients shown in (2.19).

The probability that the qubit pair will be observed in each possible state is represented by the coefficients in (2.19).

2.3.3.1.5 Entanglement Entanglement is another feature that distinguishes quantum computation from classical computation: in which, quantum particles, such as qubits, can interact in a way after which they are no longer independent. The Bell state is a well-known example of this phenomenon, which explains entanglement as follows:

$$\frac{1}{\sqrt{2}} (|00\rangle + |11\rangle)$$

The probability of observing $|00\rangle$ is equal to the probability of observing $|11\rangle$ which equal to 0.5. While the probability of observing $|10\rangle$ or $|01\rangle$ is zero. What entanglement tells us is that if we measure one of the qubits as zero in a Bell state, the other qubit is zero. The same happens if the measurement results in one of the qubits being one; the other qubit will be one. Thus, if two qubits are entangled and far away from each other, measuring one determines the state of the other one uniquely. Due to entanglement, probabilities associated with multi-qubit states, such as $|11\rangle$ cannot be reduced into products of individual probabilities, such as $\beta_1\beta_2$ in equation (2.19).

2.3.3.1.6 Quantum Circuits Assume that we have a quantum register Q with n qubits. A vector $|\phi\rangle$, of length $N = 2^n$, can represent the superposition state of Q.

As showed by Deutsch [19], any classical function f defined over n bits can be computed using a quantum circuit composed of reversible quantum logic gates. Let C_f be a circuit that is applied to register Q, yielding a new superposition state, $C_f |\phi\rangle = |\phi'\rangle$. This circuit C_f can be represented by a matrix of size $N \times N$. One example is the Controlled-NOT gate (C-NOT) which operates on two qubits. If the first qubit (the control qubit) is $|1\rangle$, the CNOT gate toggles the second qubit (the target qubit). Otherwise, it leaves the second qubit unchanged. This circuit can be represented as follows:

$$C_{NOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

2.3.3.1.7 The Gate Model The quantum gate model of computation is based on the notion of a register Q, with n qubits, and quantum circuits, $C1; C2; C3 \ldots$, that run in parallel and in series, operating on the register Q. The computation involves three steps; initializing Q to a known state, applying the circuits to produce the required state transformations, and measuring the value of the output.

The measuring process results in a probabilistic "collapse" to a classical state, which is considered the computation's answer.

Superposition enables Q to hold all $N = 2^n$ states at the same time. Entanglement enables the quantum circuit to act on all n bits (and consequently on all 2^n states) in a constant time. In comparison, a classical computing model would need to update 2^n classical registers to reach the same manipulation on a classical state register R.

This quantum parallelism is the fundamental source of the expected massive speedups in computing time over classical computers. The interplay between entanglement and superposition allows a quantum algorithm to enhance the probability of the right answer. Sara Gamble [82] provides an elegant description of this: "In reality a quantum computer leverages entanglement between qubits and the probabilities associated with superpositions to carry out a series of operations (a quantum algorithm) such that certain probabilities are enhanced (i.e., those of the right answers) and others depressed, even to zero (i.e., those of the wrong answers). When a measurement is made at the end of a computation, the probability of measuring the

correct answer should be maximized. The way quantum computers leverage probabilities and entanglement is what makes them so different from classical computers."

However, the "quantum" algorithm needs to be cleverly designed so that only the true solution remains at the end of the computation. Several problems can benefit from quantum parallelism. For instance, Shor's algorithm, which solves the integer factorization problem, has a polynomial-time complexity on a quantum computer. In contrast, the factorization problem is considered an NP-hard problem on a classical computer. Similarly, quantum simulation benefits from quantum computation, since a quantum computer can express natively the problem itself. Other problems cannot benefit or benefit marginally from quantum computation. Such examples include the parity problem that determines whether a string of 0s and 1s has an even or an odd number of 1s. The parity problem has the same scaling in both quantum and classical environments. The search problem benefits marginally from a quantum environment. A classical search scales linearly with the number of search items while the quantum search scales as the square root of the number of search items [82].

2.3.3.1.8 Quantum Dynamical Systems Consider the case where the n qubits in register Q are particles in a quantum dynamical system that evolves due to external forces. Some forces come from outside sources, while others result from interactions between qubits (including entanglement). These forces are completely characterized at any time t by a time-varying Hamiltonian H_t. The Hamiltonian of a system is an operator that corresponds to the system's total energy. In classical mechanics, the system energy is defined as the sum of the kinetic and potential energies. In quantum mechanics, the elements of this energy are transformed into the corresponding quantum mechanical operators. The Hamiltonian comprises the operations associated with kinetic energy and potential energy. Mathematically, the time-varying Hamiltonian H_t is a Hermitian matrix. This means it has some properties related to symmetry and decomposability. One such property is that all its eigenvalues are real.

Each observable state is associated with energy, which is a real scalar that varies depending on H_t. The energy spectrum is defined as the set of all possible energies in the system, with a maximum size of N. The ground state is the state with the lowest energy among all basis states (observable states). An excited state is one that is not the ground state.

In this framework, the Hamiltonian serves a dual purpose:

- It can be thought of as an operator that alters the state of the system. It should

be noted, however, that Hamiltonians operate in continuous, not discrete-time.

- The eigenstates of the matrix H_t resemble the observable states of the system, and their associated eigenvalues correspond to the energies of those eigenstates. Hence, if $|\phi_t\rangle$ is an eigenstate, we can measure its energy λ because $H_t\,|\phi_t\rangle = \lambda|\phi_t$.

A quantum computation system seeks the ground state of such a Hamiltonian function. The mechanism (e.g., a quantum computer or a quantum annealer) embodies the physics of the quantum computation that seeks the ground state of the Hamiltonian.

2.3.3.2 Universal Quantum Gate Model

Deutsch [19] suggested a universal computation model (the quantum gate model) that can simulate any Turing machine computation with only a polynomial cost overhead.

The universal quantum gate model focuses on constructing quantum structures using stable qubits and solving the problems with quantum circuits today. A universal gate quantum computing framework is based on building basic quantum circuit operations, i.e., quantum gate or quantum logic gate. These quantum gates can be put together to construct any sequence, running increasingly complex algorithms. Therefore, a quantum gate is a primitive quantum circuit that operates on a small number of qubits, as discussed in subsection 2.3.3.1.

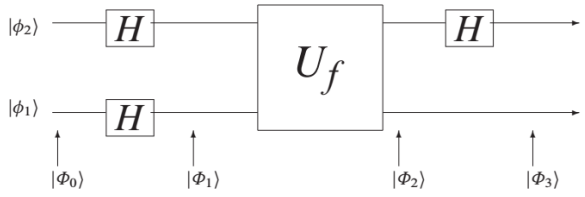
A set of gates is said to be universal if any operation (transformation) can be approximated, up to arbitrary accuracy, using only those gates. For instance, a basic set of universal two-qubit quantum gates is the controlled-NOT gate, the Hadamard gate (H), a phase rotation gate (R), and $\pi/8$ gate. However, one of the main concerns is the efficiency of implementing an arbitrary transformation using universal gates. That is, how many gates are required to create a given function? Does it grow polynomially or exponentially? Nielsen and Chuang demonstrated in [81] that there are unitary transforms that require exponentially many gates to approximate.

Quantum speedup is based on the notion that quantum algorithms may be able to take advantage of these properties to perform the required computation in asymptotically fewer basic operations than any classical algorithm is capable of. However, it is exceedingly difficult to maintain qubits at a given state for a long time. This problem is increasing as the qubit count increases.

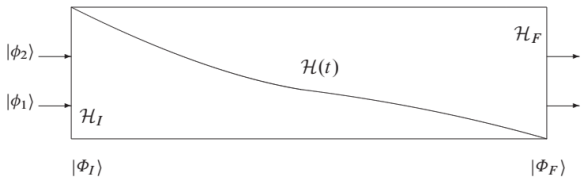
2.3.3.3 Adiabatic Quantum Computation (AQC)

AQC is an alternative computation model to the quantum gate model. AQC shows potential as being more "realizable" and probably more "analyzable" than the gate model through a set of analysis techniques based on quantum mechanics. Results show that the two models are polynomially equivalent [2, 24].

Because quantum computation is probabilistic, we examine the trade-off between the time and the probability that the output is correct rather than the computation time of a given algorithm. One fundamental difference between the two models is their discrete versus analogue nature, which results in very different algorithms, as shown in Figure 2.19.



(a) Gate model.



(b) AQC model.

Figure 2.19: The gate model vs. AQC [77]

Figure 2.19.a shows a simplified quantum gate model diagram. The two horizontal lines represent the evolution of two qubits through time. The boxes represent the quantum circuits that change the states of the two qubits. The notation Φ_i along the bottom represent the computation at different processing time i, before and after the application of each quantum circuit.

Figure 2.19.b shows the analog computation of AQC. Notice the absence of the gates and the discrete-time steps. Instead, the qubit states evolve gradually over time in response to certain forces described by Hamiltonians. The algorithm carries on a transition from the initial Hamiltonian H_I to the final Hamiltonian H_F. At the end of the transition, the qubit states are read. As we mentioned in Section 2.3.3.1.8, each observable qubit state is associated with energy; the ground state has the lowest

energy of all states.

2.3.3.3.1 AQC Algorithms AQC algorithms are intended to solve optimization problems formulated as follows; Given $f(x)$ an objective function $f : D^n \rightarrow R$ defined over n variables $x = x_1 \ldots x_n$ belong to some discrete domain D, find an assignment of x that minimizes $f(x)$.

To solve an optimization problem P associated with an objective function $f(x)$, an AQC algorithm works on a quantum register Q of n qubits. At time t the quantum register Q has the superposition state $|\phi_t\rangle$. We can describe the algorithm through a time-varying Hamiltonian $H(x)$ defined by three components:

1. An initial Hamiltonian H_I is chosen in such a way that the system's ground state is easily found.

2. A final Hamiltonian H_F that encodes the objective function. The objective function is encoded in such a way that the ground state of Q is an eigenstate of H_F with the smallest eigenvalue. Thus, a ground state of Q corresponds to an optimal solution to P.

3. An adiabatic evolution path. That is a function $s(t)$ that decreases from 1 to 0 as time progresses from 0 to t_f for a given amount of time t_f.

The Hamiltonian $H(t)$ produces a gradual transition from H_I to H_F as follows:

$$H(t) = s(t) H_I + (1 - s(t)) H_F$$

According to the Adiabatic theorem [77], if the qubits start in the ground state and the transition is carried out slowly enough, the system will finish in the ground state with high probability. The idea is to define H_I in such a way that finding a ground state is simple, and to define H_F in such a way that its ground state corresponds to an optimal solution to a given optimization problem P.

2.3.3.4 Quantum Annealing (QA)

QA is a quantum computing method used to find the optimal solution to specific combinatorial optimization problems [26]. This is achieved by using properties of quantum mechanics such as quantum tunnelling, entanglement, and superposition.

The term quantum annealing refers to a heuristic search method for solving combinatorial optimization problems. Quantum annealing is similar to the more well-known simulated annealing (SA) heuristic as both are intended to mimic natural physical processes.

Thus, QA serves as a conceptual link between AQC and classical optimization. Both terms, AQC and QA, are used interchangeably in some works. However, there are some distinctions we outline some of them below:

• The term AQC algorithm refers to a universal AQC model algorithm. AQC algorithms can be programmed to solve any Turing-computable problem. They are able to simulate quantum algorithms in the gate model with a polynomial increase in computation time.

• A QA algorithm is typically designed to solve an NP-hard optimization problem. This necessitates a constraint on the final Hamiltonian H_F in order for it to represent a classical objective function. On the other hand, certain AQC model properties are relaxed. For instance, we do not assume that the entire computation occurs in the ground state.

Similar to simulated annealing, the process of quantum annealing can be thought of as a type of heuristic search that moves across a solution landscape looking for low-lying areas. However, QA uses a transverse field coefficient (also known as the tunnelling coefficient) to control the traversability of the landscape of the solution instead of using a temperature parameter T (as in simulated annealing) to control the probability of taking an uphill step at an iteration. Therefore, we define Γ, a scaling parameter that serves a similar purpose as T. Γ, like T, begins with a high value and is gradually decremented over time according to a predetermined schedule.

The objective function $f(x)$ on the vector of bits x is represented by a final Hamiltonian H_F, which is an $N \times N$ Hermitian matrix with costs $f(x)$ on the matrix diagonal and zeros everywhere else. The goal is to construct a Hamiltonian in which each eigenstate that corresponds to an assignment to x has an eigenvalue that corresponds to $f(x)$. QA introduces a disordering Hamiltonian H_D (also called the transverse field Hamiltonian), which does not commute with H_F.

H_D is scaled by $\Gamma(t)$ and H_F is scaled by $\Lambda(t)$. $\Gamma(t)$ is initialized to 1 and gradually reduced to 0, while $\Lambda(t)$ is initialized to 0 and gradually increased to 1. This results in the time-dependent Hamiltonian:

$$H(t) = \Gamma(t) H_I + \Lambda(t) H_F \tag{2.21}$$

Quantum annealing on this system is achieved by the gradual evolution of the Hamiltonian system [52]. If the annealing is performed slowly enough, the system stays in the ground state of $H(t)$ for all times, t, ending up at the end of the annealing at the ground state of H_F.

The added term H_D introduces kinetic energy to the process of annealing that takes the form of quantum fluctuations in the space of the solution. Decreasing $\Gamma(t)$, and increasing $\Lambda(t)$, transforms the system closer to H_F while mitigating the quantum fluctuations.

There is an apparent similarity between equation (2.21) and the AQC algorithm introduced in Section 2.3.3.3. The Hamiltonian of both problems serves the same purpose. The disorder introduced by the orthogonal Hamiltonian H_D of QA allows the heuristic search to escape local minima. This is similar to the initial Hamiltonian H_I introduced by AQC to ensure that initial superposition states are equiprobable. The scaling parameters, $\Gamma(t)$ and $\Lambda(t)$, generate a particular type of adiabatic path.

However, there are some differences in the standard approaches to analysis in the AQC and the QA research domains. AQC analysis derives a bound on the probability of the system finishing in the ground state if it starts in the ground state. QA analysis derives the convergence probability to a solution within ε of the optimal solution when starting the algorithm from an arbitrary state.

2.3.3.4.1 Ising Model The Ising model is a mathematical model that is used to study phase transitions and other properties of physical systems as they evolve over time. The Ising is a generalization of the primary problem that D-Wave's quantum annealers are designed to solve. The problem involves a set of n particles assembled on the vertices of a graph $G = (V, E)$, which is commonly assumed to be a d-dimensional grid. Each particle can exist in one of two states known as spins, which are represented by $+1$ and -1. A spin configuration $s = s1 \ldots s_n$ is a set of spin values assigned to particles.

The motivation application is to characterize different properties for certain configuration given that external forces h_i are applied to each individual particle an interaction forces J_{ij} are applied between adjacent particles ($J_{ij} = 0$ for non-adjacent particles). The energy of a certain configuration is defined as:

$$H(s) = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j \tag{2.22}$$

Many statistical mechanics applications require to find the ground state of such a system. That is the spin configuration that minimizes $H(s)$. The quantum computational systems, such as the ones by D-Wave, use quantum annealing to locate the ground-state of an artificial Ising system [52]. The Ising is a generalization of the primary problem that D-Wave's quantum annealers are designed to solve.

An Ising Hamiltonian describes the behaviour of such a system as:

$$H_p = \sum_{i=1}^{N} h_i \sigma_i^z + \sum_{i,j=1}^{N} J_{ij} \sigma_i^z \sigma_j^z \tag{2.23}$$

where h_i is the energy bias for spin i, J_{ij} is the coupling energy between spins i and j, σ_i^z is the Pauli spin matrix (as described in Section (2.3.3.1.2)), and N is the number of qubits.

2.3.3.4.2 QUBO Model The Hamiltonian in (2.22) can be rewritten in the form of a QUBO problem by encoding the spin variable s_i (that represents the two values $+1$ and -1) to be a binary variable x_i (that represents the two values 1 and 0) [112]. This conversion necessitates the adjustment of the coefficients h_i and j_{ij} to a_i and b_{ij}, respectively.

As used in the rest of this thesis, the objective function is expressed in QUBO form in scalar notation, and is defined as follows:

$$C(x) = \sum_{i} a_i x_i + \sum_{i<j} b_{i,j} x_i x_j \tag{2.24}$$

where $x \in \{0;1\}^n$ is a vector of binary variables and $\{a_i; b_{i;j}\}$ are real coefficients.

Many problems can be formulated to take advantage of quantum annealing, and it is advantageous because it converges faster than other techniques to an optimum solution [53].

2.3.3.4.3 Quantum Annealing and Simulated Annealing Quantum annealing can be compared to simulated annealing by identifying that the temperature parameter T in simulated annealing performs a similar role to quantum tunnelling in quantum annealing. The temperature in simulated annealing defines the probability of moving from a single current state to a higher energy state to escape local minima. The assumed advantage of quantum annealing over simulated annealing is that tunnelling allows the system to directly pass through high-energy barriers without having

to climb over them. Figure 2.20 shows an analogy between Quantum annealing and simulated annealing.

The high value of $\Gamma(t)$ enables the system to escape a local minimum by tunnelling through the landscape hills instead of climbing these hills as in simulated annealing. This allows the quantum annealing algorithm to traverse faster and further in the landscape of the problem at the early stage of the process.

Indeed, when the algorithm is implemented on a quantum platform, it is not in one state at a point of time t, but in a superposition of states with probabilities determined by $H(t)$.

Analytical and numerical evidence indicates that quantum annealing can outperform simulated annealing [38]. Therefore, quantum annealing is an excellent potential solver for the InSAR phase unwrapping problem.

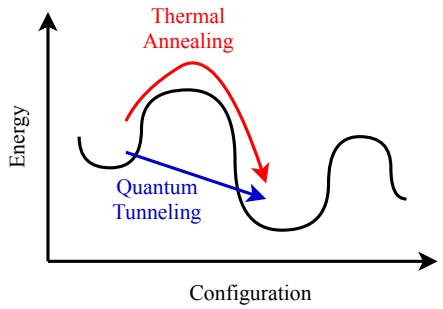


Figure 2.20: Quantum annealing versus simulated annealing

2.3.3.5 Universal Quantum Gate Model vs. Quantum Annealer

Universal quantum computers can be utilized in many more problems than a quantum annealer. However, they come with their challenges and a different design than quantum annealers. Quantum annealers' main advantage over the quantum gate model is the ability of quantum annealers to comprise more qubits. For instance, IBM's current largest quantum computer contains 65 qubits, while D-Wave annealer contains over 5000 qubits. This relatively large number of qubits is driven by the quantum annealer robustness against decoherence (decoherence is the tendency of qubits to lose the entangled states).

Quantum devices suffer from noise. Error correction codes promise to allow quantum computer systems to operate for sufficiently long periods of time to reach the correct solution while correcting errors that arise during the computation. However,

quantum error correcting codes are expensive in that they require several (10s) of physical qubits per logical qubit. Presently, the availability of noisy intermediate scale quantum devices (NISQ) is stimulating efforts to find applications that are less sensitive to noise [82]. Quantum annealers, like D-Wave, are becoming an excellent baseline for concept proof.

2.3.3.6 Phase Unwrapping Using Quantum Annealer

Analytical and numerical evidence indicates that quantum annealing outperforms simulated annealing [38]. Quantum annealing is a potential solver for InSAR phase unwrapping.

The phase unwrapping problem is a quadratic unconstrained problem by default. It could be mapped to a QUBO problem by simply encoding each variable into a vector of binary variable:

$$E = \sum_{(s,t)\in A} W_{st} \left(k_t - k_s - a_{st}\right)^2 + \sum_{s\in A} \omega_s \left(k_s - a_s\right)^2$$

$$E = \sum_{(s,t)\in A} W_{st} \left(\sum_i b_i x_{i,t} - \sum_i b_i x_{i,s} - a_{st}\right)^2 + \sum_{s\in A} \omega_s \left(\sum_i b_i x_{i,s} - a_s\right)^2$$

where $x_s \in \{0;1\}^n$ is a vector of binary variables that encode the variable k_s and b_i are the weighting coefficients.

Quantum annealing is expected to speed up the unwrapping process significantly. However, with the limited number of qubits in the current quantum computers, it will be hard to fit the minimization problem of an InSAR image; InSAR images could be up to 20k×30k pixels. To solve the InSAR phase unwrapping problem using a quantum annealing machine, we propose the Phase Unwrapping Decomposition approach. In Phase Unwrapping Decomposition, we divide the image into smaller sub-images that easier to unwrapped and use different methodologies to approach the global solution based on the sub-images solution.

2.4 Chapter Summary

In this chapter, we presented background materials on SAR, Neural Networks, and quantum computing. We presented the related concepts of SAR operation and In-SAR processing stages, emphasizing co-registration and phase unwrapping stages. We introduced the basic concepts of machine learning and CNNs. Moreover, we summarized the related work in CNNs. Finally, we introduced the use of quantum computing for optimization. First, we introduced the concept of optimization. Then, we summarized how quantum methods are used for optimization.

Chapter 3

Scale Invariant Super-Resolutions Methods With Application to InSAR Images

Super-resolution (SR) is the process of generating high-resolution (HR) images from a low-resolution (LR) one. SR is used in SAR images co-registration, as explained in more detail in the next chapter. The process of SR strongly influences the co-registration quality and could potentially be used in enhancing later stages of SAR images processing. We study the employment of learning-based SR for SAR applications. This chapter proposes a new learning-based SR model that targets SAR applications, and similar applications. The next chapter utilizes the proposed model in SAR co-registration and InSAR interferogram generation and assesses the results.

3.1 Introduction

SR, is used in digital image processing to reconstruct high-frequency data from low-frequency ones under the assumption that much of the high-frequency data recurs, redundantly, across different scales of an image [79]. The conventional approach employs interpolation (e.g., Bicubic interpolation [55]). This, however, does not always create high-frequency components, resulting in blurry images.

Previous research has introduced methods to address the problem. These methods are classified as Single Image Super-Resolution (SISR) or as Multi Image Super-Resolution (MISR) [84]. While SISR uses a single image to produce a HR image,

MISR, uses several LR images. In this work, we focus on the SISR approach. There are two classes of SISR algorithms: reconstruction-based algorithms, and learning-based ones, aiming to find mappings between LR and HR image pairs in a dataset [79].

In learning-based SR algorithms, ANN are used. CNN showed significant success in learning-based SR [8]. This is achieved by training the network using HR and LR image pairs and use this network later to create new HR images from LR ones.

However, in some applications (e.g., satellite or medical imaging), HR ground truth images are missing. Additionally, training the model using HR images, is computationally expensive. We introduce ***S****cale* ***Inv****ariant Super-Resolution* or ***SINV*** to address these concerns. We postulate that obtaining HR images from lower resolution ones is invariant across scales. Thus a model that is trained using lower resolution images can also generate higher resolution images at higher scales.

One uses sets of pairs of identical images at different resolutions to train the SR model. Traditionally, the SR model is trained on sets of MR-HR images and applied on MR images to obtain HR images. We refer to this as the *direct SR*. We train our model on sets of LR-MR images, but apply the trained model on MR images to obtain HR images. We refer to our method as the *SINV* method.

In this chapter we study the effectiveness of the proposed SINV method, showing that it outperforms traditional methods (i.e., Bicubic interpolation), and apply it to the InSAR problem. We evaluated SINV using different datasets, and with different upscaling factors[1] and showed that it outperforms conventional approaches. We have applied SINV to processing InSAR images.

3.2 Related Work

SISR methods [84] are either learning-based or reconstruction-based [79]. As our work focuses on learning-based SISR, we briefly review prior work in this area.

Candocia et al. [12] used local correlations to obtain a number of mapping kernels. The SR image is generated by convolving the LR image with the chosen kernels. Zhang et al. [114] solved the problem by considering it as an ill-posed inverse problem and used a Hopfield network to minimize an evaluation function calculated by an observation model based on the physical image acquisition process. Huang et al. [47] suggested an SR approach based on optimal recovery theory. Shi et al. [42] introduced

[1]The upscaling factor is the factor by which the image resolution is increased.

periodic shuffling layers that map the LR image directly to a HR image.

Ribeiro et al. [86] trained CNNs using HR-LR images. The LR images, of the same size as the HR ones, were obtained by downscaling the HR image and then re-upscaling it, using Bicubic interpolation. Higher scale-factor images are obtained by passing the image through the network repeatedly. Jia et al. [51] proposed a multi-scale CNN employing two parallel paths. The upper path deploys a larger number of layers resulting in a larger receptive field as compared to the lower path.

Contrary to our proposed *SINV* method, all SR methods reviewed use HR (ground truth) images during training. We postulate that the upscaling process is invariant of the scale, and hence a network trained to upscale lower resolution images can be used for the same purpose at higher resolutions.

3.3 Methodology

SR estimates a HR image, of similar quality as the original (ground truth) HR one. SR requires HR-LR pairs of images to train the upscaller.

An SR method uses a LR image obtained through downsampling the original HR image, to produce its estimated HR image. The process of generating LR images from HR images depends on the downsampling models that reflect the nature of the imaging process. Ideally, one needs to have both the LR and the HR images obtained directly by the imaging process. Such image pairs then incorporate the downsampling function involved. However, in most situations, such naturally obtained images do not exist, hence an approximation of the downsampling function as suggested in [49] is used.

In this work, we address the problem of estimating HR images using learning-based SR for applications where HR ground truth images are not available. The goal of SINV is not to compete with earlier proposed learning-based SR methods. The proposed methodology applies to most of the learning-based SR contributions in the literature. However, our architecture is mostly influenced by the work presented in [91]. The methodology we are proposing assumes input images given at specific medium-resolution (MR). We use the MR images and Bicubic downsampling to generate LR images, and train our CNN using the thus generated LR/MR images. The process of downsampling could follow any model. In this work, the LR images are generated from the MR images using Bicubic downsampling. The thus generated LR/MR image pairs are used to train a CNN.

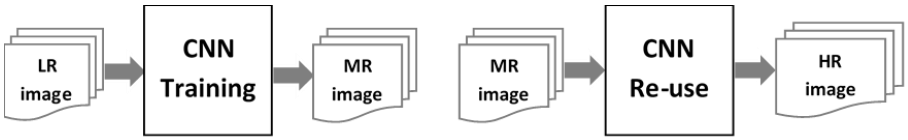


Figure 3.1: The proposed methodology to train the CNN and generate HR images without HR ground truth. The diagram on the left shows the process of training the CNN using the available images at MR and the derived images at LR. The diagram on the right shows the process of using the trained CNN to generate images at HR from images at medium, (MR) resolution

We postulate that the downsampling process is not affected by the scale of the image. Hence, if the inverse downsampling is to be determined at one scale, the same function could be employed at a different scale. We denote this postulate as *scale invariance*, and through this work, we shall show that it indeed holds. Figure 3.1 shows the proposed methodology.

Since the scales of the images vary, the *periodic shuffling layer* is used to scale the images accordingly (Section 3.3.1).

3.3.1 Periodic Shuffling Layer

To up-scale the LR image, we follow Shi et. al [91] where all the layers of the CNN process the LR image and only the last layer preforms the up-scaling as follows:

$$I_{SR} = PS\left(W * I_{LR} + b\right)$$

where I_{SR} is the SR image, PS is a periodic shuffling, I_{LR} is the LR image, W is the network weights, and b is the bias. The periodic shuffling operator rearranges the elements of an $M \times N \times C \cdot r^2$ tensor to a tensor of shape $r \cdot M \times r \cdot N \times C$:

$$PS\left(I\right)_{x,y,c} = I_{\lfloor \frac{x}{r} \rfloor, \lfloor \frac{y}{r} \rfloor, c \cdot r \cdot (y\%r) + c \cdot (x\%r)}$$

where I is the input image, x and y are the coordinates of a pixel, c is the number of channels, r is the upscaling factor, and $\%$ is the modulo operation.

3.3.2 Network Architecture and Training

Several CNN architectures were considered during a design space exploration phase. The proposed network consists of three stages. First, a multi-layer stage of convo-

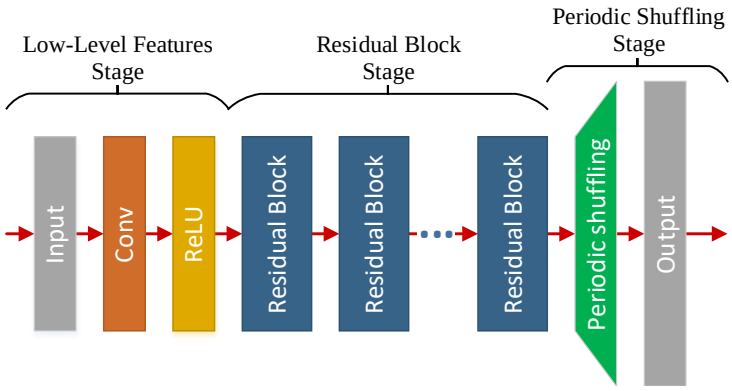


Figure 3.2: The architecture of the proposed CNN. There are three stages of the CNN; low-level features, residual blocks, and periodic shuffling stage

lutional and ReLU layers extract the low-level features. Second, a stage of multiple residual block layers, deep enough to learn features that enable the network to be reused at different resolutions. Each residual block consists of a convolutional layer, ReLU activation function, another convolutional layer, bypass link from the input of the block, and finally another ReLU activation function. We varied the number of the residual block during a design space exploration phase to achieve the best performance. Finally, a periodic shuffling layer stage converts the depth of the last layer into the width and height of the output HR image resolution. The last residual layer has $r^2 \cdot c$ feature maps, where r is the upscaling factor and c is the number of the HR image channels. These $r^2 \cdot c$ feature maps are then shuffled to construct the HR image as per the discussion in Section 3.3.1. Figure 3.2 shows the architecture of the proposed CNN.

We trained networks of different number of layers, filter sizes, and filters per layer. For each configuration, the network is trained for 5 epochs. After 5 epochs the network converges. The results are shown in Table 3.1.

3.3.3 Upscaling Network

During training, we expose the network to LR input images of size $m \times n \times c_{in}$, where m is the width, n is the height, and c_{in} is the depth or the number of the channels. For all the convolutional layers, the width and the height are kept the same. In the periodic shuffling, the last convolutional layer is reshaped by reducing the depth c, to match the number of the output channels, c_{out}, while the width and the height are extended. Since, $c = r^2 \cdot c_{out}$, the width and the height are extended to be $r \cdot m \times r \cdot n$.

For the HR image production, we apply MR input images of size $M \times N \times c_{in}$. The trained filters are now convolved with larger tensors of size $M \times N \times c$ instead of size $m \times n \times c$. The periodic shuffling rearranges the output tensor to have c_{out} channels instead of $r^2 \cdot c_{out}$ channels. This will be reflected on the width and height of the tensors and transform them from $M \times N$ to $r \cdot M \times r \cdot N$.

3.3.4 Network Testing

Networks trained using LR/MR images were used to generate HR images. Performance was measured by comparing the generated HR images and the ground truth HR images as well as Bicubic- and Direct SR- upscaled MR images. Two metrics are used: Structural Similarity Index (SSIM) [103] and Root Mean Square Error (RMSE).

3.4 Experimentation and Results

We studied SINV under different upscaling factors, datasets, and network configurations. We experimented with upscaling factors ranging from 2 to 4, varied the filter sizes from 3×3 to 9×9, the number of layers of the CNN from 4 to 24, and experimented with different network structures. For all the networks, the last layer is a periodic shuffling layer. The size of the images were typically 32×32 or smaller for LR, 64×64 for MR images and 128×128 or larger for HR images. We tested 12 different configurations by varying the three parameters of the network mentioned above. Assuming the parameters to be independent, we varied one of the parameters, obtained the best performing value, and used it during the variation of the other parameters.

3.4.1 Datasets

Even though SINV was developed mainly to target SAR application, in the early development of SINV we experimented using photo image. This because the photo images are easier to interpret and more accessible compared to SAR images. However, we used a single large SAR image in Section 3.4.6 to provide a preliminary evaluation of the applicability of the approach to SAR images.

We used the following benchmark datasets: "MIRFLICKR" (Mflickr), CelebFaces Attributes (CelebA), and DIVerse 2K high-quality images (DIV2K) [36]. Each dataset

is divided into a testing set and a training set. For each testing and training set, three subsets are created; HR, MR, and LR. The original dataset represents HR. MR images are generated from the HR images using Bicubic downsampling with a scaling factor r. LR images are generated from the MR images using Bicubic downsampling function with the identical scaling factor. The scaling factors we used in this work are $r = 2$ and $r = 4$.

The results are summarized in Table 3.1. Asides from the parameter we are varying, the default parameters are 16 residual layers, 3×3 filter size and 64 filters/layer. In the subsequent sections, we shall analyze these results in more detail.

Table 3.1: The results of varying the network parameters for an upscaling of 2 ($r = 2$) and upscaling factor of 4 ($r = 4$). The highlighted (in yellow) entries depict the configurations where the performance saturates

a) Varying the number of residual layers.

		Dataset	# residual layers (64 filters/layer, 3×3 filter size)						Bi-cubic
			4	8	12	16	20	24	
$r = 2$	SSIM	CelebA	88.1	88.3	88.4	88.5	88.5	88.5	86
		Mflickr	86.3	86.9	87.2	87.8	87.3	87.4	84
		DIV2K	91.8	91.9	92.4	92.5	92.6	92.4	90.1
	RMSE	CelebA	1.18	1.16	1.15	1.15	1.15	1.15	1.26
		Mflickr	1.01	0.99	0.99	0.95	0.99	0.97	0.95
		DIV2K	0.85	0.83	0.8	0.8	0.8	0.8	0.79
$r = 4$	SSIM	CelebA	73.8	74.9	75.2	75.3	75.2	75.2	72.8
		Mflickr	71	71.6	72	73.6	72.3	71.7	71.9
		DIV2K	76.5	77.1	77.5	77.5	77.2	77.2	76.3
	RMSE	CelebA	1.51	1.46	1.45	1.45	1.45	1.45	1.48
		Mflickr	1.35	1.31	1.28	1.23	1.28	1.29	1.11
		DIV2K	1.24	1.2	1.21	1.2	1.21	1.22	7.05

b) Varying the number of filters per layer.

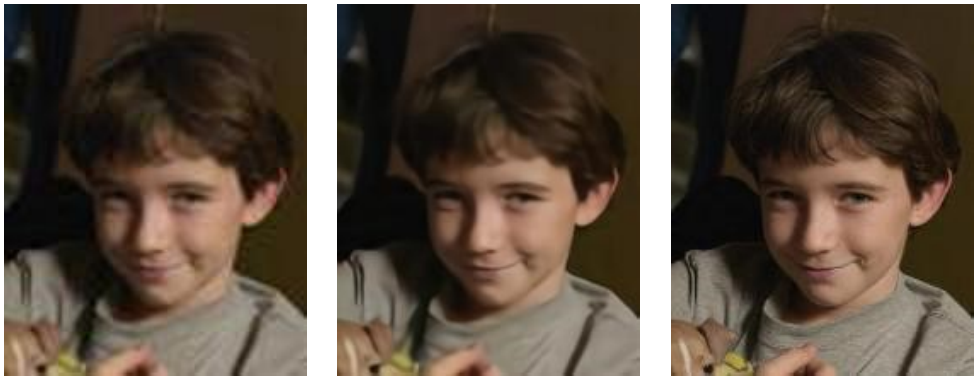
		Dataset	# filters/layer (16 residual layers, 3×3 filter size)				Bi-cubic
			32	64	128	256	
$r = 2$	SSIM	CelebA	88.2	88.5	88.7	88.9	86
		Mflickr	86.1	87.8	87.7	87.7	84
		DIV2K	91.6	92.5	92.8	92.9	90.1
	RMSE	CelebA	1.17	1.15	1.14	1.13	1.26
		Mflickr	1.02	0.95	0.96	0.97	0.95
		DIV2K	0.86	0.8	0.78	0.76	0.79
$r = 4$	SSIM	CelebA	74.6	75.3	76	76	72.8
		Mflickr	70.7	73.6	72.9	74	71.9
		DIV2K	76.2	77.5	78.2	78.7	76.3
	RMSE	CelebA	1.48	1.45	1.42	1.41	1.48
		Mflickr	1.38	1.23	1.25	1.22	1.11
		DIV2K	1.29	1.2	1.18	1.14	7.05

c) Varying the filter size.

		Dataset	Filter size (16 residual layers, 64 filters/layer)				Bi-cubic
			3×3	5×5	7×7	9×9	
$r = 2$	SSIM	CelebA	88.5	88.4	88.3	88.3	86
		Mflickr	87.8	86.9	86.7	86.1	84
		DIV2K	92.5	92.2	91.9	91.6	90.1
	RMSE	CelebA	1.15	1.16	1.16	1.16	1.26
		Mflickr	0.95	0.99	1	1.02	0.95
		DIV2K	0.8	0.83	0.86	0.85	0.79
$r = 4$	SSIM	CelebA	75.3	74.5	72.9	63.8	72.8
		Mflickr	73.6	71.2	70	68.3	71.9
		DIV2K	77.5	76.8	75.7	74	76.3
	RMSE	CelebA	1.45	1.49	1.57	1.85	1.48
		Mflickr	1.23	1.35	1.39	1.59	1.11
		DIV2K	1.2	1.25	1.43	1.64	7.05

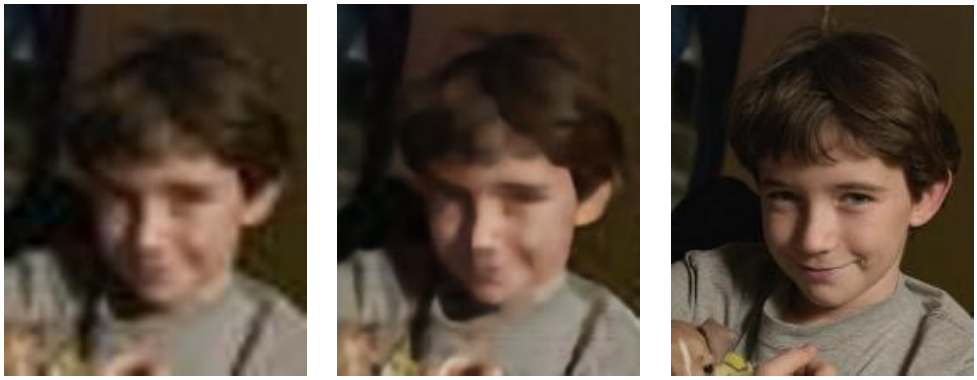
3.4.2 Varying Different Network Parameters

Table 3.1 shows the performance in terms of SSIM and RMSE as a function of network parameters. Performance improves with the number of layers. The improvement saturates after 16 layers. Increasing filter size beyond 3×3 does not improve performance. Increasing the number of filters per layers enhances the performance, saturating after 128. The best performance is obtained for 16 residual layers, 64 filters/layer, and 3×3 filter size.



(a) Bicubic interpolation (b) SINV (c) Ground truth

Samples for $r = 2$.



(d) Bicubic interpolation (e) SINV (f) Ground truth

Samples for $r = 4$.

Figure 3.3: Samples from CelebA dataset

3.4.3 Comparing SINV With Bicubic Interpolation

Table 3.1 and Figure 3.3 summarize the result obtained. As can be seen, from Table 3.1, SINV outperforms the Bicubic interpolation while visually from Figure 3.3, SINV results in a sharper image with more details visible as compared to Bicubic.

3.4.4 Comparing SINV With the Direct SR

In SINV, we train the CNN using images in LR space, i.e., using LR-MR images pairs. However, the direct method of training the CNN for SR is to use MR-HR images pairs. Table 3.2 show the performance of SINV and the direct SR for different datasets and different scaling. SINV achieves performance that is very close to that of the direct SR, and in some cases it outperforms it. As shown, SINV at worst, has a SSIM that is 4.08% lower that that of the direct method (CelebA $r = 4$). while It

Table 3.2: Comparing the performance of the direct SR and the scale-invariant methods for different datasets and different upscaling factors

	SSIM				RMSE			
	SINV		Direct SR		SINV		Direct SR	
	$r=2$	$r=4$	$r=2$	$r=4$	$r=2$	$r=4$	$r=2$	$r=4$
CelebA	88.4	75.3	89.1	79.4	1.15	1.45	1.12	1.29
Mflickr	87.8	**73.6**	88.6	71.3	0.95	**1.23**	0.93	1.28
DIV2K	**92.5**	**77.5**	91.3	76.6	0.8	**1.21**	0.78	1.22

outperforms the direct method for Mflicker ($r=2$) and DIV2K ($r=2$, $r=4$).

3.4.5 Timing Results

By using SINV, we achieve faster training time as we use the MR-LR image pairs instead of HR-MR pairs during training.

The forward calculations, dependent on the image size, scale as r^2, while the backpropagation step is independent of the image size. For large images, the forward step dominates, while for smaller images, the backpropagation step does. Considering the fact that each image dimension increased by a factor of r so the total pixel count multiplied by r^2. This will cause all forward calculations, which depend on the input size, to increase with the same ratio. However, this is the upper bound since the backpropagation step is image size invariant relying only the network structure. For large images, the image dependent forward step dominates and we achieve close to the r^2 improvement while for smaller images the backpropagation step dominates and the improvement is inferior to the expected r^2. As shown in Table 3.3 for $r=2$ the images used are larger, 32×32, while for the $r=4$ case the images are smaller, 11×11.

3.4.6 Radar Imaging Application

In this section, we examine the applicability of SINV to SAR images using a small set of SAR images. The next chapter provides more comprehensive study about applying SINV for SAR/InSAR images.

Table 3.3: SINV performance improvement over direct SR method for the scale factor of 2 and 4. The batch size is equal for both configurations and its equal to 16

	$r = 2$		$r = 4$	
# Residual Layers	12	24	12	24
LR/MR Training Time (sec)	1083.5	2052.8	80.7	145.1
MR/HR Training Time (sec)	4306.9	8056.6	677.1	1254
Improvement (Speedup)	398%	393%	834%	865%

3.4.6.1 Our Approach

For InSAR applications, the conventional way of generating SR images is to use Sinc interpolation [68]. Using SR via CNN is expected to outperform the conventional approach. In our approach, we upscale the SAR images using *SINV* to attain a better approximation as compared to Sinc. Since SAR images include both phase and magnitude, pixels are represented as complex numbers. Super-resolution is applied separately to the real and imaginary parts.

3.4.6.2 Evaluation

Coherence is computed between the generated image and the ground truth. Coherence is computed within a moving window covering the SAR image. The mean of the resulting coherence values is used for evaluation; the higher the mean, the better the generated image.

3.4.6.3 Data

The SAR image (ERS-1/2 single look complex image) for the city of Vancouver was used.

3.4.6.4 Results

Table 3.4 summarizes the mean of the coherence for SINV SR and Sinc interpolation for Vancouver area. For upscaling factor of 2 and 4, the mean of the coherence for SINV SR is higher than Sinc interpolation.

For upscaling factor of 2 and 4, the mean of the coherence difference is positive. Meaning the SINV SR has higher coherence compared to Sinc interpolation.

Table 3.4: The mean of the coherence for SINV SR and Sinc interpolation for upscaling factor of 2 and 4

	SINV	Sinc
Upscaling Factor $r = 2$	87.32	84.10
Upscaling Factor $r = 4$	81.74	81.1

3.5 Discussion

This work proposes *SINV*, an SR method where the upscaling system is trained using lower resolution images, and applied to higher resolution environments successfully. Our results show that *SINV* works well for upscaling factors below 4. Beyond this, texture detail is curtailed in the reconstructed SR images. Ledig [65] used a GAN network to provide artificial texture details; however, the resulting images, although visually appealing, had large errors compared to the ground truth and are not applicable in applications (e.g., SAR) where non-artificial details are required.

Our results show that *SINV* provides improvements over Bicubic interpolation in most cases, ranging from 3.7% SSIM improvement for upscaling factor of 2 to 3.1% improvement for upscaling factor of 4. *SINV* is comparable to direct SR, and in some cases it outperforms it.

3.6 Chapter Summary

We proposed SINV as a Super-Resolution solution in the absence of ground truth. This helps with some applications where ground truth is not available or when the processing power in the training phase is limited. Performance evaluation over several datasets shows the proposed method provides better numerical and visual results compared to Bicubic interpolation.

Chapter 4

Improving InSAR Image Quality and Co-Registration Through CNN-Based Super-Resolution

InSAR is a measuring technology that uses the phase information contained in the images of the SAR. InSAR has been recognized as a potential method for digital elevation models (DEMs) generation and ground surface deformation measurement. Nonetheless, the quality of InSAR data is influenced by many critical factors. Including image co-registration, interferogram generation, phase unwrapping and geocoding. Image co-registration aims to align two or more images so that the same pixel in each image corresponds to the same point of the target scene. This study proposes a new algorithm for improving image co-registration and interferogram generation of SAR using learning-based images SR. We show that our approach improves the conventional approaches.

4.1 Introduction

SAR is a microwave imaging system that uses a coherent side-looking radar [17]; the antenna being "flown" over the target area. SAR systems produce remote sensing imagery of high resolution. The phase and the amplitude of the received signal carries detailed information of the scatterer (i.e., terrain). Signal processing algorithms are used to combine several received signals (SAR images) to extract pertinent information of the scattering terrain. Including elevation as well as information about the

nature of the terrain (e.g., water vs. vegetation).

The InSAR uses two or more SAR images of the same target to extract any changes (including elevation) of the target. Since the images are acquired at different times and from different antenna position, they are typically process as follows [116]:

1. SAR image acquisition
2. Co-registration of two or more SAR images
3. Interferogram generation
4. Phase unwrapping
5. Geocoding of digital elevation models

The quality of products resulting from InSAR images is influenced by various critical factors including co-registration. Co-registration is the alignment of two SAR images taken by the same antenna, but a different repeat passes (usually several days later in a slightly different imaging position). This alignment is essential to ensure the same pixel in the two SAR images reflects the same ground target. InSAR technology benefits from enhancing the accuracy of co-registration [116]. For more details about the co-registration process, see Section 2.1.6.

Machine learning models, specifically CNNs, have achieved great success in learning-based SR algorithms. However, since HR ground truth images are not available, or hard to access, to train super-resolution CNNs for SAR applications, we have developed SINV [39] that achieves excellent SR results in the absence of HR ground truth. Chapter 3 discusses SINV in more details.

In this work, we employ SINV [39] in the interferometric SAR processing to improve the accuracy of the obtained results. In our approach, we use SINV to generate super-resolution images which we then use to achieve superior co-registration and interferogram generation. Many papers in the literature proposed to improve the co-registration [33, 72, 80, 99, 108, 113]. However, none of these papers employed super-resolution nor machine learning to address this problem. Hence, our work, to the best of our knowledge, is the first to pursue this approach.

4.1.1 SAR Interferometry Generation

InSAR is used in many applications, including measuring tectonic deformation, glacial motion, and infrastructure monitoring. However, one of the most prevalent applica-

tions for InSAR is mapping the deformation of the ground. A target area can be imaged several times by the same satellite. However, images obtained at different times have been imaged from different satellite perturbed orbital positions (due to natural drift in orbits.) Co-registration is used to find correspondences of pixels between two SAR images. Then the interferogram is generated using two SAR images. These two SAR images must preserve the phase. The two SAR images are conventionally referred to as master and slave.

The InSAR interferogram is generated by cross-multiplying the master SAR image with the complex conjugate of the co-registered slave image, resampled to the master's geometry, pixel by pixel [7, 27, 76]. Thus, the amplitude of the interferogram is the amplitude of the first image multiplied by the amplitude of the second image, while its phase is the phase difference between the two images. Interforgram generation is discussed in more detail in Chapter 2.

4.1.2 Co-registration

SAR interferometry generation requires a pixel-to-pixel match between the features of two SAR images. The co-registration is the alignment of SAR images captured from the same antenna, at different acquisitions in time. This alignment is necessary to ensure the same pixel in the master and the slave image reflects the same ground target. Therefore, co-registration is a fundamental step for the precise determination of interferometric phase difference and for reducing the noise [68]. The alignment between the master and the slave is performed on a pixel by pixel basis. The alignment accuracy should be at the order of at least one to the tenth of the resolution [25]. Hence, a subpixel level co-registration must be performed.

Since the master and the slave images capture the same terrain area but from two different antenna positions, there is a deformation due to dilation and different orientation of the coordinate systems of the two images. The transformation that describes the deformation between the master and the slave can be well approximated by the following polynomial [69]:

$$\begin{cases} r_s = a \cdot r_M^2 + b \cdot r_M + c \cdot a_M + d \\ a_s = e \cdot r_M^2 + f \cdot r_M + g \cdot a_M + h \end{cases} \tag{4.1}$$

where: (a_M, r_M) are the azimuth and range coordinates of the master image and (a_S, r_S) are the azimuth and range coordinates of the slave image corresponding. The

most common approach for determining the transformation coefficients is to calculate the amplitude cross-correlation between the master and the slave images [67]. After estimating the transformation function between the master and the slave, the slave image is then resampled to ensure that the final interferogram is in the same master image reference. Methods of interpolation are used to approximate the subpixel values. Usually, Sinc interpolation gives the best performance [35,68].

4.1.3 SINV SR

SINV consists of three stages. First, a low-level features extraction stage of a convolutional layer followed by a ReLU layer. Second, multiple residual block layers stage. This stage is typically 16 residual blocks, each block is two convolutional layers with a residual connection. Finally, a periodic shuffling layer that converts the last layer depth into the output HR image width and height.

We trained a network of 16 layers, 3×3 filter sizes, and 128 filters per layer. The network is trained for 5 epochs. After 5 epochs the network converges. SINV is trained using 10 SAR images and tested using 5 SAR images. The sizes of theses images are summarized in table 4.1. During training, the images are partitioned into non-overlapping sub-images of 256×256 pixels. The training on a GeForce GTX TITAN GPU requires about 77 hours of a 10 images training set. Further experimentation is required to fully qualify the time complexity of the SINV training process.

Table 4.1: The sizes of the training and testing data images

Training data				**Testing data size**	
Site #	**Image size**	**Site #**	**Image size**	**Site #**	**Image size**
1	11500×5440	6	10020×20100	test1	19484×32435
2	4240×4900	7	16734×26920	test2	10600×9520
3	10470×12300	8	3970×2200	test3	14310×20810
4	8910×16400	9	11300×10490	test4	6070×9210
5	6140×5140	10	6840×9710	test5	11040×9290

Figure 4.1: SINV upscalling methodology

4.2 Methodology

We have used a super-resolving system to increase the accuracy of co-registration. Image resolution is increased and then co-registration is performed on the enhanced resolution images. Since SAR images include both phase and magnitude, pixels are represented as complex numbers. Super-resolution is applied separately to the real and imaginary parts.

4.2.1 Enhancing Co-registration

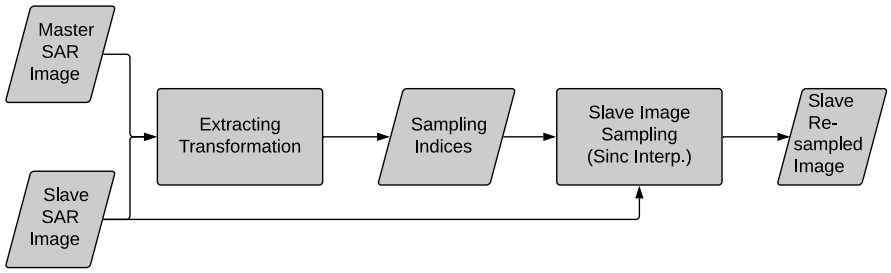
For co-registration enhancement, we upscale the SAR image using SINV CNN to obtain a better approximation of the subpixel values compared to the direct Sinc interpolation. Then the image is resampled using the polynomial model defined by (4.1) generated earlier through co-registration. Typically, images are upscaled with a factor of 2. Then we apply Sinc interpolation to generate continuous values for the subpixels. Figure 4.2 shows a diagram for the process.

4.2.2 Enhancing InSAR Images

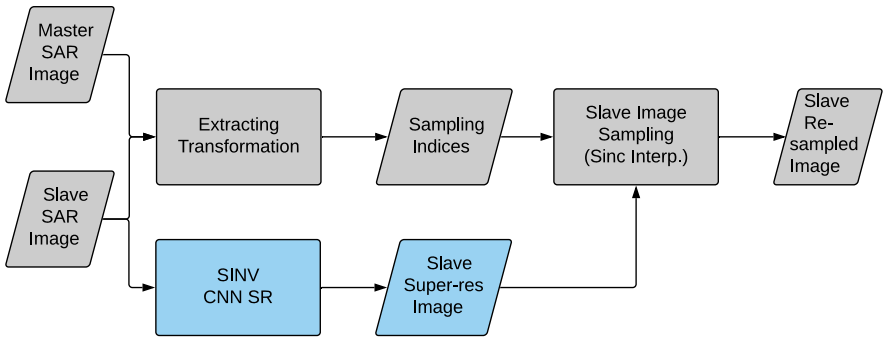
For enhancing the InSAR images, SINV is applied directly to the InSAR images to generate more accurate images that can be used later to improve processes, like phase-unwrapping for instance. Here, the complex InSAR image is split into real and imaginary images. The SRs for both images are then generated separately. Finally, the real SR and the imaginary SR images are combined into a single complex SR InSAR image. The process is illustrated in Figure 4.3. The resulting images are compared with images enlarged using Sinc interpolation.

4.3 Experimental Results

In this section, we present the experiments in applying SR in co-registration and interferometric SAR generation.



(a) Sinc co-registration.



(b) SINV co-registration.

Figure 4.2: Proposed co-registration diagram. On the top, the conventional approach for co-registration using only Sinc interpolation. On the bottom, our approach of adding SINV to give a better approximation for the subpixels

4.3.1 Evaluation

Coherence, as discussed in Section 2.1.5, is the metric of choice to compare two InSAR images [68]. Given two InSAR images, A and B, their per-pixel coherence map is computed using a moving window method. The coherence maps are subtracted and the difference $coh(A) - coh(B)$ is formed. The histogram of these differences is used to evaluate how similar in quality the two images are and which is of higher quality. Thus if the mean is positive and the standard-deviation (Std) is relatively small, image A is considered better. We have used this approach to compare our super-resolution obtained co-registered images to the ones obtained through simply Sinc interpolation. Our super-resolution co-registered images correspond to the A images and are compared to the non-enhanced B images.

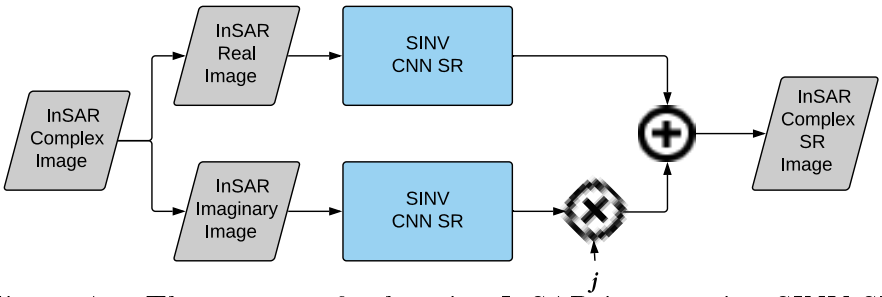


Figure 4.3: The process of enhancing InSAR image using SINV SR

4.3.2 Data

Five SAR images from different sites are used for experimenting with co-registration enhancement and one large InSAR image (19486×32441 pixels) for the city of Vancouver is used for experimenting InSAR image enhancement. The images are generated from ERS-1/2 SLC images. Matlab and Python are the main tools that we employed for SAR image co-registration and coherence computation.

4.3.3 Results

4.3.3.1 Enhancing Co-registration

The results of using SINV to replace the conventional Sinc interpolation for 5 different SAR images are shown in table 4.2. For three cases, the mean of the coherence difference (Sinc Coreg minus from SINV Coreg) is positive. Meaning the SINV Coreg has higher coherence. For two sites, Sinc Coreg is better.

Table 4.2: The mean, variance, and standard deviation for the coherence difference between SINV and Sinc co-registration

Site	Mean	Var	Std
test1	-0.009	0.003	0.0548
test2	0.0019	0.0026	0.0512
test3	-0.0433	0.0048	0.0691
test4	0.0014	0.0029	0.0542
test5	0.0004	0.0026	0.0508

4.3.3.2 Enhancing InSAR Images

We have used the network trained with SAR images as discussed in Section 4.1.3 and applied it to InSAR images as discussed in Section 4.2.2. Table 4.3 presents the results of one such image of Vancouver while Figure 4.4 shows details of the enhanced image. Table 4.3 summarizes the mean of the coherence for SINV SR and Sinc interpolation for the Vancouver area. The mean of the coherence difference is positive. Meaning the SINV SR has higher coherence compared to Sinc interpolation.

Table 4.3: The mean of the coherence difference between SINV SR and Sinc interpolation for upscaling factor of 2

Site	Mean	Var	Std
Vancouver	0.0006	0.0037	0.0609

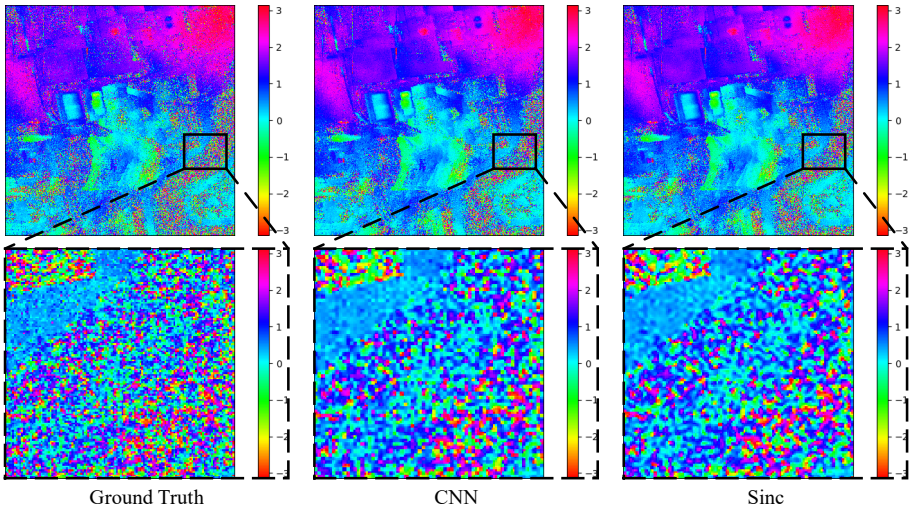


Figure 4.4: A section of an enhanced InSAR image (Vancouver) showing the phase component of the image and for upscaling factor of 2. From left to right, the figure shows the ground truth image, the CNN enhanced image, and the Sinc interpolated image. The CNN enhanced image shows more detail as compared top the Sinc interpolated one

4.4 Chapter Summary

In this chapter, we introduced a new method for enhancing the co-registration process and the Interferometric SAR generation using learning-based super-resolution. The

results show that CNN SR has the potential to enhance SAR imaging at different processing levels; at the co-registration stage and at the InSAR image generation. This work can be extended to employ CNNs in phase unwrapping as well. Moreover, other methods for InSAR images evaluation can be used such as Sum of Phase Differences (SPD) proposed by Li et al. [70].

Chapter 5

Quantum Annealing Methods and Experimental Evaluation to the Phase Unwrapping Problem in Synthetic-Aperture Radar Imaging

The focus of this work is to explore the use of quantum annealing solvers for the problem of phase unwrapping of SAR images. Although solutions to this problem exist based on network programming, these techniques do not scale well to larger-sized images. Our approach involves formulating the problem as a QUBO problem, which can be solved on a quantum annealer. Given that present embodiments of quantum annealers remain limited in the number of qubits they possess, we decompose the problem into a set of subproblems that can be solved individually. These individual solutions are close to optimal up to an integer constant, with one constant per sub-image. In a second phase, these integer constants are determined as a solution to yet another QUBO problem. This basic idea is extended to several passes where each pass results in an image which is subsequently decomposed to yet another set of sub-problems until the resulting image can be accommodated by the annealer at hand. Additionally, we explore improvements to the method by decomposing the original image into overlapping sub-images and ignoring the results on the overlapped (marginal) pixels. We test our approach with a variety of software-based QUBO solvers and on a variety of images, both synthetic and real. Additionally, we experiment using D-Wave Systems quantum annealer, the D-Wave 2000Q_6 and developed an embedding

method which, for our problem, yielded improved results. Our method resulted in high quality solutions, comparable to state-of-the-art phase-unwrapping solvers.

5.1 Introduction

Two-dimensional phase unwrapping is the process of recovering unambiguous phase values from a two-dimensional array of phase values known only modulo 2π rad. The measured phase is also affected by random noise and systematic distortions. This problem arises when the phase is used as a proxy indicator of a physical quantity, which is the time delay between two signals in the case of InSAR [17]. This time delay is significant, as it is affected by the height differences of the illuminated target. It can thus be used to extract accurate three-dimensional topography and reveal topographical changes that occur over time. As the phase is observable only on a circular space where all measured values are mapped to the range $(-\pi, \pi]$, the observed data must be mapped back to the full range of real phase values to be meaningful. For unwrapping purposes, the sampling rate is typically assumed to be suitable for most datasets to prevent aliasing, that is, the absolute difference in phase between two adjacent data points is assumed to be smaller than π. This phase unwrapping problem represents a class of imaging techniques such as InSAR, magnetic resonance imaging, and optical interferometry.

The development of InSAR and many other applications have stimulated interest in building accurate two-dimensional phase unwrapping algorithms. The most commonly used unwrapping technique is based on network programming strategies that formulate the problem as a minimum cost flow (MCF) [15] problem. One of these solvers is the sequential tree-reweighted message passing (TRWS) algorithm [57]. However, since the InSAR images can be quite large—normally larger than $600M$ pixels—the process of phase unwrapping via TRWS can take a prohibitively long time on a classical computer. Hence, we explore whether a quantum computing system could be a potential candidate for solving the phase unwrapping problem.

Quantum computing exploits the laws of quantum mechanics to process information [81]. In contrast to classical computers, which use bits to process information, quantum computers use quantum bits, or qubits, as the basic units of quantum information. Analogously to bits, qubits encode state information. Qubits may be in either of the two distinct states of $|0\rangle$ or $|1\rangle$, but they may also encode a superposition of these states, (i.e., $\alpha\,|0\rangle + \beta\,|1\rangle$ with complex-valued coefficients α and β).

Quantum annealing is a quantum computing method used to find the optimal solution to certain combinatorial optimization problems [26]. This is achieved by using properties of quantum mechanics such as quantum tunnelling, entanglement, and superposition.

Quantum annealing systems are able to solve problems in QUBO form. Any unconstrained quadratic integer problem with bounded integer variables can be transformed by a binary expansion into QUBO form [30]. The phase unwrapping problem is a quadratic unconstrained problem by default, and it can be mapped to a QUBO problem by simply encoding each variable (e.g., k_t in (5.1)) into a vector of binary variables.

Because of limitations in accessing actual quantum annealing infrastructure, we have tested our methodology using a variety of QUBO solvers. As shown in this chapter, the results we obtain match the results obtained using the classical network optimization method (i.e., the TRWS method), which is considered the benchmark in addressing the unwrapping problem.

InSAR images tend to be quite large, often exceeding $20k \times 30k$, or $600M$, pixels. In the simplest problem where each pixel label would require one qubit, a 600M-qubit quantum annealer would be required; such a machine is not currently available. To overcome the limitations of present day technology, we have developed a method where we partition the image and then use quantum annealing on the individual partitions to obtain suboptimal labelling, and then use quantum annealing in a second phase to obtain labels that approach the ones obtained through classical methods. We have named our method "super-pixel decomposition".

The concept of the super-pixel decomposition can be extended to handle bigger images by recursively partitioning the image until the partitions fit in the quantum annealer. Then recursively apply the second phase of the super-pixel decomposition until we obtain labels that approach the global solution. We name this method "multi-pass super-pixel decomposition".

The results can be further enhanced by utilizing additional (marginal) pixels in each of the sub-images processed in the first phase of the method.

This work is an extention and refinement of the work we presented in [54]. In this work, we are presenting the results of the super-pixel decomposition method, the single [54] and the multi-pass approach. In addition, we study the effect of the sub-image size on the quality of the unwrapping. Finally, we are examining the addition of a margin to the sub-image.

The rest of the chapter is organized as follows: In Section 5.2 we provide a background, in Section 5.3 we explain our methodology, in Section 5.4 we present the experimental results, and we conclude with Section 5.5.

5.2 Background

5.2.1 Phase Unwrapping Formulation

As we discussed in Section 2.1.7, two-dimensional phase unwrapping is the process of recovering unambiguous phase values from a two-dimensional array of phase values known only modulo 2π rad.

We concluded that the cost function can be formulated as:

$$E = \sum_{(s,t)\in A} W_{st} \left(k_t - k_s - a_{st}\right)^2 + \sum_{s\in A} \omega_s \left(k_s - a_s\right)^2 \tag{5.1}$$

where k_i are the labels that will determine the original phase, A is the set of pixels in the SAR image, W_{st} are weights defining the neighbourhood structure, a_{ij} are the label estimation constants, and ω_s and the bias a_s are chosen heuristically and represent ad-hoc information one may have on the scene represented in the image.

5.2.2 Quantum Annealing

Quantum annealing employs the quantum tunnelling effect to ensure that a system is able to escape local minima as it traverses the state space of an energy function toward its way to ground-state settlement. The quantum computational systems, such as the ones by D-Wave, use quantum annealing to locate the ground-state of an artificial Ising system [52]. An Ising Hamiltonian describes the behaviour of such a system as

$$H_p = \sum_{i=1}^{N} h_i \sigma_i^z + \sum_{i,j=1}^{N} J_{ij} \sigma_i^z \sigma_j^z \tag{5.2}$$

where h_i is the energy bias for spin i, J_{ij} is the coupling energy between spins i and j, σ_i^z is the Pauli spin matrix, and N is the number of qubits. Quantum annealing

on this system is achieved by the gradual evolution of the Hamiltonian system [52],

$$H(t) = \Gamma(t) \sum_{i=1}^{N} \Delta_i \sigma_i^x + \Lambda(t) H_p \tag{5.3}$$

As time passes, Γ decreases from 1 to 0 while Λ increases from 0 to 1. If the annealing is performed slowly enough, the system stays in the ground state of $H(t)$ for all times, t, ending up at the end of the annealing at the ground state of H_p. The Hamiltonian in (5.2) can be rewritten in vector form as $H(s) = s^T J s + s^T h$, in the form of a QUBO problem [112].

As used in the rest of this chapter, the objective function is expressed in QUBO form in scalar notation, and is defined as follows:

$$C(x) = \sum_i a_i x_i + \sum_{i<j} b_{i,j} x_i x_j \tag{5.4}$$

where $x \in \{0;1\}^n$ is a vector of binary variables and $\{a_i; b_{i;j}\}$ are real coefficients.

Before an application problem can be solved on a quantum annealer, it must first be mapped into QUBO form. As a first step in transforming the InSAR problem into a QUBO problem, the k_i label that is non-binary valued must be transformed into binary valued. Let $k_i \in \{0, D_i - 1\}$, where D_i is the number of allowed values (labels) for k_i. This can be achieved by writing k_i in binary. The binary transformation restricts the number of new-valued binary variables required to represent k_i. Let $d_i = \lceil \log_2 D_i \rceil$ and $k_i = \langle \mathbf{2}, \mathbf{x_i} \rangle$ where the vector $\mathbf{x_i} = [x_{i,d_i}, \cdots, x_{i,1}, x_{i,0}]$ represents the bits of k_i and $\mathbf{2} = [2^{d_i}, \cdots, 2, 1]$ is the vector of powers of two. Equation (5.1) can be written in QUBO form as:

$$E = \sum_{(s,t)\in A} W_{st} \left(\sum_i b_i x_{i,t} - \sum_i b_i x_{i,s} - a_{st} \right)^2 + \sum_{s \in A} \omega_s \left(\sum_i b_i x_{i,s} - a_s \right)^2 \tag{5.5}$$

where b_i is the weighting coefficient for the binary variable x_i ($b_i = 2^i$ in the case of the binary encoding).

Many problems can be formulated to take advantage of quantum annealing, and is advantageous because it converges faster than other techniques to an optimum solution [53].

Quantum annealing can be compared to simulated annealing by identifying that

the temperature parameter in simulated annealing performs a similar role to quantum tunnelling in quantum annealing. The temperature in simulated annealing defines the probability of moving from a single current state to a higher energy state to escape local minima. The assumed advantage of quantum annealing over simulated annealing is that tunnelling allows the system to directly pass through high energy barriers without having to climb over them.

Analytical and numerical evidence indicates that quantum annealing can outperform simulated annealing [38]. Therefore, quantum annealing is a good potential solver for the InSAR phase unwrapping problem.

5.2.3 Optimizers

As we mentioned earlier, the size of the problem does not allow the direct use of currently available annealing infrastructure. Similarly, the size of the problem results in a prohibitively expensive QUBO computation if we elect to perform a global optimization on the full scale image. Rather, our methodology partitions the image, and QUBO solvers are applied first on the partitions and then in a second phase on an abstraction of the image comprised of what we call super-pixels, each one representing a partition of the original image.

In the following sections, (5.2.3.1 and 5.2.3.2), we shall discuss the QUBO solvers we have employed, and then our approach of partitioning the image and the super-pixel methodology.

5.2.3.1 Classical Optimizer

5.2.3.1.1 TRWS [57] The TRWS algorithm is used for discrete energy minimization where the energy function can be formulated as follows:

$$E\left(x|\theta\right) = \theta_{\text{const}} + \sum_{s\in\nu} \theta_s\left(x_s\right) + \sum_{(s,t)\in\varepsilon} \theta_{st}\left(x_s, x_t\right) \tag{5.6}$$

where ν corresponds to the set of pixels; x_s indicates the label of pixel $s \in \nu$, ε corresponds to the set of edges (each edge connects two related pixels), $\theta_s(\cdot)$ is the penalty function (i.e., a term of an unconstrained objective function added to add some constraint to it) of unary data, and $\theta_{st}(\cdot,\cdot)$ is the penalty function of the pairwise terms. This energy function is usually derived in the context of Markov random fields [29]. The algorithm is widely used in phase unwrapping problems, where the unary

penalty functions represent the unary terms in (5.1) where the pixels are penalized for having large values, while the pairwise penalty functions represent the pairwise terms in (5.1) where the two pixels k_t and k_s are penalized for having a difference not equal to a_{st}.

5.2.3.2 QUBO Optimizer

5.2.3.2.1 Microsoft Quantum-Inspired Optimization (MQIO) Quantum-Inspired Optimization (QIO) is a classical computing method. It refers to a class of algorithms inspired by quantum computing that are applied to solve optimization problems on traditional hardware. MQIO supports four solvers; Simulated Annealing, Parallel Tempering, Tabu Search, and Quantum Monte Carlo. We are using the two solvers; Simulated Annealing and Parallel Tempering.

5.2.3.2.2 Parallel Tempering This solver is one of MQIO solvers and accessible through a cloud-client interface. However, we have limited early access to this solver. Hence, we used it only on small images.

5.2.3.2.3 Simulated Annealing This solver provides an implementation of the simulated annealing method [96]. The solver is also one of the Microsoft QIO solvers. Therefore, we used this solver on small images for the same reasons mentioned above.

5.2.3.2.4 D-Wave Annealing D-Wave Systems provides implementations of different quantum annealing systems, starting from the D-Wave One announced in 2011 [52]. We used the D-Wave 2000Q_6 machine to unwrap the InSAR sub-images. The machine contains 2041 qubits. The qubits are sparsely connected in an architecture known as a "Chimera" graph. The Chimera architecture comprises sets of connected unit cells. Each unit cell has two columns of four vertical qubits that are connected via couplers. All qubits in one column are connected to all qubits in the other column. However, the qubits within a column are not connected (see figure 5.1). Unit cells are tiled horizontally and vertically with adjacent qubits connected. The qubits are logically mapped into a matrix of 16×16 unit cells, with eight qubits per cell. Each qubit is connected to six neighbours in the Chimera topology, four connections to the other qubits within the cell and two connections to the horizontally adjacent cells (or vertically adjacent cells if the qubit is in the left column). In theory, the Chimera architecture comprises $16 \times 16 \times 8 = 2048$ qubits. In practice,

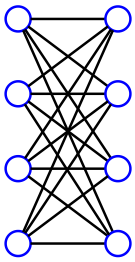


Figure 5.1: Chimera architecture unit cell. The circles represent qubits, and the edges represent couplers

however, the largest number of embeddable qubits is slightly smaller (2041 qubits) due to missing, or faulty, qubits, an issue that arises during manufacturing. This also results in there being some nonexistent connections.

Solving a QUBO problem on a quantum annealer requires embedding (mapping) each binary variable to one physical qubit or multiple chained (connected) qubits and mapping the quadratic coefficients of the objective function (in a QUBO form) into couplers. Couplers provide the connections between the qubits to form the Chimera graph as explained previously. Couplers are "programmable", i.e., their strength can be adjusted to reflect the coefficients of the cost function. D-Wave uses superconducting loops to implement couplers.

The embedding process varies based on the problem and the architecture of a given quantum annealer. D-Wave Systems provides a tool [11] that heuristically embeds the binary variables to the quantum annealer's qubits. In our experience, the tool does not provide the optimal embedding in terms of keeping more related qubits closer to each other. Hence, we used a manual approach to embed the logical binary variables onto physical qubits.

In manually embedding our variables onto qubits, we strived to produce a symmetric embedding. The symmetry of the embedding contributed to improved solution quality.

To each pixel, there corresponds a label. This is the unknown integer variable, which we are trying to determine its value by minimizing the cost function developed earlier. Assuming the values the labels attain are always less or equal to a maximum value of D, then one needs $d = \lceil log_2 D \rceil$ binary bits to encode the label (integer variable). In an ideal case where the underlying topology of the quantum annealer is a complete graph, one would map each of the binary variables of the encoding to a single qubit. In cases where the underlying topology is not a complete graph (as

in the case of the D-Wave annealers), one needs to map a binary variable to several linked qubits to ensure the availability of a sufficient number of couplers to express the coefficients of the cost function. The chosen qubits are chain-linked together, i.e., have the linking couplers set to a maximum value to ensure that all the linked qubits attain the same value when they are read.

By expanding the equation (5.5), we can observe that each binary variable x_i (logical bit) needs to be connected (because it falls in the same term) to all binary variables belonging to the same integer variable (the same pixel) and all the binary variables belonging to the connected integer variables (the connected pixels). In our implementation, each pixel is connected to its four adjacent pixels. This is the minimum number of connections needed to unwrap an image using a grid that connects all the pixels horizontally and vertically. This means that each binary variable needs to be connected to $5d$ binary variables. First, it needs to be connected to d binary variables belonging to the same pixels. Second, it needs to be connected to $4d$ binary variables belonging to the four neighbouring pixels.

In addition to the couplers between the qubits in the same cell, each qubit of a Chimera cell is connected to two qubits in an adjacent cell as discussed earlier. Therefore, to allow a four-neighbor connectivity one needs to chain at least two qubits. This will realize an embedding where $D = 2$, i.e., a problem with a maximum label value of 2. For more complex problems, one would opt of chain linking more than 2 qubits. In this work , we have elected to experiment with problems with a maximum label of 4. This is a realistic constraint as the real images in our dataset have labels with a maximum value of 5. In the remaining of the section, we shall discuss our approach to embedding the super-pixel approach on the D-Wave 2000Q_6 system. We use two binary variables q_0 and q_1 to encode a maximum-label of 4 problem. We shall map each binary variable to four chain-linked qubits. Choosing 4 qubits to chain-link allows for a graph of degree 8 which is sufficient to express the four nearest neighbor connectivity of our problem.

Figure 5.2 depicts the embedding of an integer variable comprised of two binary variables (q_0 and q_1) into a unit cell and the connections with the adjacent cells. In the figure, the chain link (in black) represents the chains between the qubit within the cell (the chain that constructs the integer variable), and the graph link (in red) represents the chains between qubits of different cells (the connections between integer variables).

Since there are no couplers available in the Chimera graph between the qubits

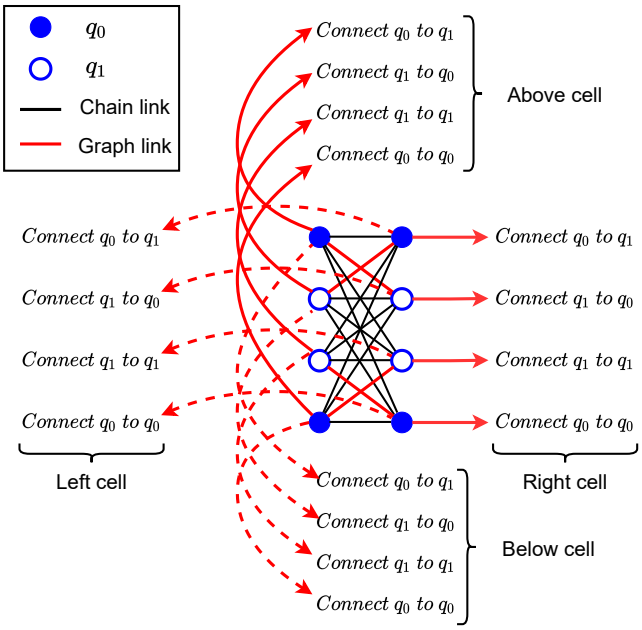


Figure 5.2: The embedding of an integer variable comprised of two binary variables (q_0 and q_1) into a unit cell and the connections with the adjacent cells

within the cell column, the four qubits needed to be chained (to connect to the adjacent cells) must not all fall within the same column of the cell. Otherwise, we will not be able to chain them. Therefore, we selected the first and the last rows of the unit cell to represent the least significant bit of the integer variable (q_0) and the middle rows to represent the most significant bit (q_1) and called this representation unit cell A representation. Moreover, the last two rows' representation must be switched for the cells adjacent to cell unit A to allow different bits of different integer variables to connect (we call this representation unit cell B).

To sum up, we mapped each integer label to a single Chimera cell. In the present embedding, each integer label represents four distinct label values and is encoded using two binary variables. With each binary label variable mapped to 4 qubits of the D-Wave annealer, the integer label is mapped to a single Chimera cell (a Chimera cell comprises eight qubits). The interconnections between the qubits of the same logical bit (chain links), the interconnections between the logical bits within the cell, and the interconnections between the cells (graph links) are shown in Figure 5.3.

The largest image our manual embedding approach can map onto the D-Wave 2000Q_6 is 16×16 pixels in size. Bypassing the faulty qubits resulted in asymmetries, which lowered the quality of the solutions. To avoid such issues, we have experimented with smaller images 10×10 pixels in size which, when mapped, avoid the faulty qubits with a concomitant increase in the quality of the solution.

In order to translate the state of qubits to the problem binary variables, the obtained solution from the annealer must be unembedded. That is translating the values of the chained qubits into a binary value. Since the chained qubits represent the same binary variable, their values must agree. If the values are different due to the probabilistic nature of the quantum annealer, we use the majority vote to determine the value of the binary variable.

In experimenting with the two options, i.e., the manual embedding and the automatically-generated embedding, we found that on average, the automatically generated embedding provided unwrapping solutions with a matching fraction of 62% while the manual embedding resulted in an average matching fraction of 83%.

5.3 Methodology

In this section, we describe our methodology in breaking down the phase-unwrapping problem into smaller problems that are easier to solve. The smaller problems can be

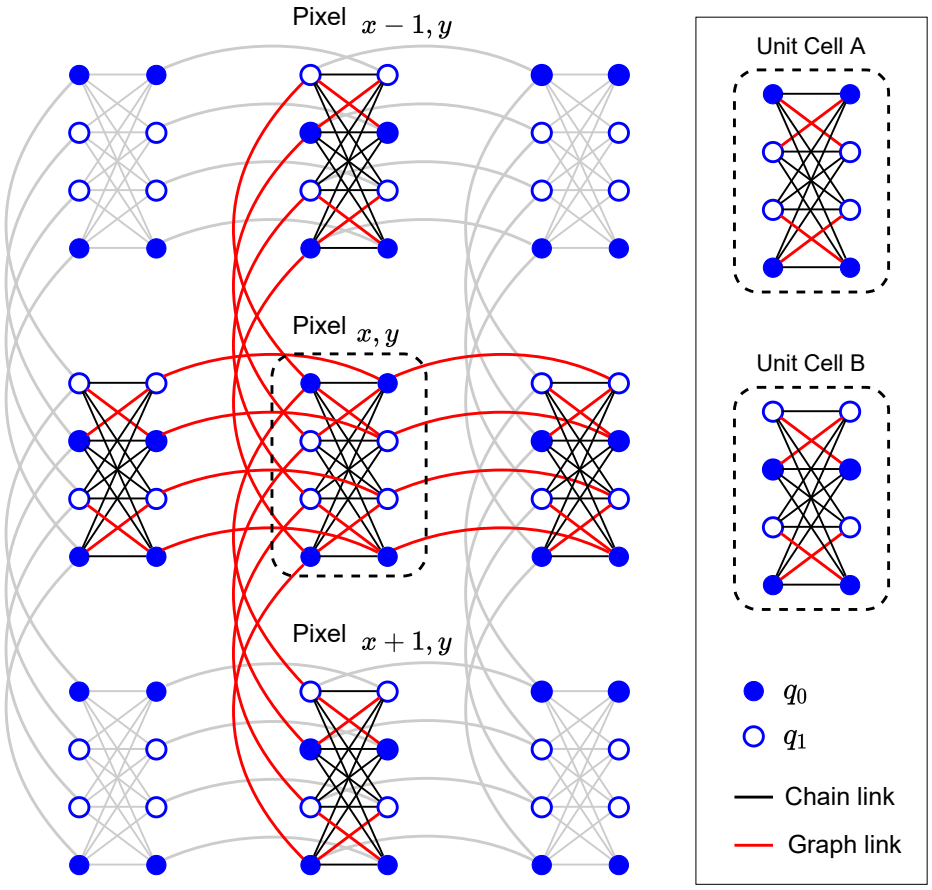


Figure 5.3: The manual embedding mapping

solved in parallel, and approach the global solution.

5.3.1 Phase Unwrapping Decomposition

Equation (5.1) describes the energy function for phase unwrapping. The equation consists of two terms: the pair-wise term, $W_{st}\left(k_t - k_s - a_{st}\right)^2$, that describes the relationship between two pixels s and t, and the unary term, $\omega_s\left(k_s - a_s\right)^2$, that describes the energy of the pixel s.

Phase Unwrapping Decomposition is based on a divide-and-conquer approach. A given large InSAR image is subdivided into smaller sub-images that fit onto a quantum annealing machine. Each of these sub-images are unwrapped independently. Then the sub-images are stitched together to form the final unwrapped large InSAR image.

This approach introduces a problem at the boundaries of the sub-images. Since we are unwrapping the sub-images separately, there is no guarantee that the labelling of two pixels, each belonging to a boundary of two adjacent sub-images, will attain optimal labelling consistent with that which would be obtained if both pixels had been part of the same optimization problem (i.e., if we had unwrapped the entire image all at once).

We assume that the pixels in each sub-image are labelled correctly up to an integer additive factor, where all the pixels of one sub-image share the same additive factor. Then, our objective is to find those additive factors such that all the pixels at the boundaries will be consistent.

We propose a super-pixel heuristic to determine these additive factors as follows:

The wrapped InSAR image is divided into non-overlapping sub-images, where each sub-image contains a subset of the pixels. The energy function determined by (5.1) can be rewritten as:

$$E = \sum_{g \in G} \left[\sum_{(s,t) \in A_g} W_{st}\left(k_t - k_s - a_{st}\right)^2 + \sum_{s \in A_g} \omega_s\left(k_s - a_s\right)^2 \right] + \sum_{\substack{t \in A_i, \\ s \in A_j, \\ i \neq j}} W_{st}\left(k_t - k_s - a_{st}\right)^2 \quad (5.7)$$

where G is the set of the sub-images, and A_x is the set of pixels in the sub-image x.

In the equation above, the terms within the square brackets correspond to an

energy function for each sub-image while the last sum collects all the terms that connect the sub-images. Our approach is to optimize each of the sub-images, that is, to determine the labels that optimize the energy functions corresponding to each sub-image separately. We assume next that the obtained solutions are correct—that are identical plus or minus a sub-image wide integer shift—to the solution obtained when we optimize the image in its totality. The next step is to determine these additive factors, which can be formulated as a QUBO problem. Let K_s denote additive factor corresponding to sub-image s, and let $k_i^{'}$ denote the label of pixel i as determined by the QUBO of each sub-image. Then label k_i of pixel i can be written as $k_i = k_i^{'} + K_s$ for i in A_s.

Equation (5.7) can now be rewritten as:

$$E = \sum_{g \in\ G} \left[\sum_{(s,t) \in A_g} W_{st} \left(k'_t + K_g - k'_s - K_g\right)^2 + \sum_{s \in A_g} \omega_s \left(k'_s + K_g - a_s\right)^2 \right] + \sum_{\substack{t \in A_i, \\ s \in A_j, \\ i \neq j}} W_{st} \left(k'_t + K_i - k'_s - K_j - a_{st}\right)^2 \quad (5.8)$$

The first sum within the bracket is devoid of K_g and it is constant since the labels $k_s^{'}$ have been determined by the previous QUBO operation. Ignoring constant terms, (5.8) can be rewritten as:

$$\widetilde{E} = \sum_{g \in G} \left[\sum_{s \in A_g} \omega_s \left(k'_s + K_g - a_s\right)^2 \right] + \sum_{\substack{t \in A_i, \\ s \in A_j, \\ i \neq j}} W_{st} \left(\left(k'_t - k'_s\right) + \left(K_i - K_j\right) - a_{st}\right)^2 \quad (5.9)$$

denoting as $a_s^{'} = a_s - k_s^{'}$ and $a_{st}^{'} = a_{st} - \left(k_t^{'} - k_s^{'}\right)$ then (5.9) is written as:

$$\widetilde{E} = \sum_{g \in G} \left[\sum_{s \in A_g} \omega_s \left(K_g - a'_s\right)^2 \right] + \sum_{\substack{t \in A_i, \\ s \in A_j,, \\ i \neq j}} W_{st} \left(K_i - K_j - a'_{st}\right)^2 \quad (5.10)$$

The first term of (5.10) is a second order function while the second term regularizes the solution by selecting K_g to be as small as possible. The coefficient ω_s and the term a_s as per (5.1) were chosen arbitrarily. To ensure that our energy function conforms to the form of (5.1) and without affecting the accuracy of the solution, we select $\omega_s = \omega_g$ and $a_s^{'} = a_g \; \forall s \in A_g$. This results in the following expression for the energy function:

$$\hat{E} = \sum_{\substack{t \in A_i, \\ s \in A_j, \\ i \neq j}} W_{st} \left(K_i - K_j - a_{st}^{'} \right)^2 + \sum_{g \in G} \omega_g \left(K_g - a_g \right)^2 \tag{5.11}$$

This is the energy function of the super-pixel level where K_g represents the sought labels for each sub-image (i.e., super-pixel). Since it is a quadratic unconstrained integer optimization problem, it is amenable to a QUBO solution.

5.3.2 Multi-pass Super-pixel

In the Phase Unwrapping Decomposition explained above. The image is unwrapped in two stages. First, unwrap the sub-images, and then unwrap the super-pixel image. The incentive was to fit the problem onto a quantum annealing machine. The size of the sub-image is limited by the size of the problem the quantum annealer can solve. The size of the super-pixel image is in turn determined by the sub-image size. Since the super-pixel image is also unwrapped by a quantum annealing machine, the super-pixel image might not fit onto the quantum annealing machine. In this case, the super-pixel can be divided into sub-images and the super-pixel algorithm can be performed again to obtain a solution for the super-pixel image. In other words, the image are unwrapped using two passes of the super-pixel algorithm.

In this way, the super-pixel algorithm can be recursively applied through multiple passes of the super-pixel decomposition.

For instance, consider an image of size 400×400 pixels, while the annealer can handle at maximum 10×10 pixels. Selecting the sub-image size of 10×10 will result in a super-pixel image of 20×20 pixels, where each pixel represents a sub-image additive integer. The 20×20 super-pixel image will not fit into the annealer. Hence, we consider re-applying the super-pixel algorithm again on the super-pixel image by dividing it into four images of the size 10×10 pixels. This will result in a 2×2 second pass super-pixel image to unwrap the 20×20 first pass super-pixel image.

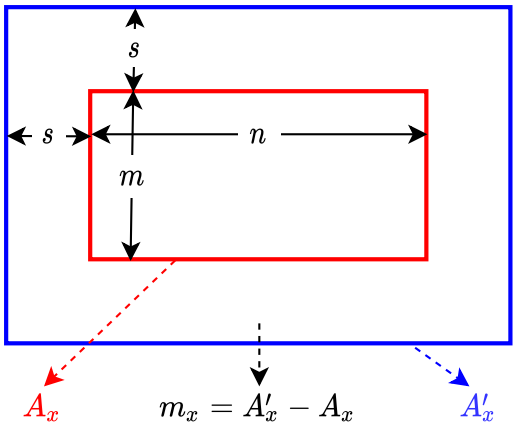


Figure 5.4: The approach of adding margin to the sub-image. The red box represents the sub-image with $A_x = m \times n$ pixels, the blue box is the sub-image after adding the margin with total $A'_x = (m + s) \times (n + s)$ pixels, and the area in between is the margin of $m_x = A'_x - A_x$ pixels added around the sub-image

5.3.3 Adding a Margin to the Sub-images

We postulated earlier that the pixels in each of the sub-images were labeled correctly (within an additive constant) when these sub-images were unwrapped independently of each other. However the boundary pixels belonging to neighboring sub-images do play a role in deciding the label of the pixels in the sub-image in question. Thus, labeling errors are introduced. In order to improve the quality of the unwrapping, we propose to unwrap sub-images that overlap. That is, include a margin of s pixels around the sub-image. However, as we proceed with the second phase of our method, we only keep the labels of the non-overlapping parts of the sub-image. The approach is shown in Figure 5.4, and discussed in the subsequent paragraphs.

Consistent with the notation introduced above, we denote by A'_x the set of pixels of the overlapping sub-image x while the set of pixels of the non-overlapping sub-image is denoted as A_x. The set of pixels in the margin is therefore $m_x = A'_x - A_x$ where the minus $(-)$ sign denotes a set-theoretic subtraction.

Equation (5.7) can now be written as:

$$
\begin{aligned}
E = \sum_{g \in G} (& \sum_{(s,t) \in A_g} W_{st} \left(k_t - k_s - a_{st}\right)^2 + \sum_{s \in A_g} \omega_s \left(k_s - a_s\right)^2 \\
& + \sum_{(s,t) \in m_g} W_{st} \left(k_t - k_s - a_{st}\right)^2 + \sum_{s \in m_g} \omega_s \left(k_s - a_s\right)^2 \\
& + \sum_{\substack{s \in m_g, \\ t \in A_g}} W_{st} \left(k_t - k_s - a_{st}\right)^2 + \sum_{\substack{s \in A_g, \\ t \in m_g}} W_{st} \left(k_t - k_s - a_{st}\right)^2 \\
& - \sum_{(s,t) \in m_g} W_{st} \left(k_t - k_s - a_{st}\right)^2 - \sum_{s \in m_g} \omega_s \left(k_s - a_s\right)^2 \\
& - \sum_{\substack{s \in m_g, \\ t \in A_g}} W_{st} \left(k_t - k_s - a_{st}\right)^2 - \sum_{\substack{s \in A_g, \\ t \in m_g}} W_{st} \left(k_t - k_s - a_{st}\right)^2 \\
&) + \sum_{\substack{t \in A_i, \\ s \in A_j, \\ i \neq j}} W_{st} \left(k_t - k_s - a_{st}\right)^2 \quad (5.12)
\end{aligned}
$$

In equation (5.12) we added and subtracted the terms that correspond to the margins of the sub-images. We can now consolidate the margins with the non overlapping sub-images to formulate an energy function based on the overlapping sub-images as follows:

$$
\begin{aligned}
E = \sum_{g \in G} (& \sum_{(s,t) \in A'_g} W_{st} \left(k_t - k_s - a_{st}\right)^2 + \sum_{s \in A'_g} \omega_s \left(k_s - a_s\right)^2 \\
& - \sum_{(s,t) \in m_g} W_{st} \left(k_t - k_s - a_{st}\right)^2 - \sum_{s \in m_g} \omega_s \left(k_s - a_s\right)^2 \\
& - \sum_{\substack{s \in m_g, \\ t \in A_g}} W_{st} \left(k_t - k_s - a_{st}\right)^2 - \sum_{\substack{s \in A_g, \\ t \in m_g}} W_{st} \left(k_t - k_s - a_{st}\right)^2 \\
&) + \sum_{\substack{t \in A_i, \\ s \in A_j, \\ i \neq j}} W_{st} \left(k_t - k_s - a_{st}\right)^2 \quad (5.13)
\end{aligned}
$$

The energy function described by the above equation includes the energy function of the overlapping sub images, i.e., $E'(k) = \sum_{g \in G} [\sum_{(s,t) \in A'_g} W_{st} \left(k_t - k_s - a_{st}\right)^2 + \sum_{s \in A'_g} \omega_s \left(k_s - a_s\right)^2]$. We propose to optimize these overlapping sub-image energy functions, i.e., $\arg\max_k E'(k)$, separately. Then, by introducing these labels back into

equation (5.8), and following the subsequent steps outlined in Section 5.3.1 we can unwrap the entire image based on the labels computed through the margin enhanced sub-images.

5.4 Experiments

In this part of the work, we present the experimental approach we followed in order to study the effectiveness of the proposed super-pixel method. Our methodology included the use of different solvers, a number of image datasets, and the analysis of the results we obtained. The datasets we used included synthetic InSAR images as well as real InSAR images. We have used synthetic images as they afforded us the opportunity to control the complexity and their noise content. The solvers we employed included TRWS, which is the current industry standard, as well as QUBO solvers including a quantum annealer (D-Wave 2000Q_6).

The TRWS solver is readily accessible and since it is the industry standard, it was used to study the effectiveness and establish the performance baseline of our proposed super-pixel method on a wide variety of images. The QUBO solvers being less readily accessible, were used in more restrictive experiments to demonstrate the feasibility of our proposed method on such solvers. Also, a comparisons of the effectiveness of the different solvers used could be compared.

In the subsections that follow, we describe the image datasets we used. Then, we shall present our experimentation using the TRWS solver and then the experimentation using the QUBO solvers. Each solver, or class of solvers, shall be used to study the one- and two-pass super-pixel method as well as the effect of adding margins to the sub-images. After each experiment, we shall summarize our findings. All the methods have been described in Section 5.3.

5.4.1 Image Datasets

5.4.1.1 Real Images

A real InSAR images dataset is constructed using 3 large real InSAR images. The images are partitioned at random to have the size 400×400 to be consistent with the results of the synthetic images. The partitioning resulted in generating multiple images, in some case, when the original image size is large enough to fit more than

Table 5.1: The properties of the original real images

Image Name	Max label	Size	Generated Image Size	# Generated Images
Mexico City	4	500×500	400×400	4
Las Vegas	4	960×1080	400×400	1
Vancouver	5	1619×974	400×400	4

Table 5.2: The maximum label for the real InSAR images dataset

Image Name	Max label
Mexico City	4
Las Vegas 0	4
Las Vegas 1	3
Las Vegas 2	4
Las Vegas 3	3
Vancouver 0	5
Vancouver 1	5
Vancouver 2	5
Vancouver 3	5

one 400×400 image. The images have a maximum label that varies between 3 and 5. Table 5.1 summarizes the original images properties.

Table 5.2 summarizes the properties of the generated dataset.

5.4.1.2 Synthetic Images Dataset 1

This is the main synthetic dataset. This dataset presents a larger exploration space, with a wide spectrum that includes high-frequency data that present a challenge for the phase unwrapping process. 400×400 pixels synthetic images are generated with the parameters:

1. Image complexity (three levels; low, medium, and high).

2. Noise content (four levels; free, low, medium, and high).

Table 5.3: The properties of the images generated for the synthetic dataset 1

Max label	4	8	16
Low- noise SNR (dB)	13	13	13
Medium- noise SNR (dB)	10	10	10
High- noise SNR (dB)	7	7	7
Low- complexity correlation (Perlin noise)	12	6	3
Medium- complexity correlation (Perlin noise)	18	9	4.5
High- complexity correlation (Perlin noise)	24	12	6

3. The number of labels (three different numbers of labels; 4, 8, and 16).

This gives us 36 different sets. 10 images are generated per set to give in total 360 images. The synthetic images are generated using Perlin Noise Generator (discussed briefly in Appendix C). Table 5.3 summarizes the properties of the generated images.

5.4.1.3 Synthetic Images Dataset 2 (Pseudo-real)

This dataset includes images that have a spectrum that is similar to that of the real images' dataset. We postulate that such synthetic images have computational complexity similar to that of the real images and our methodology can be evaluated fairly using these synthetic images. We generated 10 images with 400×400 pixels. The maximum label (maximum number of wrappings) of the real images varies between 3 and 5 with average maximum label of 4 (Table 5.3). Therefore, we generated the pseudo-real images with the maximum label of 4. The images are considered of low complexity with a Perlin noise correlation of 0.5 and very noisy with an SNR of 3 db. Appendix C summarizes how we generated this dataset to have a close spectrum to the real dataset.

5.4.2 Metrics

To determine how close two images (of identical size) are to each other, we use the matching fraction metric defined as the fraction of pixels that are identical in the two images. In order to evaluate the accuracy of an image processing method, we compare the obtained image to the "ground truth" through their matching fraction.

However, depending on the goals of the analysis, different "ground truth" images may be used.

Following, we enumerate the possible "ground truth" cases we used in this research.

5.4.2.1 Noise-free Ground Truth

This is the original image obtained through an ideal noise-free sensor. However, such images cannot be had in reality as sensors introduce noise and artefacts. Synthetic images though, can be synthesized as noise and artefact free and they can be used as noise-free ground truth images in our experimentation and analysis.

5.4.2.2 Noisy Unwrapped Ground Truth

Noisy Unwrapped Ground Truth images are the images obtained by a sensor. Such images are common in photography; however, SAR (and InSAR) images are wrapped. Noisy Unwrapped Ground Truth images can be obtained synthetically by adding noise to synthetic noise-free ground truth images.

5.4.2.3 Deemed Noisy Unwrapped Ground Truth

Deemed Noisy Unwrapped Ground Truth images are images that were obtained from wrapped noisy images through the application of a state-of-the-art unwrapping method. For our purposes, the state-of-the-art unwrapping method used is TRWS.

In this work, we primarily use the Noisy Unwrapped Ground Truth images as the base to compare the images obtained as a result of our methods. Where the Noisy Unwrapped Ground Truth does not exist, we use the Deemed Noisy Unwrapped Ground Truth. These are the cases of real images. As we show in Appendix D, the Deemed Noisy Unwrapped Ground Truth images are very close to the Noisy Unwrapped Ground Truth.

5.4.3 Solvers

5.4.3.1 TRWS

Since we have no restrictions on using TRWS, we use it to provide a detailed analysis of the performance of our super-pixel approach in solving the unwrapping problem. The experiments using TRWS is applied to all images in the synthetic dataset 1 and/or the real and the pseudo-real datasets.

5.4.3.2 QUBO

QUBO solvers extend the solvers used with the super-pixel algorithm to include a hardware-based QUBO solver (D-Wave annealer) and software-based QUBO solvers (MQIO solvers). The discussion presented in the TRWS experiments provides a detailed analysis of the performance of our super-pixel approach in solving the unwrapping problem.

We present an argument supporting the use of quantum and/or quantum inspired solvers in the super-pixel algorithms. Since the quantum solvers are not widely accessible and cannot yet support very large problems, by necessity our experiments are limited in the size and number of problems we can solve. The experiments here are to be considered as complementary and validating the analysis we presented in the previous sections.

Given the limited access we have to the QUBO solvers, we had to choose between applying the one-pass super-pixel approach and the two-pass super-pixel approach. We used the two-pass super-pixel approach as it suits the QUBO solvers best (large images of 120×120 pixels with small sub-images of maximum size of 10×10 pixels). Also, the two-pass super-pixel algorithm underperforms the one-pass super-pixel one. Hence, it is expected that the one-pass super-pixel algorithm to yield superior performance should the solver have sufficient resources (e.g., number of qubits) to accommodate it.

For all the images we used with the QUBO solvers, the images are cropped to the size of 120×120. We selected the cropped part randomly within the selected image. The selected size of the image fits different sub-image sizes. The sub-image sizes we used are 6×6, 8×8, and 10×10.

5.4.3.2.1 D-Wave Annealer The super-pixel algorithm is tested in this work using a D-Wave annealer (D-Wave 2000Q_6 with 2041 qubits) as a sub-image solver. The D-Wave annealer is configured to $maximum\ label = 4$, even for images with a higher maximum label due to the limited number of qubits. Table 5.4 summarized the configurations for the D-Wave annealer.

5.4.3.2.2 MQIO Solvers We present in this work the results of using MQIO as a sub-image solver for the super-pixel algorithm. Namely, the two solvers, Simulated Annealing and Parallel Tempering. MQIO solvers can support maximum label greater than 4 for sub-images that are relatively small (less than 20×20 pixels). However,

Table 5.4: The configurations of the D-Wave annealer

Property	Value	Property	Value
chain scale	0.6	use_ising	True
num_reads	100	manual_embedding	True
chain_strength	1	solver_name	'DW_2000Q_6'
h_threshold	0.0003	max label	4
J_threshold	0.002	encoding	binary
auto_scale	False		

we fixed the MQIO solvers to have a maximum label of 4 to have results that are compatible with those of the D-Wave solver.

5.4.4 Evaluating the Super-pixel Algorithm

In this section, we test the one-pass super-pixel and the multi-pass super-pixel algorithms. We start with the TRWS and apply it to the datasets. Then we experiment with the QUBO solvers and apply it to limited number of real and synthetic images.

5.4.4.1 TRWS as Sub-image Solver

In this subsection, we test the one-pass super-pixel and the multi-pass super-pixel algorithms using TRWS as sub-image solver. First, the 400×400-pixel images are partitioned into 400 20×20-pixel sub-images and processed with the one-pass super-pixel algorithm. Second, the 400×400-pixel images are partitioned into 1600 10×10-pixel sub-images and processed with the two-pass algorithm. In both cases, we used TRWS as the sub-image solver.

5.4.4.1.1 Synthetic Dataset 1 Results We experimented with all the synthetic images in dataset 1 (360 images in total). All images are 400×400 pixels. We tried the one-pass super-pixel algorithm (using 20×20 sub-images) and the two-pass algorithm (using 10×10 sub-images). Our methods unwrap each of the sub-images separately and then combine the results to obtain the unwrapping of the full 400×400 image. To evaluate our methods, we use the Noisy Unwrapped Ground Truth images as a basis of comparison.

Figure 5.5 and Figure 5.6 compare the performance of our methods. Figure 5.5 and Figure 5.6 compare the unwrapped image obtained by our methods to an expected image for the one-pass and two-pass super-pixel algorithm, respectively. This expected image (Noisy Unwrapped Ground Truth) is the unwrapped noisy image which is wrapped and the wrapped image is then used by our methods.

Analyzing the results as presented in Figure 5.5 and Figure 5.6, one can draw some general observations. The noise and image complexity are the important factors in the quality of the unwrapping process.

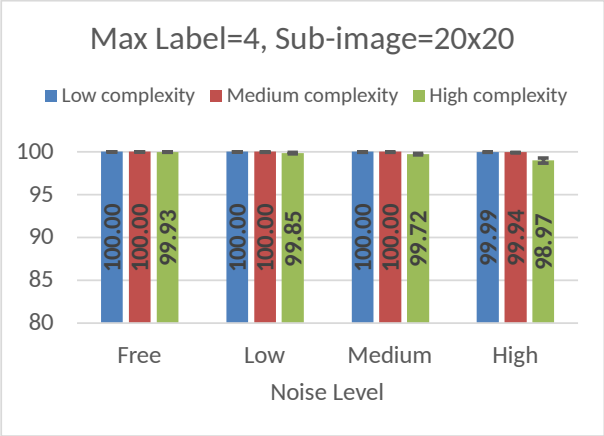
If the images are noise-free and of low or medium complexity (blue and red bars in the figures), our approach unwraps the image successfully (the matching fraction is 1) and produces identical results with the ground truth. This can be seen in Figure 5.5-a,b,c and Figure 5.6-a,b,c. However, as these figures show, the complexity of the image affects the quality of the solution of our approach marginally. Our approach, in the worst case of highly complex images and a large number of labels (16), deteriorates by 3.4% to a matching fraction of .966 as shown in Figure 5.6-c.

As one would expect, noise plays a major role in the success of unwrapping algorithms. In the worst case, depicted in Figure 5.6-c, the quality of the resulting image as compared to the original noisy image deteriorates by about 12% to a matching fraction of 0.8837. This result covers images with high noise and high complexity.

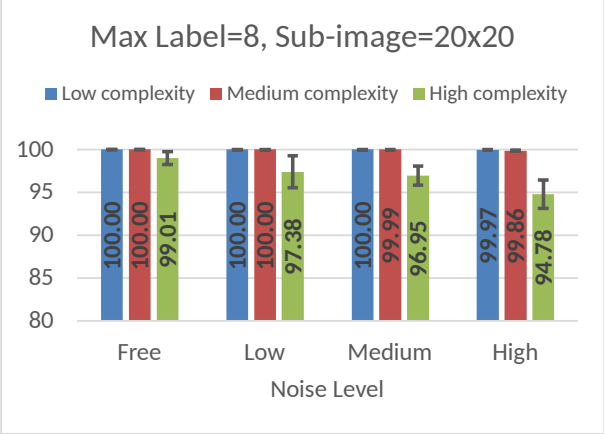
As the size of the sub-images decreases (as in our two-pass approach), there is some deterioration of the quality of the solution which is more pronounced for high-complexity and high-noise images. In the worst case, the deterioration is about 2% (0.9023 for a high-noise high-complexity 16-label image to .8837 Figure 5.5-c and Figure 5.6-c). We can reach the same conclusion by gradually decreasing the size of the sub-image size as presented in more details in Appendix G.

Overall, we can conclude that both the one-pass or two-pass super-pixel algorithms produce excellent results for low and medium complexity images (the worst case is depicted in Figure 5.6-c with a matching fraction of 0.9983). Real SAR and InSAR images are typically of low or medium complexity. Our approach produces acceptable results for high-complexity images (the worst case is depicted in Figure 5.6-c at 0.8837).

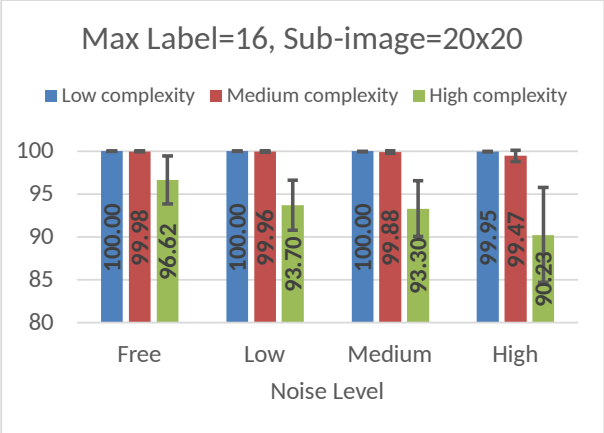
5.4.4.1.2 Real Image and Pseudo-real Image Datasets Results This section presents and compares the Synthetic dataset 2 (pseudo-real) to the real dataset.



(a) Max label 4

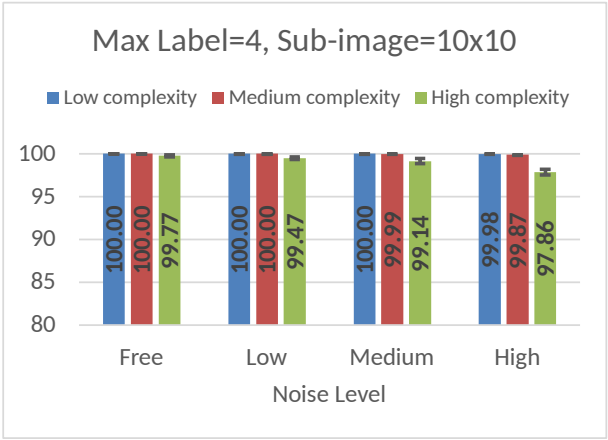


(b) Max label 8

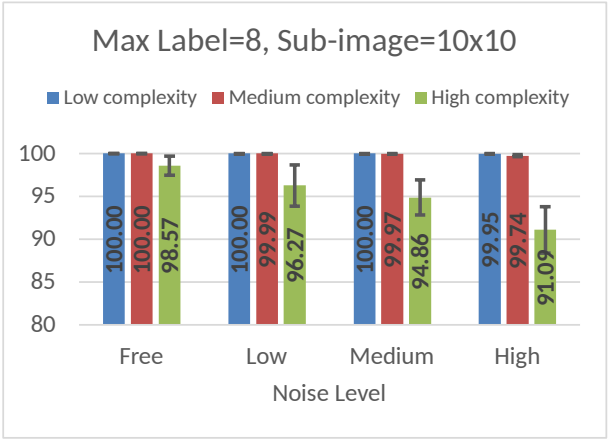


(c) Max label 16

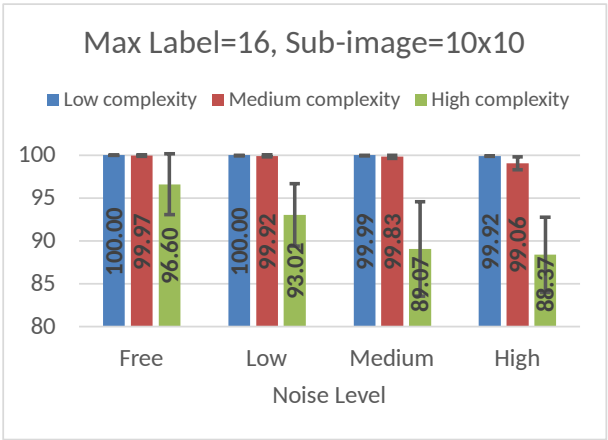
Figure 5.5: The mean matching fraction of the images in dataset 1 obtained through the one-pass super-pixel algorithm for different maximum labels, compared to the Noisy Unwrapped Ground Truth. The error bars depict the standard deviation



(a) Max label 4



(b) Max label 8



(c) Max label 16

Figure 5.6: The mean matching fraction of the images in dataset 1 obtained through the two-pass super-pixel algorithm for different maximum labels, compared to the Noisy Unwrapped Ground Truth. The error bars depict the standard deviation

5.4.4.1.2.1 Synthetic Dataset 2 (Pseudo-real) Results We experimented with the synthetic images in dataset 2 (10 images in total). All images are 400×400 pixels. We tried the one-pass super-pixel algorithm using 20×20 sub-images and the two-pass algorithm using 10×10 sub-images. In both cases, we used TRWS as the sub-image solver and compared it to the Deemed Noisy Unwrapped Ground Truth. We selected to compare to the Deemed Noisy Unwrapped Ground Truth because the Noisy Unwrapped Ground Truth does not exist for the real dataset. As we discussed earlier and present in Appendix D, the Deemed Noisy Unwrapped Ground Truth and the Noisy Unwrapped Ground Truth images are almost identical. We summarized the results in Figure 5.7.

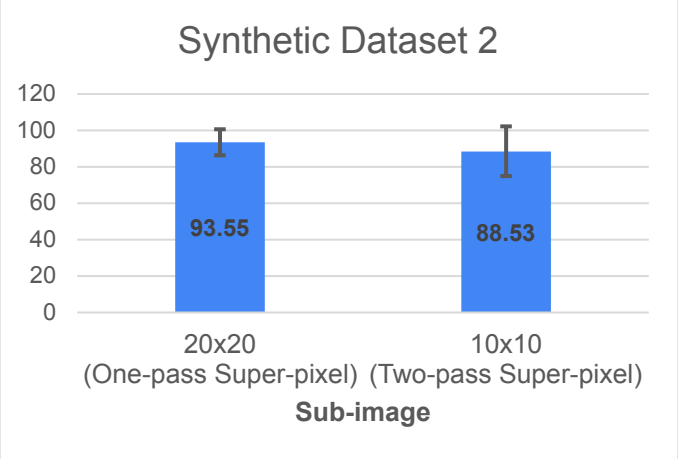


Figure 5.7: The mean matching fraction of the images in dataset 2, compared to the Deemed Noisy Unwrapped Ground Truth. The error bars depict the standard deviation

5.4.4.1.2.2 Real Images Results We experimented with the real images dataset (9 images in total). All images are 400×400 pixels. We tried the one-pass super-pixel algorithm using 20×20 sub-images and the two-pass algorithm using 10×10 sub-images. In both cases, we used TRWS as the sub-image solver and compared the results to the Deemed Noisy Unwrapped Ground Truth (as it is the only available ground truth). We summarized the results in Figure 5.8.

5.4.4.1.2.3 InSAR Images Samples Figure 5.9 and 5.10 show samples of the unwrapped images using the super-pixel method. The two figures show how close the unwrapped images to the ground truth.

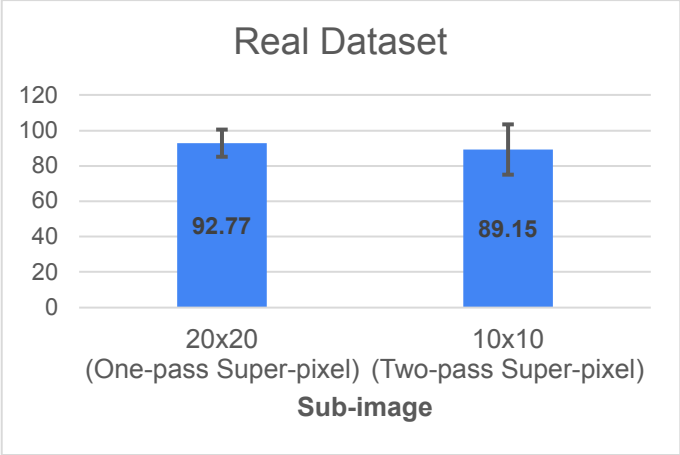


Figure 5.8: The mean matching fraction of the images in the real dataset, compared to the Deemed Noisy Unwrapped Ground Truth. The error bars depict the standard deviation

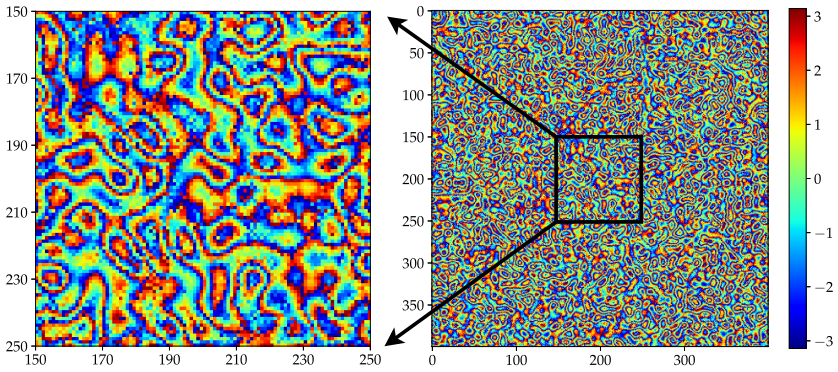
5.4.4.1.2.4 Discussion The results obtained with real images are comparable to those of the synthetic images (pseudo-real) in dataset 2. Both data sets produce results with a matching fraction within 1% of each other.

The results obtained are congruent with the results obtained in Section 5.4.4.1.1 above. The images analyzed in this section have a much higher noise content (SNR of 3db) as compared to the images in dataset 1 (SNR of 7db for high noise images). As we discussed in Section 5.4.4.1.1, noise and complexity contribute to the rapid deterioration of the quality of the solution. Using a larger sub-image gives better results.

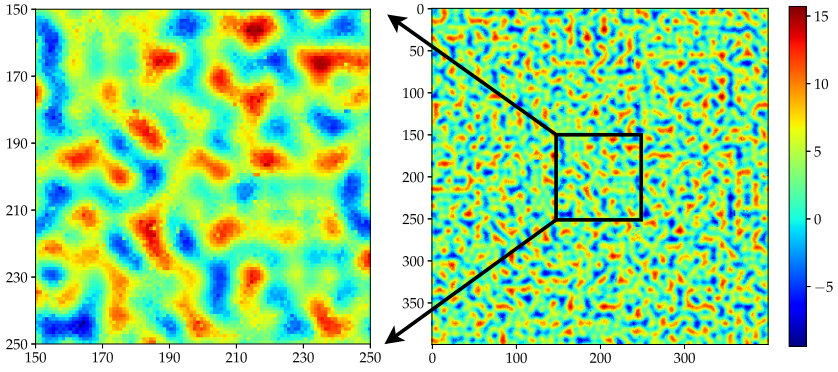
Compared to the synthetic images' results (synthetic dataset 2, Figure 5.7), the real images (Figure 5.8) show very close average matching (less than 1% difference).

5.4.4.2 QUBO as Sub-image Solver

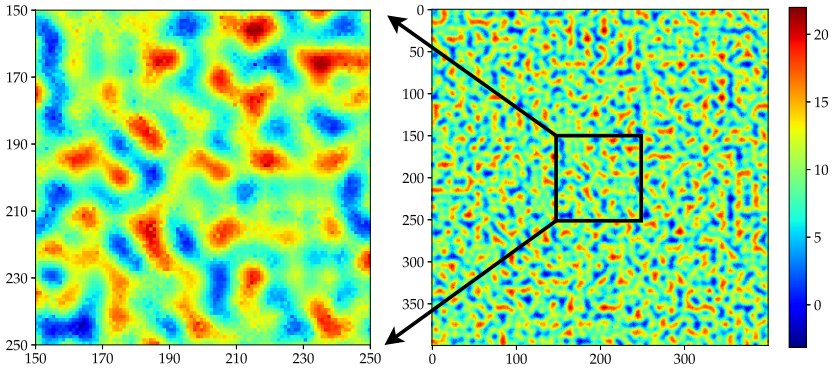
The primary focus of the experimentation is to study the applicability of QUBO solvers to our approach and to compare the relative performance of these solvers. As we discussed in Section 5.4.3.2, and given that the D-Wave quantum annealer cannot accommodate very large problems, we have elected to experiment with our two-phase super-pixel method with sub-images that do not exceed 10×10 pixels in size, and a maximum label of 4. We apply only with the two-pass super-pixel because of the limited access we have to the QUBO solvers. That is, we use large images of 120×120 pixels with sub-images of maximum 10×10.



(a) Wrapped simulated image

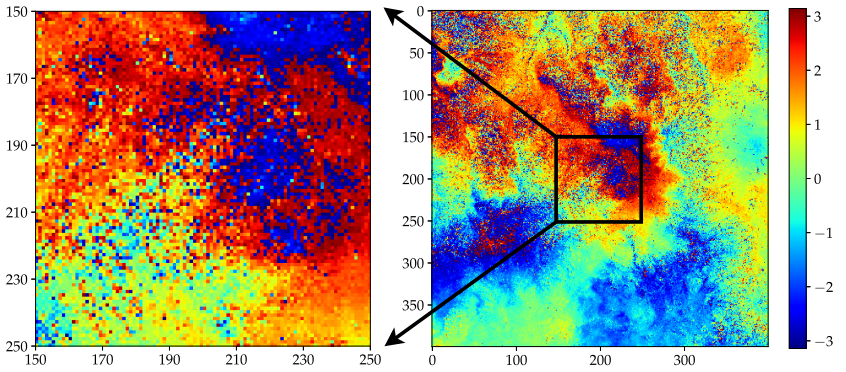


(b) Unwrapped simulated image

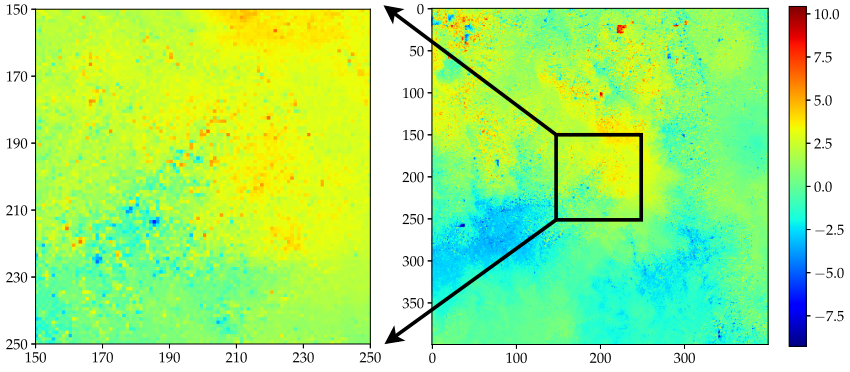


(c) Simulated image ground truth

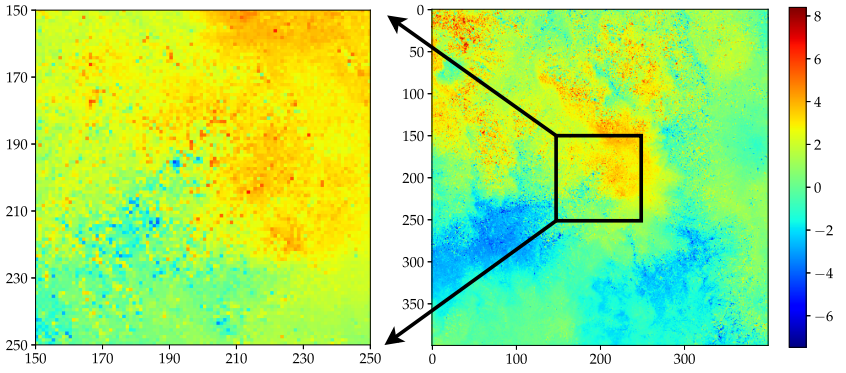
Figure 5.9: Samples from the images unwrapped using the super-pixel approach. The figure shows the unwrapping of a synthetic high-noise image. Top: the wrapped images; middle: the unwrapped images; bottom: the ground truth images to compare against (the Noisy Unwrapped Ground Truth)



(a) Wrapped real image



(b) Unwrapped real image



(c) Real image ground truth

Figure 5.10: Samples from the images unwrapped using the super-pixel approach. The figure shows the unwrapping of a real image. Top: the wrapped images; middle: the unwrapped images; bottom: the ground truth images to compare against (the Deemed Noisy Unwrapped Ground Truth)

Table 5.5: The cases considered for experiment set 1 (testing a synthetic image)

Case	Max label	Complexity	Noise	Size	Sub-image size
0	4	High	High	120×120	6×6
1	4	High	High	120×120	8×8
2	4	High	High	120×120	10×10

Given the limited capacity, in term of qubits, of the QUBO solvers, the QUBO solvers are configured to have maximum label of 4 (see Section 5.4.3.2). However, not all the images in the different datasets are limited by a maximum label of 4. In this experiment set we consider only the images that have a maximum label of 4 to study the applicability of QUBO solvers to the super-pixel algorithm. In an extension of this experiment (see Appendix F), we have also succeeded in unwrapping images with maximum labels larger than 4 and using the same solvers as with the case of maximum labels of 4 or less.

In the experiments reported here, we gradually increase the sub-image size to study its effect on the quality of the unwrapping.

In the following experiments, we used one synthetic InSAR image and one real InSAR image to experiment with. We cropped the images at random to a size of 120×120 pixels instead of the original large size (400×400). For this experiment set (and for the experiment in Section 5.4.5.2), we are experimenting with the same two images, the synthetic and the real, with a maximum label of 4. We selected this maximum label so that the image could be fully accommodated by the various QUBO solvers we experimented with. We varied each experiment's configuration, i.e., the sub-image size, the margin size, and the solver we used.

5.4.4.2.1 Synthetic Dataset 1 Results We selected the image randomly from the already generated dataset (synthetic dataset 1). Since we had to select a single image, we selected an image of high complexity and high noise. If the approach is giving good results for the highly complex and noisy image, it is more likely to give good results for images with lower complexity and less noise.

We assess the quality of the unwrapping as expressed by the matching fraction to the Noisy Unwrapped Ground Truth. Table 5.5 shows the cases we considered. All cases use the same image, and they are distinguished by the sizes of the sub-images we have employed.

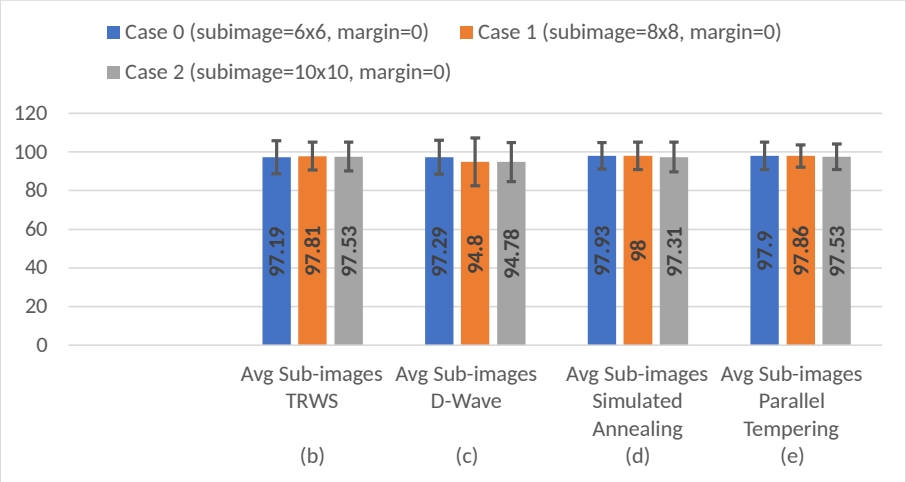


Figure 5.11: The average sub-image matching fraction for a synthetic image for the different sub-image sizes and different solvers (TRWS, D-Wave, Simulated annealing, and parallel tempering)

5.4.4.2.1.1 QUBO Results Figure 5.11 and Figure 5.12 summarize the results we obtained for different sub-image sizes and different solvers (TRWS, D-Wave, Simulated annealing, and parallel tempering). We are reporting two results. The first (reported in Figure 5.11) is the average of the matching fraction of the unwrapped sub-images while the second result (reported in Figure 5.12) is the matching fraction of the whole (120×120 pixel) image. Given that all the sub-images (in each case) are of the same size, we have used the arithmetic mean to calculate the average of the matching fraction we report for the first set of results.

We follow the same format in presenting the matching fractions for the several other experiments we report in the subsequent sections.

5.4.4.2.1.2 TRWS Results To ensure that the choice of the image we used in our experiment 1 was not biasing the results, we processed 10 randomly chosen 120×120 pixel sections from the synthetic images dataset 1 having the same complexity and noise as the section we used in experiment 1 and unwrapped these sections using our super-pixel approach and TRWS as the sub-image solver. As it can be verified by comparing Figure 5.12-b and Figure 5.13-b, the results of the chosen image (Figure 5.12-b) are within one standard deviation of the mean of the results of all the images processed (Figure 5.13-b)

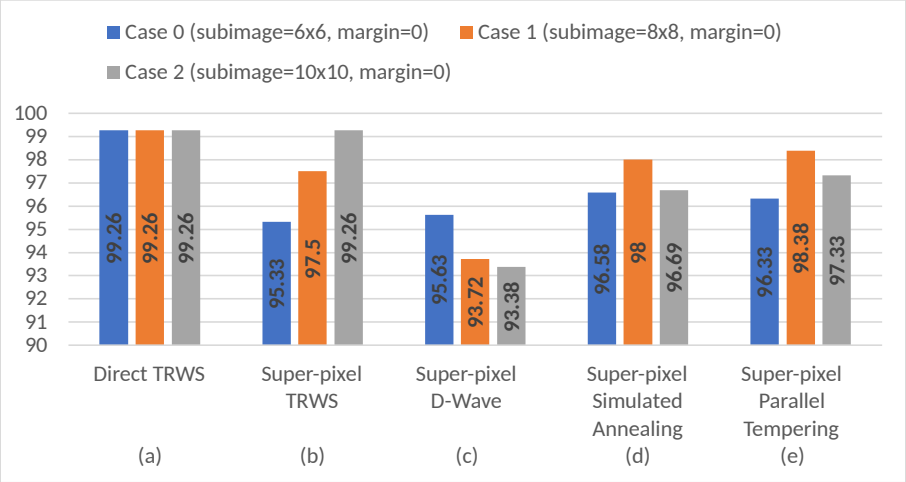


Figure 5.12: The super-pixel matching fraction for a synthetic image for the different sub-image sizes and different solvers (TRWS, D-Wave, Simulated annealing, and parallel tempering)

5.4.4.2.2 Real Dataset Results We selected an image randomly from the real images dataset with maximum label of 4 (i.e., Mexico City, Las Vegas 0, and Las vegas 2) . We cropped the image at random to a size of 120×120 pixels instead of the original large size (400×400). We assessed the quality of the unwrapping as expressed by the matching fraction to the Deemed Noisy Unwrapped Ground Truth. Table 5.6 shows the cases we considered. All cases use the same image, and they are distinguished by the sizes of the sub-images we have employed.

Case	Max label	Size	Sub-image size
0	4	120×120	6×6
1	4	120×120	8×8
2	4	120×120	10×10

Table 5.6: The cases considered for experiment set 1 (testing a real image)

5.4.4.2.2.1 QUBO Results Figure 5.14 and Figure 5.15 summarize the results we obtained for different sub-image sizes and different solvers (TRWS, D-Wave, Simulated annealing, and parallel tempering). We are reporting two results. The first (reported in Figure 5.14) is the average of the matching fraction of the unwrapped sub-images while the second result (reported in Figure 5.15) is the matching fraction of the whole (120×120 pixel) image.

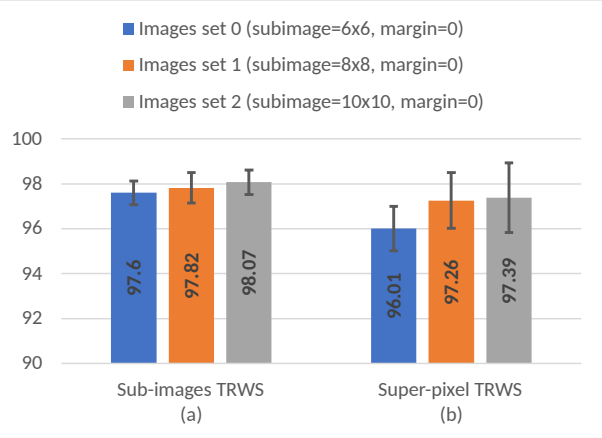


Figure 5.13: The average matching for 10 randomly chosen 120×120 pixel sections from the synthetic images dataset 1 having the same complexity and noise as the section we used in experiment set 1

5.4.4.2.2.2 TRWS Results To ensure that the choice of the image we used in this experiment was not biasing the results, we processed 5 120×120 pixel sections from the real images dataset having the same maximum label as the section we used in this experiment and unwrapped these sections using our super-pixel approach and TRWS as the sub-image solver. As it can be verified by comparing Figure 5.15-b and Figure 5.16-b, the results of the chosen image (Figure 5.15-b) are within one standard deviation of the mean of the results of all the images processed (Figure 5.16-b).

5.4.4.2.3 Discussion As presented earlier, using TRWS as a sub-image solver for the super-pixel algorithm gives results that are close to the ground truth. In this experiment, the QUBO solvers, the MQIO solvers and the D-Wave annealer, provide close results to the TRWS results when used as sub-images solvers. Particularly close are the results obtained by the MQIO solvers (Simulated annealing and Parallel Tampering) while the ones from the D-Wave annealer have a slightly lower matching fraction (about 3 percentage points) as compared to the matching fractions obtained by the TRWS, and the MQIO solvers.

The one and the multi-pass super-pixel algorithm can be extended to use various phase unwrapping solvers; this includes the classical TRWS and the QUBO based solvers (the D-Wave annealer and the MQIO solvers). However, each sub-image solver gives better results using specific configurations. Most of the QUBO solvers impose limitations on the number of pixels and the image's size.

For D-Wave, the sub-image size 6x6 has the best results for both; the average sub-

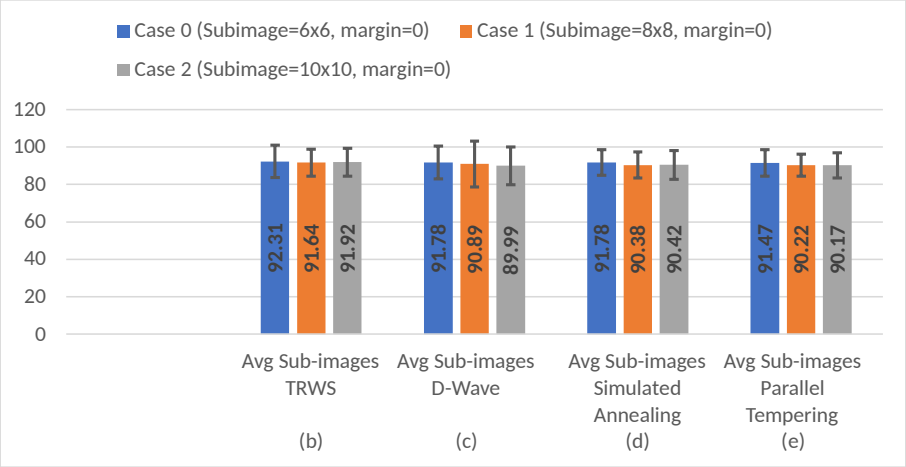


Figure 5.14: The average sub-image matching fraction for a real image for the different sub-image sizes and different solvers (TRWS, D-Wave, Simulated annealing, and parallel tempering)

images matching and the super-pixel solution. This is because the D-Wave annealer works better with smaller images (smaller images mean fewer qubits and better qubits coherence!).

While for the MQIO solvers, the sub-image size 8x8 has the best results for synthetic image and the sub-image size 6x6 has the best results for the real image. For the MQIO solvers, the bigger the sub-image, the less accurate the results because the search space for the optimal solution is bigger.

5.4.5 Evaluation of the Addition of Margins to the Sub-images

In this section, we evaluate the impact of adding a margin to the sub-images during the phases of the super-pixel algorithm. As we discussed in Section 5.3.3, including a margin, provides additional constraints to the pixels at the boundaries of the sub-images, and thus better informs the optimization process resulting in improved unwrapping quality. In the subsequent sections, we shall evaluate our approach first employing TRWS as the solver and then then a number if QUBO solvers.

5.4.5.1 TRWS as Sub-image Solver

In this part of the work, we are analyzing the impact of introducing a margin around the sub-images to the quality of the solution.

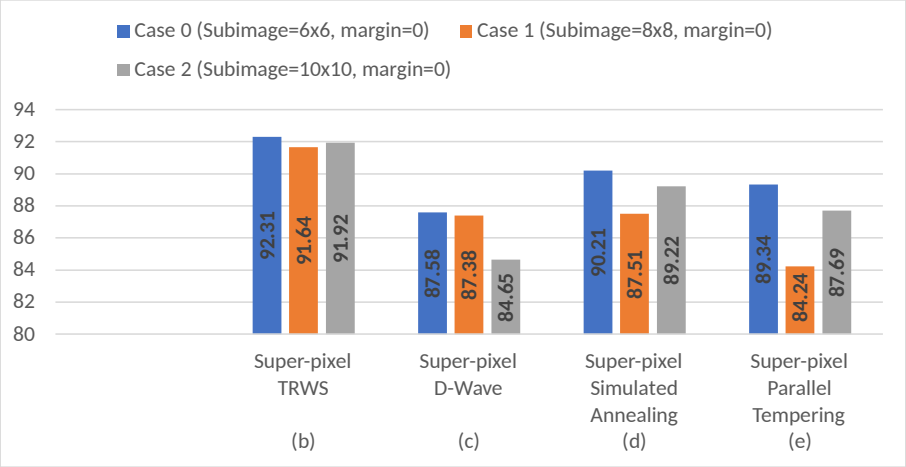


Figure 5.15: The super-pixel matching fraction for a real image for the different sub-image sizes and different solvers (TRWS, D-Wave, Simulated annealing, and parallel tempering)

5.4.5.1.1 One-pass Super-pixel Algorithm We have conducted two studies. For the first, we have kept the total size of the sub-image (i.e., the sub-image itself plus the margin) constant. The second study kept the size of the sub-image constant but gradually increased the size of the margin. In particular, the sizes of the sub-images and the margins used in these studies are summarized in Table 5.7. The results are presented in Figure 5.17 and Figure 5.18.

5.4.5.1.1.1 Discussion As it can be seen Figure 5.17 and Figure 5.18, adding a margin, in general, improves the quality of the unwrapping.

For the case where the total number of pixels in a sub-image is kept constant but the margin varies as shown in Figure 5.17, the quality of the unwrapping improves marginally as the margin increases. However, there is a lot of variability, and this can be explained since for higher margins, the size of the non-overlapping sub-image decreases, and as we have seen, smaller sub-images tend to result in poorer unwrapping quality. Smaller margins (1 and 2) seem to provide the maximum benefit, especially for the high-noise cases.

The case where the size of the non-overlapping sub-images is kept constant (in our case 20×20 pixels) but the size of the margin increased gradually, is depicted in Figure 5.18. In this figure, we plot both the cases where the sub-images have constant size but an increasing margin (solid lines) and the cases where the sub-images are increasing in size but have no margin (dashed lines). In both instances, the size of

Table 5.7: The sizes of the sub-images and the margins used in testing adding margin (one-pass super-pixel)

Case A0	20×20 margin of 0
Case A1	18×18 margin of 1
Case A2	16×16 margin of 2
Case A3	14×14 margin of 3
Case A4	12×12 margin of 4

Case B0	20×20 margin of 0 & 20×20 no margin
Case B1	20×20 margin of 1 & 22×22 no margin
Case B2	20×20 margin of 2 & 24×24 no margin
Case B3	20×20 margin of 3 & 26×26 no margin
Case B4	20×20 margin of 4 & 28×28 no margin
Case B5	20×20 margin of 5 & 30×30 no margin

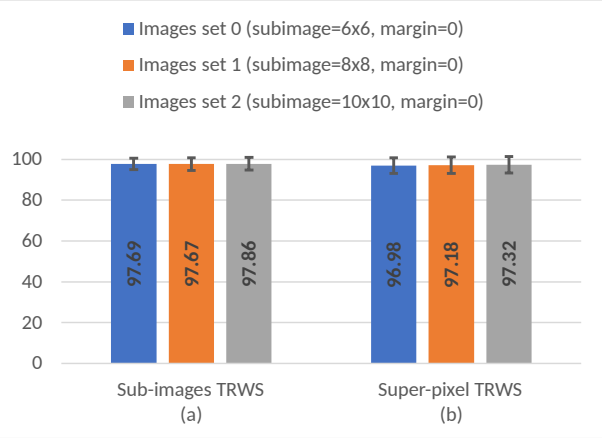


Figure 5.16: The average matching for 5 120×120 pixel sections from the real images dataset having the same maximum label as the section we used in experiment set 1 for real images

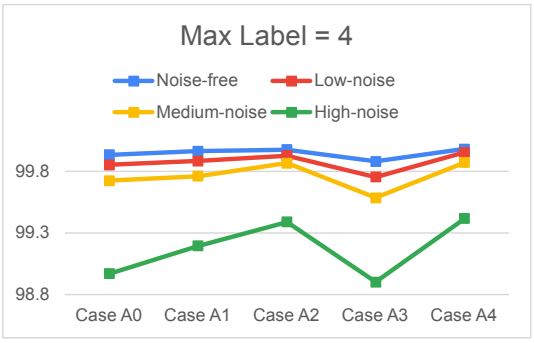
the sub-image is the same and the actual sizes we used are shown in Table 5.7. As this figure demonstrates, adding a margin results in a better quality unwrapping (the solid lines are "above" the dashed lines). As per the previous observation, the higher the noise, the larger the benefit.

5.4.5.1.2 Two-pass Super-pixel Algorithm We have conducted similar studies to the to two studies discussed above in Section 5.4.5.1.1. The sizes of the sub-images and the margins used in these studies are summarized in Table 5.8. The results are presented in Figure 5.19 and Figure 5.20.

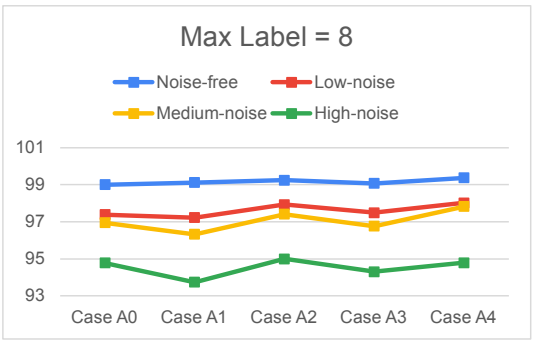
5.4.5.1.2.1 Discussion For Figure 5.20, adding a margin, in general, improves the quality of the unwrapping. However, this is not the case for Figure 5.19.

For the case where the total number of pixels in a sub-image is kept constant but the margin varies as shown in Figure 5.19, the quality of the unwrapping does not improve as the margin increases. There are two factors that affect the quality of the results, as we discussed in Section 5.4.5.1.1.1; the margin size which while increasing improves the quality of the solution, and the sub-image size which when decreased results in poorer unwrapping quality. In the case of the sub-image of 10×10, the sub-image size is the dominating effect. Hence, we do not notice improvement when we increase the margin while decreasing the sub-image size.

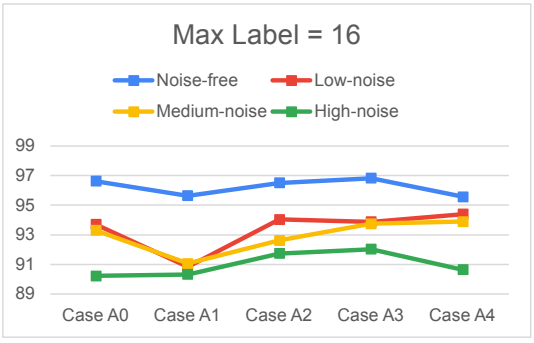
The case where the size of the non-overlapping sub-images is kept constant (in our case 10×10 pixels) but the size of the margin increases gradually, is depicted in



(a) Max label 4

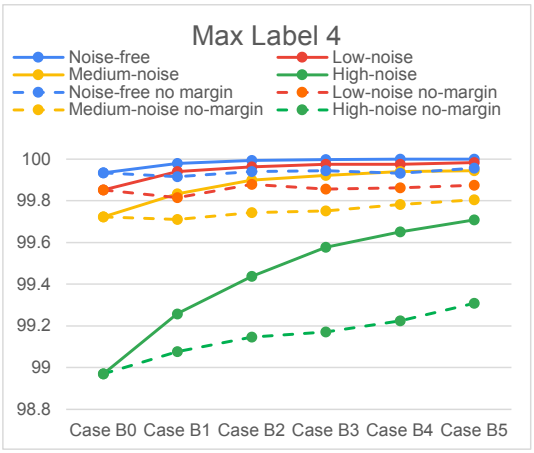


(b) Max label 8

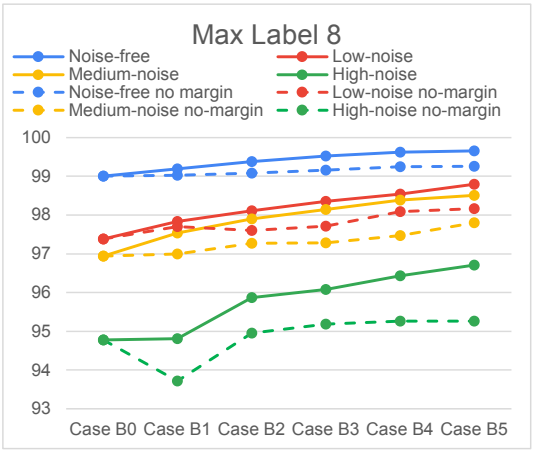


(c) Max label 16

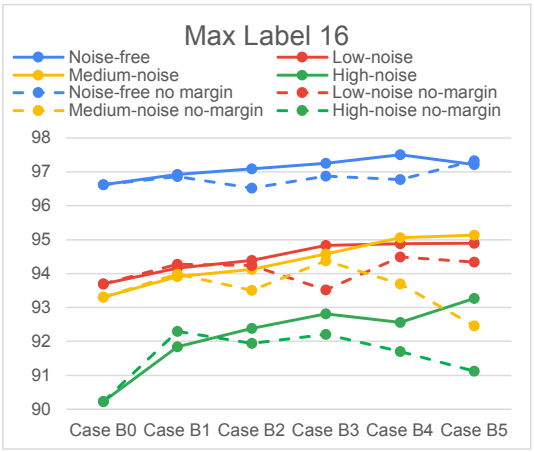
Figure 5.17: The matching fraction vs. the margin when the total sub-image image size A'_x is fixed at 20×20 pixels



(a) Max label 4



(b) Max label 8



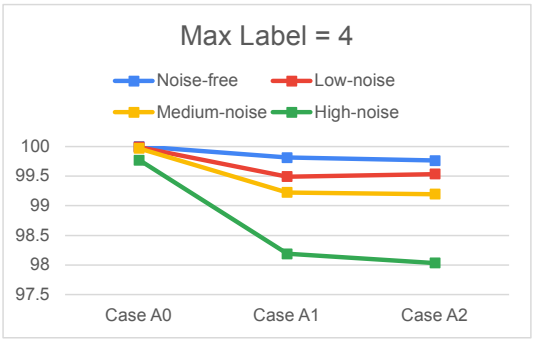
(c) Max label 16

Figure 5.18: The matching fraction vs. the margin for the sub-image size A_x of 20×20 pixels

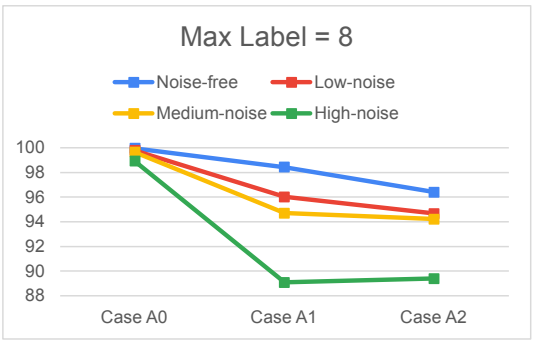
Table 5.8: The sizes of the sub-images and the margins used in testing adding margin (two-pass super-pixel)

Case A0	10×10 margin of 0
Case A1	8×8 margin of 1
Case A2	6×6 margin of 2

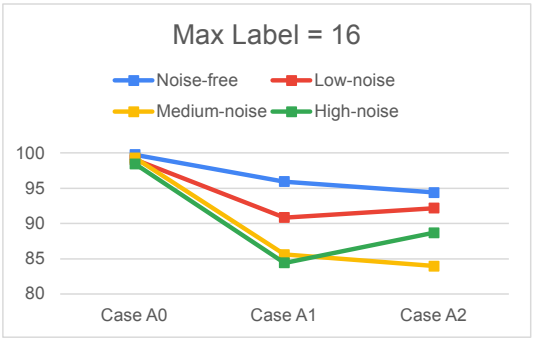
Case B0	10×10 margin of 0 & 10×10 no margin
Case B1	10×10 margin of 1 & 12×12 no margin
Case B2	10×10 margin of 2 & 14×14 no margin
Case B3	10×10 margin of 3 & 16×16 no margin
Case B4	10×10 margin of 4 & 18×18 no margin
Case B5	10×10 margin of 5 & 20×20 no margin



(a) Max label 4

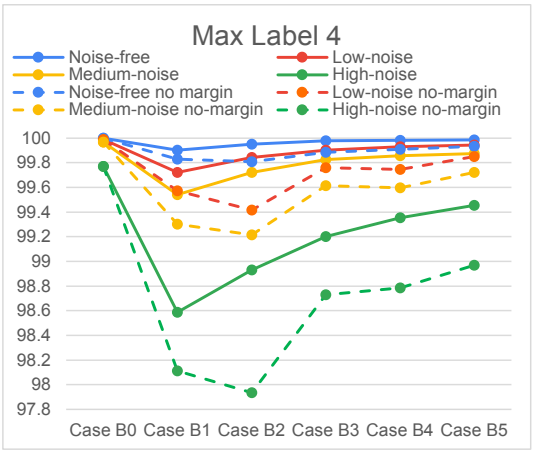


(b) Max label 8

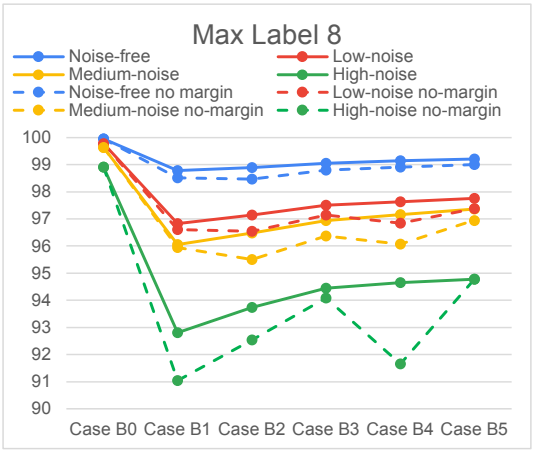


(c) Max label 16

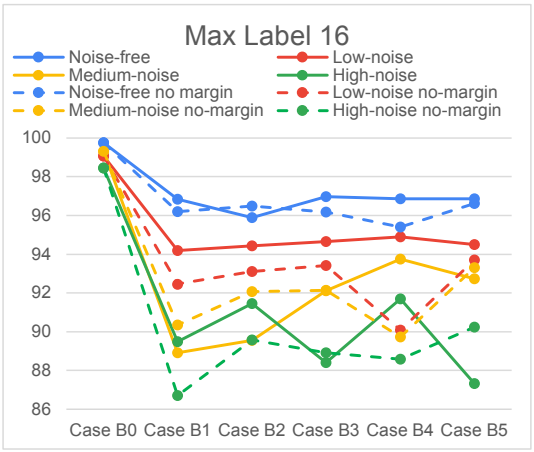
Figure 5.19: The matching fraction vs. the margin when the total sub-image image size A'_x is fixed at 10×10 pixels



(a) Max label 4



(b) Max label 8



(c) Max label 16

Figure 5.20: The matching fraction vs. the margin for the sub-image size A_x of 10×10 pixels

Figure 5.20. As this figure demonstrates adding a margin results in a better quality unwrapping (the solid lines are "above" the dashed lines).

5.4.5.2 QUBO as Sub-image Solver

Having established the effectiveness of the adding margin, we now study the effectiveness of the various QUBO solvers. Since we had limited access to the QUBO solvers, we have elected to use one of the images from our synthetic images database and one image from the real images dataset (the images selected in previously for the experiments Section 5.4.4.2.1) and do a comparative analysis of the quality of the solutions achieved by the different QUBO solvers. The synthetic image was selected at random from the set of high-complexity, high-noise images and the real image is selected at random from the real image that have maximum label of 4. However, the maximum number of labels was restricted to four because of the limited problem size that the QUBO solvers, especially the D-Wave 2000Q_6, can accommodate.

Further, to reduce the number of computations required we randomly cropped the selected image to a size of 120×120 pixels. The sub-image sizes we used for our study are 6×6, 8×8, and 10×10. We have also included the results of using TRWS as the solver to compare to the QUBO solvers' results. However, since we have no restrictions on using TRWS, we experimented with all the images that have the same characteristics (same complexity, noise level, and maximum labels). To derive the matching fractions, we compare the results to the Noisy Unwrapped Ground Truth for the synthetic image and the Deemed Noisy Unwrapped Ground Truth for the real image.

As per the analysis presented in Section 5.4.5.1, we would expect that, in general, adding a margin to the sub-images would result in increased unwrapping quality. In this experiment, we explore the impact of the particular solver employed.

As it is shown in the following subsections, the use of QUBO solvers produces results that are comparable to those of produced using TRWS. In addition, adding a margin, in many cases improves the quality of the solution. However, the size of the sub-images plays a significant role in the obtained quality.

5.4.5.2.1 Fixing Sub-image Size and Increasing the Margin Table 5.9 presents the cases considered for the experiment with the synthetic and the real image.

Table 5.9: The cases considered for adding margin with QUBO solvers experiment

Case	Max label	Size	Sub-image size	Margin	Sub-image total size
0	4	120×120	6×6	0 - 1	6×6 - 8×8
1	4	120×120	8×8	0 - 1	8×8 - 10×10
2	4	120×120	10×10	0 - 1	10×10 - 12×12

5.4.5.2.1.1 Synthetic Image Results Table 5.10 presents the results of experimenting with the synthetic image.

Table 5.10: The results of adding margin with QUBO solvers using a synthetic image

	Sub-image	Margin	Super-pixel			
			TRWS	D-Wave	Simulated Annealing	Parallel Tempering
Case 0	6×6	0	95.33	95.63	96.58	96.33
		1	96.68	94.07	96.81	98.34
Case 1	8×8	0	97.5	93.72	98	98.38
		1	97.32	91.88	98.01	98.05
Case 2	10×10	0	96.53	93.38	96.69	97.33
		1	97.77	83.33	97.85	98

Sub-image size is a critical parameter to select. The larger the sub-image, the less the influence of the uncertainty due to the boundaries, and hence the better the quality of the solution. However, this is not observed for the D-Wave solver. For D-Wave, selecting a smaller sub-image size is preferable. A smaller sub-image means more coherent qubits and a better solution. This can be seen in all cases of the D-Wave results. As the size of the sub-image processed increases, the quality of the results worsens.

Introducing a margin, generally improves the quality of the results as was analyzed in Section 5.4.5.1. This can be ascertained if we compare the cases with a margin and the cases without margin but with the same size of sub-image. That is sub-images of size 6×6 with margin of 1 and images of size 8×8 with margin of 0. In all cases, but for the Simulated Annealing solver, the introduction of a margin, improves the quality of the solution.

As the size of the sub-image increases, the quality of the solution increases and the

introduction of a margin plays a less critical role in the improvement of the quality. This can be seen for the sub-images with size 8×8 with margin of 1 and sub-images with size 10×10 and margin of 0. All have a higher quality solution as compared to the cases with the smaller sub-images. For the D-Wave and the Simulated Annealing solvers, the quality of the solutions improved as a margin was introduced, while for the other two solvers, the quality deteriorated marginally.

5.4.5.2.1.2 Real Image Results Table 5.11 presents the results of experimenting with the real image.

Table 5.11: The results of experimenting adding margin with QUBO solvers using a real image

	Sub-image	Margin	Super-pixel			
			TRWS	D-Wave	Simulated Annealing	Parallel Tempering
Case 0	6×6	0	92.31	87.58	90.21	89.34
		1	94.51	86.58	89.62	84.57
Case 1	8×8	0	91.64	87.38	87.51	84.24
		1	93.85	90.84	85.8	86.82
Case 2	10×10	0	91.92	84.65	89.22	87.69
		1	93.32	88.32	89.52	88.7

With the real image, we can reach a conclusion similar to the one discussed above, adding margin improves the quality of the solution. However, because of the single image case and the high noise, we cannot draw sufficient statistically robust conclusions.

5.4.5.2.2 Constant Total Size of the Sub-image Here, we increase the margin while keeping the total sub-image size constant. Table 5.12 summarizes the properties of the images.

5.4.5.2.2.1 Synthetic Image Results Table 5.13 summarizes the super-pixel matching fraction for the different sub-image sizes and different solvers (TRWS, D-Wave, Simulated annealing, and parallel tempering) for the synthetic image.

Table 5.12: The cases considered for adding margin while fixing the total sub-image size with QUBO solvers

Case	Max label	Size	Sub-image size	Margin	Sub-image total size
0	4	120×120	6×6	2	10×10
1	4	120×120	8×8	1	10×10
2	4	120×120	10×10	0	10×10

Table 5.13: The results of adding margin while fixing the total sub-image size with QUBO solvers using a synthetic image

	Sub-image	Margin	Super-pixel			
			TRWS	D-Wave	Simulated Annealing	Parallel Tempering
Case 0	6×6	2	97.49	91.97	97.31	97.67
Case 1	8×8	1	97.32	91.88	98.01	98.05
Case 2	10×10	0	96.53	93.38	96.69	97.33

5.4.5.2.2.2 Real Image Results Table 5.14 summarizes the super-pixel matching fraction for the different sub-image sizes and different solvers (TRWS, D-Wave, Simulated annealing, and parallel tempering) for the real image.

Table 5.14: The results of adding margin while fixing the total sub-image size with QUBO solvers using a real image

	Sub-image	Margin	Super-pixel			
			TRWS	D-Wave	Simulated Annealing	Parallel Tempering
Case 0	6×6	2	95.38	90.15	88.36	89.58
Case 1	8×8	1	93.85	90.84	85.8	86.82
Case 2	10×10	0	91.92	84.65	89.22	87.69

5.4.5.2.2.3 Discussion The total sub-image size (sub-image size plus the margin) is equal for all three images. Hence, and as expected, the D-Wave annealer gave close matching fraction for the three cases, except for one of the cases of

the real image. A slight variation, about 1.5%, is expected from the annealer. This sub-image size seems to be the main factor, if not the only factor, that affects the results obtained from applying the super-pixel algorithm, regardless of how much of the sub-image is included in the final solution and how much is considered as margin. However, the trend is different. TRWS and the other QUBO algorithms showed that including a margin improves the results. For the D-Wave and the synthetic image, we have the trend for case 0 versus case 1 (case 0 is marginally better than case 1); however, case 2 is markedly better than case 0 and case1. While for the real image, case 1 is markedly better than case 2 and marginally better than case 0. This is an indication of different optimal margins depending on the noise, and also because of the single image case, we cannot draw sufficient statistically robust conclusions.

The Parallel Tempering solver gives slightly better results compared to the Simulated Annealing solver. The three images' super-pixel solutions are very close to each other (less than 1.5% matching difference) because the three images have a similar total sub-image size (10×10).

5.5 Chapter Summary

In this work, we extended and refined the work we presented in [54]. We have successfully demonstrated that the InSAR phase unwrapping problem can be expressed and solved as a QUBO problem. By partitioning the problem, we have been able to obtain high-quality solutions for large images. We refer to this method as "single-pass super-pixel decomposition". We extended the concept of the super-pixel decomposition to handle bigger images by recursively partitioning the image until the partitions fit in the quantum annealer. Then recursively apply the second phase of the superposition until we obtain labels that approach the global solution "multi-pass super-pixel decomposition". We introduced the notion of a margin in processing sub-images in our two-phase super-pixel unwrapping method. Margins enlarge the size of the sub-images to be processed and result in overlapping sub-images. The pixels in the margins provide additional information in the unwrapping process, and this results in increased accuracy of the resulting unwrapped image.

We have tested our approach on various software-implemented QUBO solvers and the D-Wave 2000Q annealer, and for a variety of synthetic and real images. The solutions derived by the single-pass super-pixel are either identical to the ground truth or have less than 5% of pixels differing from the ground truth. The results of

the multi-pass super-pixel showed it can handle bigger images while preserving the quality of the solution. The accuracy of the results depends on the specific annealer employed, the amount of noise, and the complexity of the images. Both the one-pass or two-pass super-pixel algorithms produce excellent results for low and medium complexity images. Real SAR and InSAR images are typically of low or medium complexity. Our approach produces acceptable results for high complexity images. Moreover, our experiments have shown that introducing margins indeed improves the quality of the resulting unwrapped images for both single and multi-pass super-pixel. Our experiments have shown that for the same size of a sub-image, using margins results in better quality as compared to the case where a margin was not used. Software-implemented QUBO solvers showed slightly better results compared to the D-Wave 2000Q annealer. However, both achieved state-of-the-art solution quality.

Chapter 6

Conclusions

6.1 Summary

This thesis studied employing machine learning, using Convolutional Neural Networks, and quantum computing in improving the InSAR processing.

6.1.1 Employing Machine Learning in Improving InSAR Processing

6.1.1.1 Scale-Invariant Super-Resolution (SINV)

We proposed SINV as a Super-Resolution solution in the absence of ground truth. This CNN-based super-resolution module is critical for some applications where the ground truth is not available or when the processing power in the training phase is limited. Performance evaluation over several datasets shows the proposed method provides better numerical and visual results than Bicubic interpolation for upscaling factors of 2 and 4. In addition, our experiments show that SINV speeds up the training process by up to 400%.

6.1.1.2 Co-registration Enhancement Through CNN

We introduced a new method for enhancing the co-registration process and the Interferometric SAR generation using learning-based super-resolution based on SINV. The results show that CNN SR can enhance SAR imaging at different processing levels, at the co-registration stage and the InSAR image generation. The proposed work is the first to incorporate learning based super-resolution in enhancing SAR image.

6.1.2 Employing Quantum Computing in Improving InSAR Processing

6.1.2.1 Super-pixel Decomposition

We have successfully demonstrated that the InSAR phase unwrapping problem can be expressed and solved as a QUBO problem. This is achieved by formulating the problem as a QUBO problem, which can be solved on a quantum annealer. However, given that present embodiments of quantum annealers remain limited in the number of qubits they possess, we decompose the problem into a set of subproblems that can be solved individually. By partitioning the problem, we have been able to obtain high-quality solutions for large images. We refer to this method as "single-pass super-pixel decomposition". We extended the concept of the super-pixel decomposition to handle larger images by recursively partitioning the image until the partitions fit in the quantum annealer. Then recursively apply the second phase of the super-position until we obtain labels that approach the global solution "multi-pass super-pixel decomposition".

We introduced the notion of a margin in processing sub-images in our two-phase super-pixel unwrapping method. Margins enlarge the size of the sub-images to be processed and result in overlapping sub-images. The pixels in the margins provide additional information in the unwrapping process, and this results in increased accuracy of the resulting unwrapped image. Moreover, our experiments have shown that introducing margins improves the quality of the resulting unwrapped images for single and multi-pass super-pixels. Our experiments have shown that using margins results in better quality for the same size of a sub-image than the case where a margin was not used.

6.1.2.2 Mapping of the Problem to the Chimera Network

We targeted the D-Wave implementation of the quantum annealer (D-Wave 2000Q) to test our super-pixel methodology. For that purpose, we developed a new way to map the phase unwrapping problem on the Chimera network, the topology chosen by D-Wave for the annealer. This mapping, based on our experiments, outperformed the heuristic approaches provided by D-Wave for mapping. In addition, the mapping enables the utilization of most of the qubits in the annealer with a symmetric mapping and reduces the probability of the qubits' decoherence by placing the relevant qubits

spatially closer.

6.1.2.3 The Promising Results

We have tested our approach on various software-implemented QUBO solvers and the D-Wave 2000Q annealer, and for a variety of synthetic and real images. The solutions derived by the single-pass super-pixel are either identical to the ground truth or have less than 5% of pixels differing from the ground truth. The results of the multi-pass super-pixel showed it could handle bigger images while preserving the quality of the solution. The accuracy of the results depends on the specific annealer employed, the amount of noise, and the complexity of the images. Both the one-pass or two-pass super-pixel algorithms produce excellent results for low and medium-complexity images. Real SAR and InSAR images are typically of low or medium-complexity. Our approach produces acceptable results for high-complexity images.

6.2 Future Work

This section presents some ideas and suggestions for future work.

1. **Fully Qualify the Time Complexity of the SINV Training Process:** In Chapter 3, we presented some results for the SINV network. However, since the SAR/InSAR images are relatively large ($30k \times 20k$ pixels), the training process takes an extensively long time. The large images are partitioned during the training process. However, since the images are large, they result in many partitions. Given the limited processing power available at the time, we could not explore most of the network parameters. Hence, we could not fine-tune the network properly nor fully quantify the time complexity of the SINV network. In addition, we trained the network using a limited number of images (10 large SAR images) which might affect the network generalization.

2. **Incorporate Other Methods for InSAR Images Evaluation:** Throughout the study, we aimed to use coherence as an evaluation metric for the quality of InSAR images. This decision was driven by the fact that coherence is the most prevalent evaluation metric used in the literature. However, other studies suggested using other metrics to qualify the InSAR images. For instance, Li et al. [70] proposed using Sum of Phase Differences (SPD) and argued its effectiveness in qualifying the InSAR images.

3. **Quantum Speed-up:** The primary objective of this study is to examine the applicability of using a quantum annealer to solve the problem of phase unwrapping, and to provide a comparative review of the quantum annealer performance compared to the classical phase unwrapping solvers. To fulfil this goal, we strived with the limited number of available qubits, which forced us to develop the super-pixel method. However, this work can be extended by studying the speed up, if exists, of the quantum annealer over the classical solvers.

4. **Map to New Annealer:** Our study focused on utilizing D-Wave quantum annealer 2000Q_6, the annealer that was available when we started the study. D-Wave annealer 2000Q_6 comprises 2041 qubits and was released on January 24, 2017. In September 2020, D-Wave released its latest quantum annealer (called "Advantage") with over 5000 qubits. In this new annealer, the Chimera architecture is replaced with a new architecture known as Pegasus. The new quantum annealer is expected to accommodate larger subproblems, mitigating the errors introduced by the subproblems boundaries. In this thesis, we developed a new way to map the phase unwrapping problem on the Chimera network and we would like to extend these ideas to the Pegasus network. The most challenging aspect is finding an efficient mapping between the phase unwrapping QUBO formulation and the new quantum annealer architecture. The initial experiments of simply porting the mapping were not successful. Hence, more detailed study is needed.

5. **More Phase Unwrapping Applications:** In this thesis, we studied employing quantum annealers to solve the phase unwrapping problem with the aim of the InSAR application. However, the same concepts and methodologies are applicable for many other applications that rely on phase unwrapping [110]. These applications include wavefront distortion measurement in the field of adaptive optics, field mapping for magnetic resonance imaging (MRI), and precise profiling of mechanical parts using x-ray.

Bibliography

[1] Machine learning with scikit-learn.

[2] Dorit Aharonov, Wim Van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Review*, 50(4):755–787, 2008.

[3] Md Zahangir Alom, Tarek M Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C Van Essen, Abdul AS Awwal, and Vijayan K Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3):292, 2019.

[4] Ethem Alpaydin. *Introduction to Machine Learning*. MIT Press, 2009.

[5] Fabrizio Argenti, Alessandro Lapini, Tiziano Bianchi, and Luciano Alparone. A tutorial on speckle reduction in synthetic aperture radar images. *IEEE Geoscience and Remote Sensing Magazine*, 1(3):6–35, 2013.

[6] Stefan Auer and Richard Bamler. 3D analysis of trihedral reflection based on SAR simulation methods. In *8th European Conference on Synthetic Aperture Radar*, pages 1–4, 2010.

[7] Richard Bamler and Philipp Hartl. Synthetic aperture radar interferometry. *Inverse Problems*, 14(4):R1, 1998.

[8] Yoshua Bengio et al. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[9] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118, 2010.

[10] Jake Bouvrie. Notes on convolutional neural networks. November 2006.

[11] Jun Cai, William G Macready, and Aidan Roy. A practical heuristic for finding graph minors. *arXiv preprint arXiv:1406.2741*, 2014.

[12] F. M. Candocia and J. C. Principe. Super-resolution of images based on local correlations. *IEEE Trans. Neural Networks*, 10(2):372–380, March 1999.

[13] Curtis W Chen and Howard A Zebker. Phase unwrapping for large sar interferograms: Statistical segmentation and generalized network models. *IEEE Transactions on Geoscience and Remote Sensing*, 40(8):1709–1719, 2002.

[14] Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in Neural Information Processing Systems*, pages 2843–2851, 2012.

[15] Mario Costantini. A novel phase unwrapping method based on network programming. *IEEE Transactions on Geoscience and Remote Sensing*, 36(3):813–821, 1998.

[16] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Eötvös Loránd University, Hungary*, 24:48, 2001.

[17] John C. Curlander and Robert N. McDonough. *Synthetic Aperture Radar: Systems and Signal Processing*. Wiley series in remote sensing. Wiley, New York, 1991.

[18] Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*, pages 666–674, 2011.

[19] David Deutsch. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[20] Fred M Dickey, Louis A Romero, and Armin W Doerry. Superresolution and synthetic aperture radar. *Sandia Report, SAND2001-1532, Sandia National Laboratories, Albuquerque, NM*, 87185, 2001.

[21] Nikitas Dimopoulos. *Neural Networks From Biological Origins to Silicon*. 2014.

[22] Richard Durbin and David E Rumelhart. Product units: A computationally powerful and biologically plausible extension to backpropagation networks. *Neural Computation*, 1(1):133–142, 1989.

[23] Clément Farabet, Yann LeCun, Koray Kavukcuoglu, Eugenio Culurciello, Berin Martini, Polina Akselrod, and Selcuk Talay. Large-scale fpga-based convolutional networks. *Scaling up Machine Learning: Parallel and Distributed Approaches*, pages 399–419, 2011.

[24] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[25] A Ferretti, A Monti-Guarnieri, C Prati, F Rocca, and D Massonet. Insar principles-guidelines for sar interferometry processing and interpretation, tm-19. *The Netherlands: ESA Publications*, 2007.

[26] Aleta Berk Finnila, MA Gomez, C Sebenik, Catherine Stenson, and Jimmie D Doll. Quantum annealing: a new method for minimizing multidimensional functions. *Chemical Physics Letters*, 219(5-6):343–348, 1994.

[27] Giorgio Franceschetti and Riccardo Lanari. Synthetic aperture radar processing crc press. *Electronic Engineering Systems Series*, 1999.

[28] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988.

[29] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[30] Fred Glover, Gary Kochenberger, and Yu Du. A tutorial on formulating and using qubo models. *arXiv preprint arXiv:1811.11538*, 2018.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[32] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[33] Andrea Monti Guarnieri and Claudio Prati. Sar interferometry: a" quick and dirty" coherence estimator for data browsing. *IEEE Transactions on Geoscience and Remote Sensing*, 35(3):660–669, 1997.

[34] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

[35] Ramon Hanssen and Richard Bamler. Evaluation of interpolation kernels for sar interferometry. *IEEE Transactions on Geoscience and Remote Sensing*, 37(1):318–321, 1999.

[36] Khizar Hayat. Multimedia super-resolution via deep learning: A survey. *Digital Signal Processing*, 81:198–217, 2018.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, June 2016.

[38] Bettina Heim, Troels F Rønnow, Sergei V Isakov, and Matthias Troyer. Quantum versus classical annealing of ising spin glasses. *Science*, 348(6231):215–217, 2015.

[39] Khaled A Helal, Bardia Barabadi, Amirali Baniasadi, and Nikitas Dimopoulos. Scale invariant super-resolutions methods with application to InSAR images. In *2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, 2019.

[40] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29, 2012.

[41] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[42] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[43] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[44] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[45] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[46] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[47] Yizhen Huang and Yangjing Long. Super-resolution using neural networks based on the optimal recovery theory. *Journal of Computational Electronics*, 5(4):275–281, Dec 2006.

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

[49] M. Irani and S. Peleg. Super resolution from image sequences. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume ii, pages 115–120 vol.2, 1990.

[50] Kazuyoshi Itoh. Analysis of the phase unwrapping algorithm. *Applied Optics*, 21(14):2470–2470, 1982.

[51] Xiaoyi Jia, Xiangmin Xu, Bolun Cai, and Kailing Guo. Single image super-resolution using multi-scale convolutional neural network. In *Pacific Rim Conference on Multimedia*, pages 149–157. Springer, 2017.

[52] Mark W Johnson, Mohammad HS Amin, Suzanne Gildert, Trevor Lanting, Firas Hamze, Neil Dickson, Richard Harris, Andrew J Berkley, Jan Johansson, Paul Bunyk, et al. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, 2011.

[53] Ali Jooya, Babak Keshavarz, Nikitas Dimopoulos, and Jaspreet S Oberoi. Accelerating neural network ensemble learning using optimization and quantum annealing techniques. In *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*, pages 1–7, 2017.

[54] Khaled A. Helal Kelany, Nikitas Dimopoulos, Clemens P. J. Adolphs, Bardia Barabadi, and Amirali Baniasadi. Quantum annealing approaches to the phase-unwrapping problem in synthetic-aperture radar imaging. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 120–129, 2020.

[55] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Trans Acoust., Speech, Signal Processing*, 29(6):1153–1160, 1981.

[56] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516, 2020.

[57] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[58] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 1, 01 2009.

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[60] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[61] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[62] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recog-

nition with a back-propagation network. In *Advances in Neural Information Processing Systems*, pages 396–404, 1990.

[63] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[64] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256. IEEE, 2010.

[65] Christian Ledig, Lucas Theis, Ferenc HuszÃ¡r, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017.

[66] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial Intelligence and Statistics*, pages 464–472, 2016.

[67] Fuk K Li and Richard M Goldstein. Studies of multibaseline spaceborne interferometric synthetic aperture radars. *IEEE Transactions on Geoscience and Remote Sensing*, 28(1):88–97, 1990.

[68] Zhengxiao Li, James Bethel, et al. Image coregistration in sar interferometry. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37:433–438, 2008.

[69] Zhilin Li. Orthoimage generation and measurement from single images. In *Geographical Data Acquisition*, pages 173–192. Springer, 2001.

[70] Zhilin Li, Weibao Zou, Xiaoli Ding, Yongqi Chen, and Guoxiang Liu. A quantitative measure for the quality of insar interferograms based on phase differences. *Photogrammetric Engineering & Remote Sensing*, 70(10):1131–1137, 2004.

[71] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[72] Qian Lin, John F Vesecky, and Howard A Zebker. New approaches in interferometric sar data processing. *IEEE Transactions on Geoscience and Remote Sensing*, 30(3):560–567, 1992.

[73] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[74] Carlos Lopez-Martinez, Xavier Fabregas, Eric Pottier, and Equipe Imagerie Radar-Teledetection Polarimetrie. A new alternative for sar imagery coherence estimation. In *Proceedings of the 5th European Conference on Synthetic Aperture Radar (EUSAR'04), Ulm, Germany*, pages 25–27, 2004.

[75] Walid Mahdi, Seyyid Ahmed Medjahed, and Mohammed Ouali. Performance analysis of simulated annealing cooling schedules in the context of dense image matching. *Computación y Sistemas*, 21(3):493–501, 2017.

[76] Didier Massonnet and Kurt L Feigl. Radar interferometry and its application to changes in the earth's surface. *Reviews of Geophysics*, 36(4):441–500, 1998.

[77] Catherine C McGeoch. Adiabatic quantum computation and quantum annealing: Theory and practice. *Synthesis Lectures on Quantum Computing*, 5(2):1–93, 2014.

[78] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 196–201. IEEE, 2011.

[79] Kamal Nasrollahi and Thomas B. Moeslund. Super-resolution: a comprehensive survey. *Machine Vision and Applications*, 25(6):1423–1468, Aug 2014.

[80] Ryo Natsuaki and Akira Hirose. Insar local co-registration method assisted by shape-from-shading. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(2):953–959, 2012.

[81] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[82] National Academy of Engineering. *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2018 Symposium*. The National Academies Press, Washington, DC, 2019.

[83] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[84] C. Peyrard. *Single image super-resolution based on neural networks for text and face recognition*. Theses, Université de Lyon, 2017.

[85] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[86] E. Ribeiro, A. Uhl, F. Alonso-Fernandez, and R. A Farrugia. Exploring deep learning image super-resolution for iris recognition. In *2017 25th European Signal Processing Conference*, pages 2176–2180. IEEE, 2017.

[87] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[88] Tara N. Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8614–8618, 2013.

[89] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455, 2009.

[90] R Scharroo, KF Wakker, and GJ Mets. The orbit determination accuracy of the ers-1 mission. In *2nd ERS-1 Symposium on Space at the Service of Our Environment*, 1994.

[91] W. Shi, J. Caballero, F. HuszÃ¡r, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, June 2016.

[92] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[93] Umberto Spagnolini. 2-D phase unwrapping and phase aliasing. *Geophysics*, 58(9):1324–1334, 1993.

[94] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[95] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[96] Robert H Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Physical Review Letters*, 57(21):2607, 1986.

[97] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[98] Harold Szu and Ralph Hartley. Fast simulated annealing. *Physics letters A*, 122(3-4):157–162, 1987.

[99] G Tesfaye, Bradley Marchand, J Derek Tucker, Timothy M Marston, Daniel D Sternlicht, Mahmood R Azimi-Sadjadi, et al. Image-based automated change detection for synthetic aperture sonar by multistage coregistration and canonical correlation analysis. *IEEE Journal of Oceanic Engineering*, 41(3):592–612, 2015.

[100] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in Neural Information Processing Systems*, pages 1799–1807, 2014.

[101] Haohan Wang and Bhiksha Raj. On the origin of deep learning. *arXiv preprint arXiv:1702.07800*, 2017.

[102] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3304–3308. IEEE, 2012.

[103] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[104] CL Werner, S Hensley, and P Rosen. Application of the interferometric correlation coefficient for measurement of surface change. *Eos Trans. AGU*, 77:46, 1996.

[105] Songtao Wu, Shenghua Zhong, and Yan Liu. Deep residual learning for image steganalysis. *Multimedia Tools and Applications*, 77(9):10437–10453, 2018.

[106] Saining Xie, Ross Girshick, Piotr DollÃ¡r, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017.

[107] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[108] Nestor Yague-Martinez, Francesco De Zan, and Pau Prats-Iraola. Coregistration of interferometric stacks of sentinel-1 tops data. *IEEE Geoscience and Remote Sensing Letters*, 14(7):1002–1006, 2017.

[109] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, and Qingmin Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 2019.

[110] Leslie Ying. Phase unwrapping. *Wiley Encyclopedia of Biomedical Engineering*, 2006.

[111] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[112] Arman Zaribafiyan, Dominic JJ Marchand, and Seyed Saeed Changiz Rezaei. Systematic and deterministic graph minor embedding for cartesian products of graphs. *Quantum Information Processing*, 16(5):136, 2017.

[113] Howard A Zebker, Charles L Werner, Paul A Rosen, and Scott Hensley. Accuracy of topographic maps derived from ers-1 interferometric radar. *IEEE Transactions on Geoscience and Remote Sensing*, 32(4):823–836, 1994.

[114] S. Zhang and Y. Lu. Image resolution enhancement using a hopfield neural network. In *Fourth International Conference on Information Technology (ITNG'07)*, pages 224–228, April 2007.

[115] Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[116] Weibao Zou, Yan Li, Zhilin Li, and Xiaoli Ding. Improvement of the accuracy of insar image co-registration based on tie points–a review. *Sensors*, 9(2):1259–1281, 2009.

Appendix A

Cost Derivation

Denoting by φ_i the phase of pixel i, and by ϕ_i the wrapped phase of the same pixel, we can relate the phase and wrapped phases of pixels i and j as follows.

$$\varphi_i = \phi_i + 2\pi k_i \tag{A.1}$$

and

$$\varphi_j = \phi_j + 2\pi k_j \tag{A.2}$$

Further, due to the Nyquist criterion we introduced earlier in Chapter 2, and if pixels i and j are neighbouring, then

$$\varphi_i - \varphi_j < 2\pi \tag{A.3}$$

The wrap function is defined as:

$$wrap\left(\theta\right) = arg\left(e^{i\theta}\right) = \theta - \lfloor\frac{\theta}{2\pi}\rfloor \tag{A.4}$$

Then, we can reason as follows: From 1 and 2 we have:

$$\varphi_i - \varphi = \phi_i - \phi_j + 2\pi\left(k_i - k_j\right) \tag{A.5}$$

or applying the $wrap(.)$ function on both sides, we obtain:

$$wrap\left(\varphi_i\text{-}\varphi_j\right) = wrap\left(\phi_i\text{-}\phi_j + 2\pi\left(k_i - k_j\right)\right) \Rightarrow$$

$$\varphi_i - \varphi_j - 2\pi\lfloor\frac{\varphi_i - \varphi_j}{2\pi}\rfloor = \phi_i - \phi_j + 2\pi\left(k_i - k_j\right) - 2\pi\lfloor\frac{\phi_i - \phi_j + 2\pi\left(k_i - k_j\right)}{2\pi}\rfloor \quad \text{(A.6)}$$

Because of the Nyquist assumption (c.f. eq (A.3))

$$\lfloor\frac{\varphi_i - \varphi_j}{2\pi}\rfloor = 0$$

and therefore equation 6 can be written as:

$$\varphi_i - \varphi_j = \phi_i - \phi_j + 2\pi\left(k_i - k_j\right) - 2\pi\lfloor\frac{\phi_i - \phi_j + 2\pi\left(k_i - k_j\right)}{2\pi}\rfloor \Rightarrow$$

$$\varphi_i - \varphi_j = \phi_i - \phi_j + 2\pi\left(k_i - k_j\right) - 2\pi\lfloor\frac{\phi_i - \phi_j}{2\pi} + \left(k_i - k_j\right)\rfloor \Rightarrow$$

$$\varphi_i - \varphi_j = \phi_i - \phi_j + 2\pi\left(k_i - k_j\right) - 2\pi\lfloor\frac{\phi_i - \phi_j}{2\pi}\rfloor - 2\pi\left(k_i - k_j\right)$$

since $(k_i - k_j)$ is an integer. Therefore:

$$\varphi_i - \varphi_j = \phi_i - \phi_j - 2\pi\lfloor\frac{\phi_i - \phi_j}{2\pi}\rfloor = wrap\left(\phi_i - \phi_j\right) \quad \text{(A.7)}$$

Using equation (A.7) and (A.5), we obtain:

$$\varphi_i - \varphi_j = \phi_i - \phi_j + 2\pi\lfloor k_i - k_j\rfloor = wrap\left(\phi_i - \phi_j\right) \Rightarrow$$

$$k_i - k_j = \frac{wrap\left(\phi_i - \phi_j\right) - \left(\phi_i - \phi_j\right)}{2\pi} \quad \text{(A.8)}$$

Denoting

$$a_{ij} \stackrel{def}{=} \frac{wrap\left(\phi_i - \phi_j\right) - \left(\phi_i - \phi_j\right)}{2\pi} \quad \text{(A.9)}$$

equation 8 is written as:

$$k_i - k_j = a_{ij} \Rightarrow k_i - k_j - a_{ij} = 0 \quad \text{(A.10)}$$

This equation is the basis of the cost function the optimization of which will produce appropriate values for the labes k_i.

Appendix B

Generating Real-like Synthetic Images

We tried to generate synthetic images that have similar characteristics to the real dataset we have. Even though the real dataset we have doesn't represent a wide spectrum of the InSAR image, having a synthetic dataset (synthetic dataset 2) with similar characteristics to the real dataset helps validate our approach and the results we obtain from synthetic dataset 1 (that include a wider range of the InSAR images).

We selected the spectrum to be the factor that indicates the similarity between the images. Hence, we generated synthetic images with spectra that are close to the real image.

To compare the spectrum of two images, we measure the coherence between the filtered spectra. First, we generate the spectrum of each image. Second, we apply a uniform filter per each dimension to smooth the spectrum. The filter length is 50. Third, we measure the coherence between the filtered spectra.

The coherence between two signals, $x(t)$ and $y(t)$, is a real-valued function between 0 and 1. This function is defined as:

$$C_{xy}(f) = \frac{|G_{xy}(f)|^2}{(G_{xx}(f)\, G_{yy}(f))}$$

where $G_{xy}(f)$ is the cross-spectral density between x and y, while $G_{xx}(f)$ and $G_{yy}(f)$ are the auto spectral densities of x and y, respectively.

We have two sets. First, the set of real images which we want to imitate. Second, the set of synthetic images which we want each image in the set to have coherence as close as possible to at least one of the real images.

Table B.1: The calculated coherence for each synthetic image in dataset 2

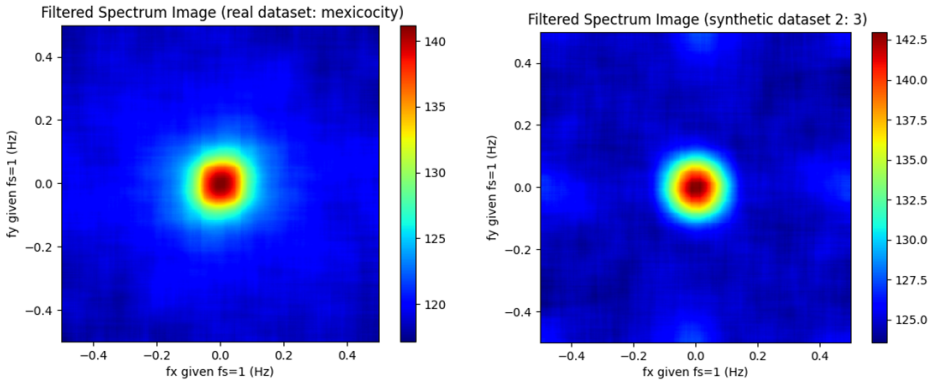
Synthetic image	Maximum Coherence
0	0.8691
1	0.8865
2	0.8737
3	0.8745
4	0.8359
5	0.9616
6	0.9655
7	0.9579
8	0.9579
9	0.9622
Mean	0.91448
Standard deviation	0.05072

To measure the similarity between the two sets, for each synthetic image we calculate its coherence to every real image and find the largest coherence.

Table B.1 shows the calculated coherence for each synthetic image. The table shows relatively large mean, with a small standard deviation, which indicates that the synthetic images are close (coherent) to the real ones.

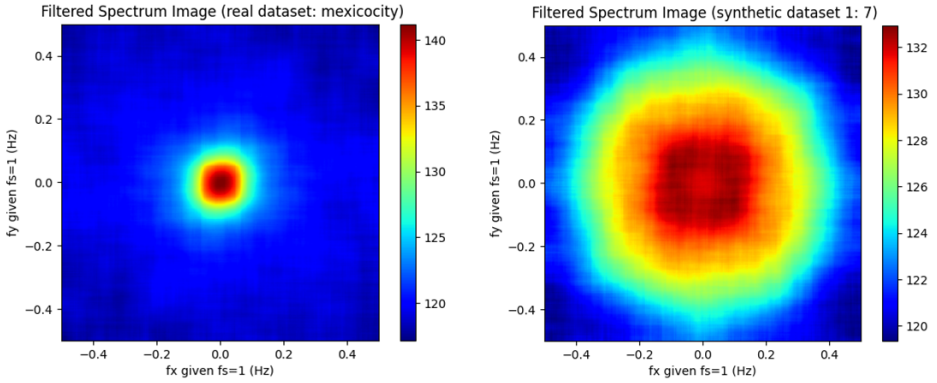
To inspect the spectra visually, Figure B.1 shows the filtered spectrum of one of the real images and one of the synthetic dataset 2 images. Both images are selected randomly.

On the other hand, Figure B.2 shows the filtered spectrum of the real image above and a randomly selected image from the synthetic dataset 1 (high-complexity high-noise images).



(a) Real image (b) Synthetic image (dataset 2)

Figure B.1: The filtered spectrum of one of the real images and one of the synthetic dataset 2 images



(a) Real image (b) Synthetic image (dataset 1)

Figure B.2: The filtered spectrum of one of the real images and a randomly selected image from the synthetic dataset 1 (high-complexity high-noise images)

Appendix C

Perlin Image Generation

Perlin noise is a popular algorithm for procedural generation invented by Ken Perlin [83]. It is used to generate textures and terrain procedurally; without being manually made by a designer.

We generate synthetic images using the Perlin image generator developed by 3vGeomatics.

The properties of the generated synthetic images do not represent all of the properties of real InSAR images. However, they have some properties in common with real InSAR images.

The generator provides control over the strength and the correlation of the image and the noise generated.

The strength of the image directly reflects the magnitude of the signal. Therefore, we can use it to control the wraps in the wrapped signal, hence the maximum number of labels needed by the phase unwrapping solver. On the other hand, the strength of the noise signal directly reflects the SNR of the image.

The correlation controls the complexity of the generated signal, how fast it changes. Increasing the correlation of an image makes it prone to aliasing due to the rapid changes in the signal. For the noise, the correlation is usually an order of magnitude higher than the image correlation.

Appendix D

Noisy Unwrapped vs. Deemed Noisy Unwrapped Ground Truth

In this appendix, we are experimenting the super-pixel algorithm and compare results to Noisy Unwrapped Ground Truth and Deemed Noisy Unwrapped Ground Truth to show how close the two ground truths are.

D.1 Synthetic Dataset 1 Results

The figures D.1 and D.2 summarize the average matching for different maximum labels and different ground truths. Figure D.1 summarize the average matching fraction for different maximum labels compared to the Deemed Noisy Unwrapped Ground Truth for the one-pass super-pixel algorithm, with sub-image size of 20×20. Figure D.2 summarize the average matching fraction for different maximum labels compared to the Deemed Noisy Unwrapped Ground Truth for the two-pass super-pixel algorithm, with sub-image size of 10×10.

D.1.1 Discussion

Figure D.1 and Figure D.2 show how close the two ground truths are; the Noisy Unwrapped Ground Truth (Figure D.1-a,c,e and Figure D.2-a,c,e) and the Deemed Noisy Unwrapped Ground Truth (Figure D.1-b,d,f and Figure D.2-b,d,f). By comparing the corresponding bars, we can see that the matching fractions compared to the two ground truths are very close, with a maximum difference of 0.45% between Figure D.2-c and Figure D.2-d (high-complexity and high-noise).

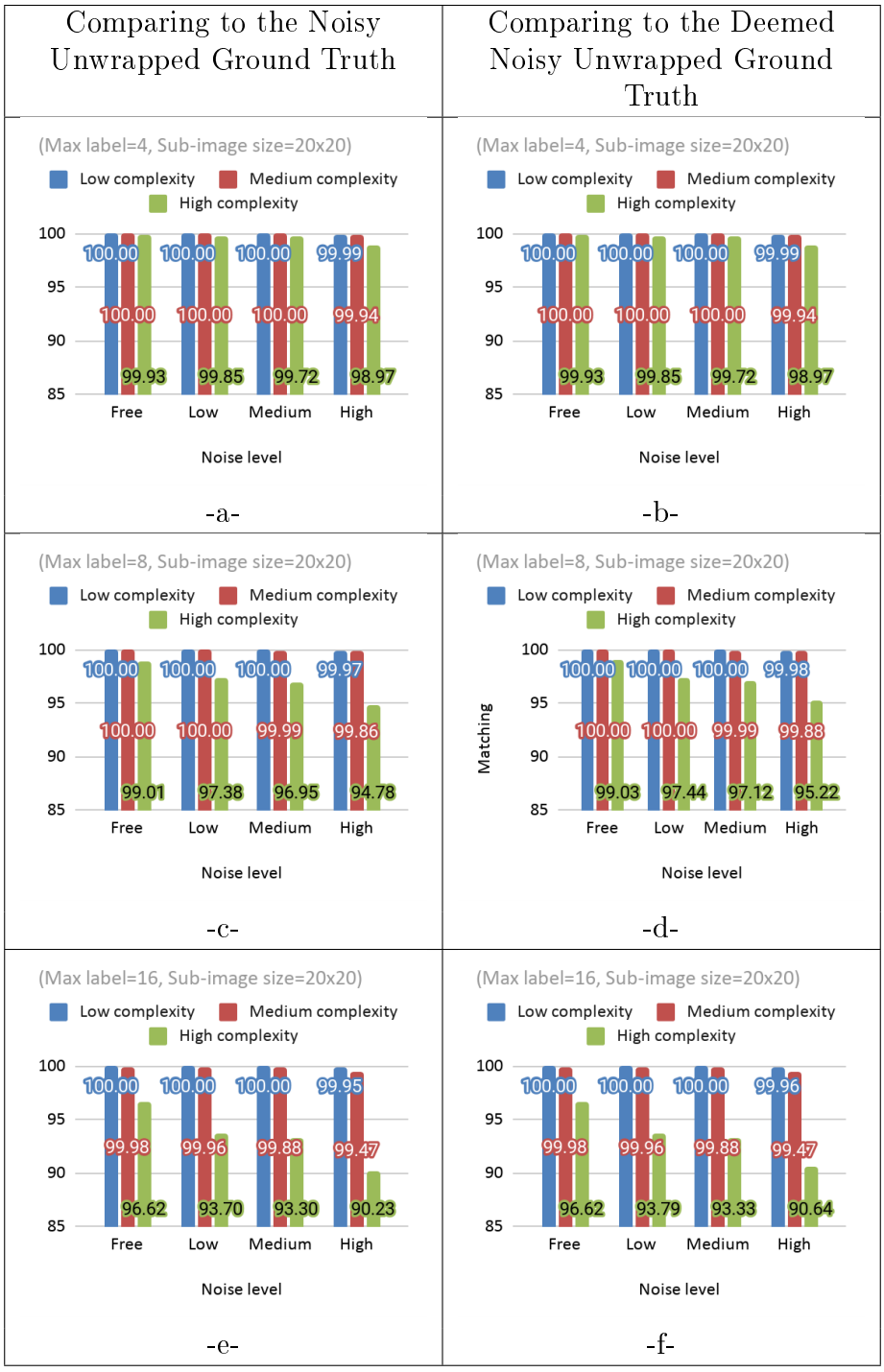


Figure D.1: The matching fraction of the one-pass super-pixel algorithm for different maximum labels compared to Noisy Unwrapped Ground Truth and Deemed Noisy Unwrapped Ground Truth

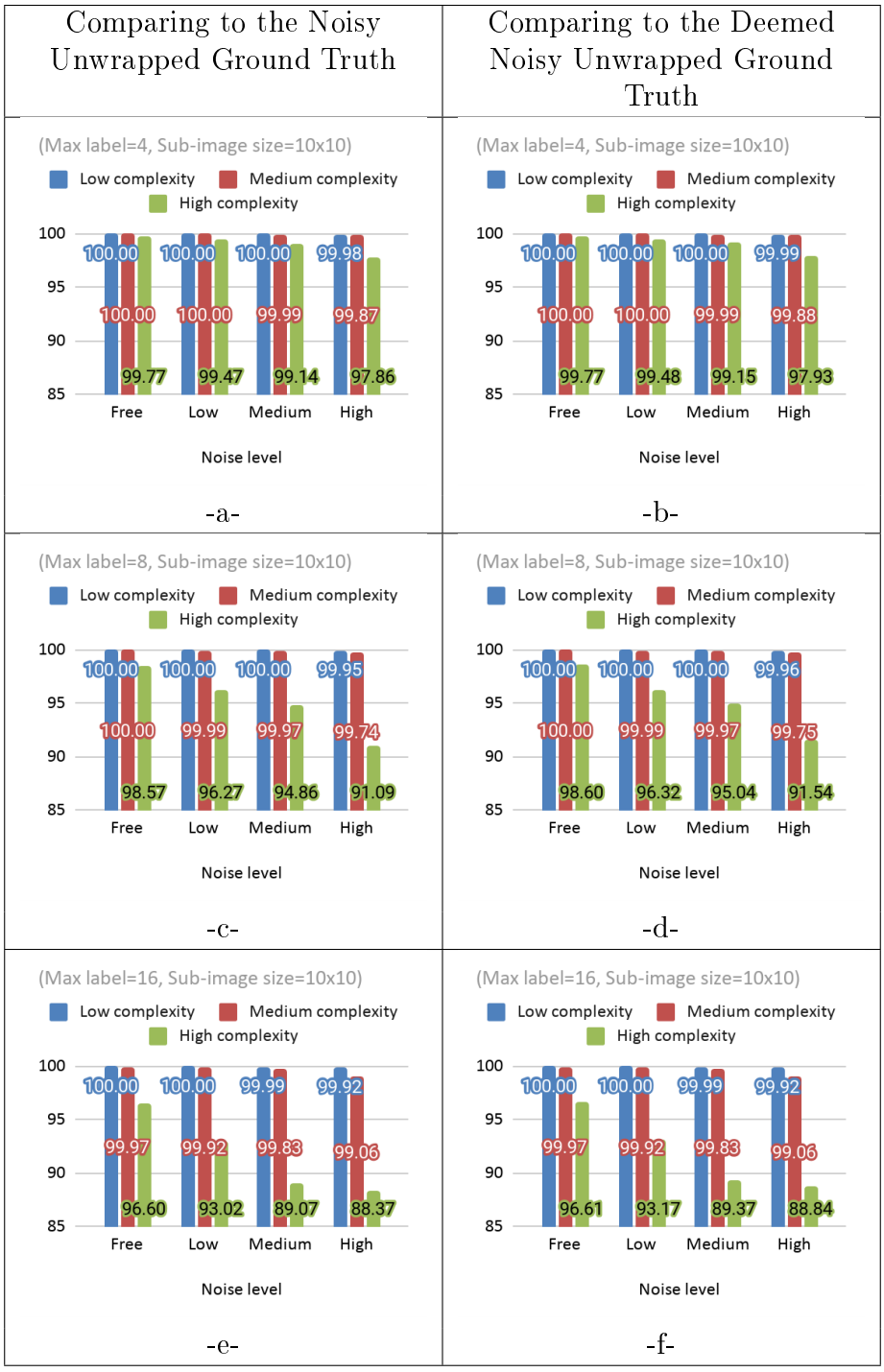


Figure D.2: The matching fraction of the two-pass super-pixel algorithm for different maximum labels compared to Noisy Unwrapped Ground Truth and Deemed Noisy Unwrapped Ground Truth

Appendix E

Aliasing

Noise and complexity both negatively impact the unwrapped image quality. In this appendix we try to relate this to the aliasing problem, showing that the aliasing is the main contributor in the super-pixel solver deterioration at high noise and high complexity. First, we directly relate the low performance to the average aliasing in the image. Second, we show samples of when the aliasing at the boundaries of the sub-images and how this affects the solution even more.

E.1 Aliasing Fraction

The aliasing is detected if the absolute phase difference between two adjacent samples is greater than π [93]. Hence, we get the total number of aliasing in an image by scanning the image horizontally and vertically and counting the adjacent pixels with aliasing. To normalize the result, we divide the total number of aliasing by the total number of the adjacency between pixels, that is:

$$Aliasing\ fraction = \frac{Total\ aliasing}{Total\ adjacency} = \frac{\sum_{x=1}^{M-1}\sum_{y=1}^{N-1}\left[\left|\theta\left(x,y\right)-\theta\left(x+1,y+1\right)\right|>\pi\right]}{(M-1)N+(N-1)M}$$

E.2 Average Aliasing Results

Figure E.1 shows the average aliasing fraction for the synthetic dataset 1. Figure E.1 reflects the performance degradation seen in Figure 5.5 and Figure 5.6 at high complexity and high noise. At high complexity and high noise we can see relatively

high aliasing that causes the matching fraction shown in Figure 5.5 and Figure 5.6 to be lower.

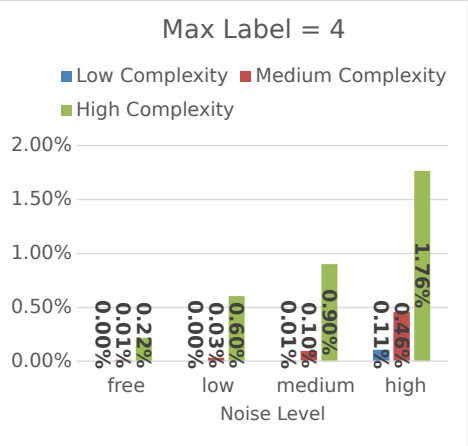
E.3 Aliasing Samples

The figures E.2, E.3, and E.4 show some sample images with different maximum labels. All images are 100×100 pixels of varying label range. These images were chosen at random from the set of synthetic images with high noise content and high complexity. Each figure shows on the first row, from left to right, the Noisy Unwrapped Ground Truth, the solution obtained from applying TRWS directly, and the solution obtained by applying the super-pixel algorithm with a sub-image size of 20×20. The grid in the figures delineates the sub-images.

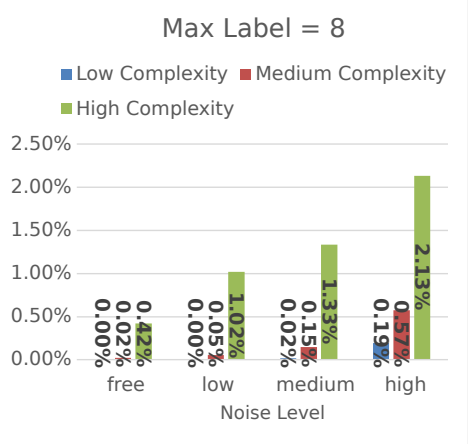
On the second row, we show from left to right the aliasing locations of the ground truth unwrapped image, the error locations of the TRWS direct solution compared to the ground truth, and the error location of the super-pixel solution compared to the ground truth.

By examining the errors in the TRWS direct solution and the super-pixel solution we can see that the error locations mostly coincide with the locations of the aliasing. It seems that if the aliased pixels are isolated, data from neighboring pixels provide sufficient information to correctly unwrap these pixels. However, where aliased pixels are concentrated or where data from neighboring pixels is not accessible, unwrapping errors seem to occur. This is shown in Figure E.2-e, Figure E.3-e, and Figure E.4-e where the errors correlate with locations of high concentrations of aliased pixels.

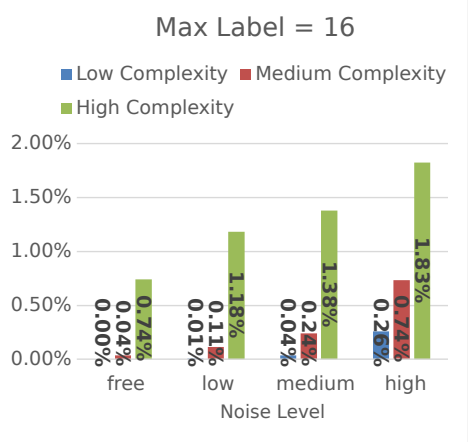
Also as one can observe in Figure E.2-f, Figure E.3-f, and Figure E.4-f, most of the errors observed exist at the sub-image boundaries and correlate with the aliased pixels.



(a) Max label = 4

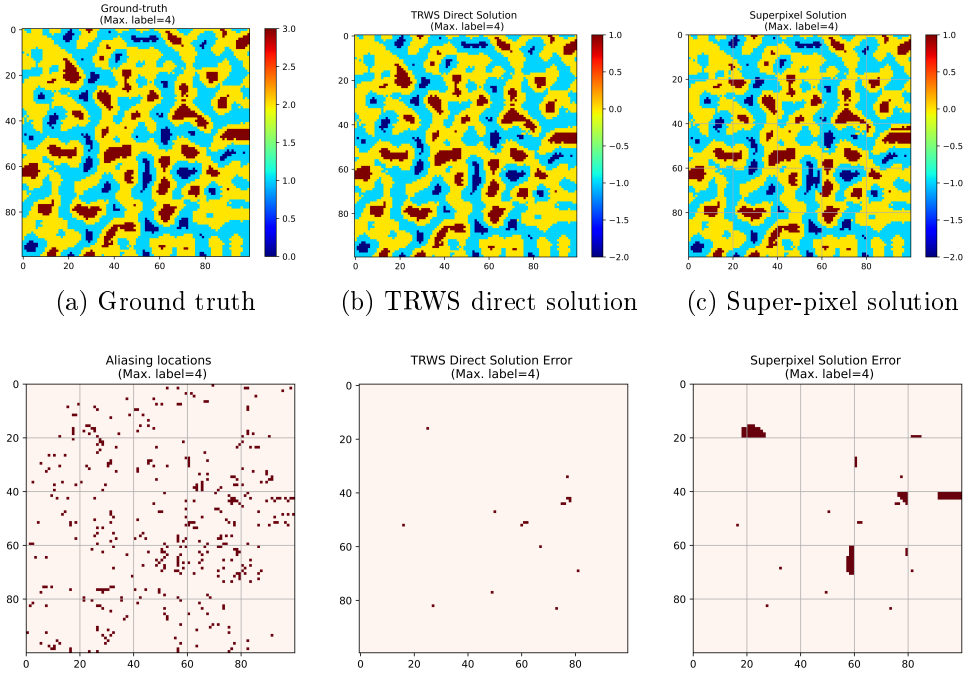


(b) Max label = 8



(c) Max label = 16

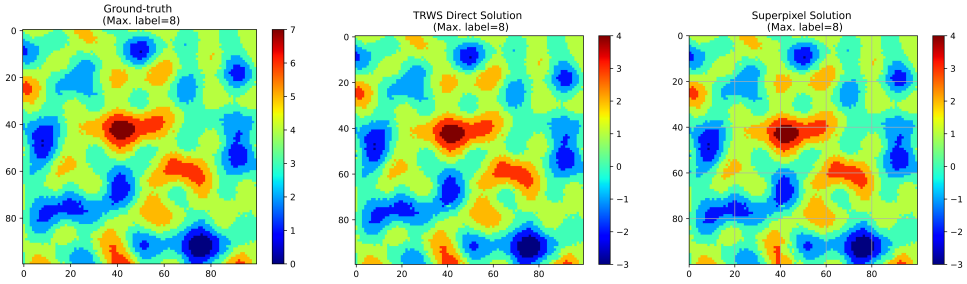
Figure E.1: The average aliasing fraction for the synthetic dataset 1



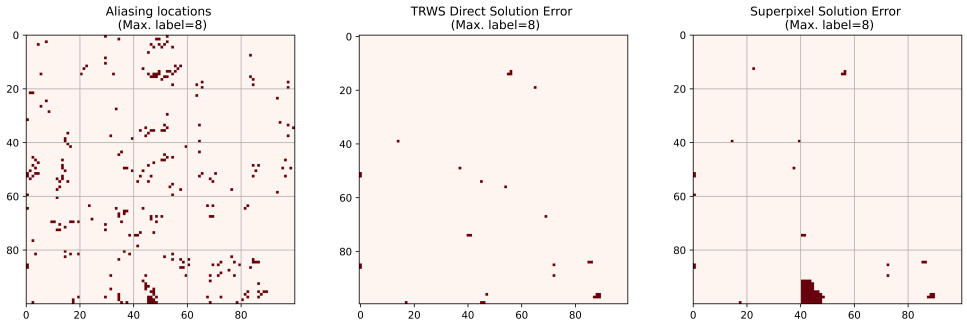
(a) Ground truth (b) TRWS direct solution (c) Super-pixel solution

(d) Aliasing locations (e) TRWS direct solution error (f) Super-pixel solution error

Figure E.2: The aliasing of a sample image with maximum label of 4

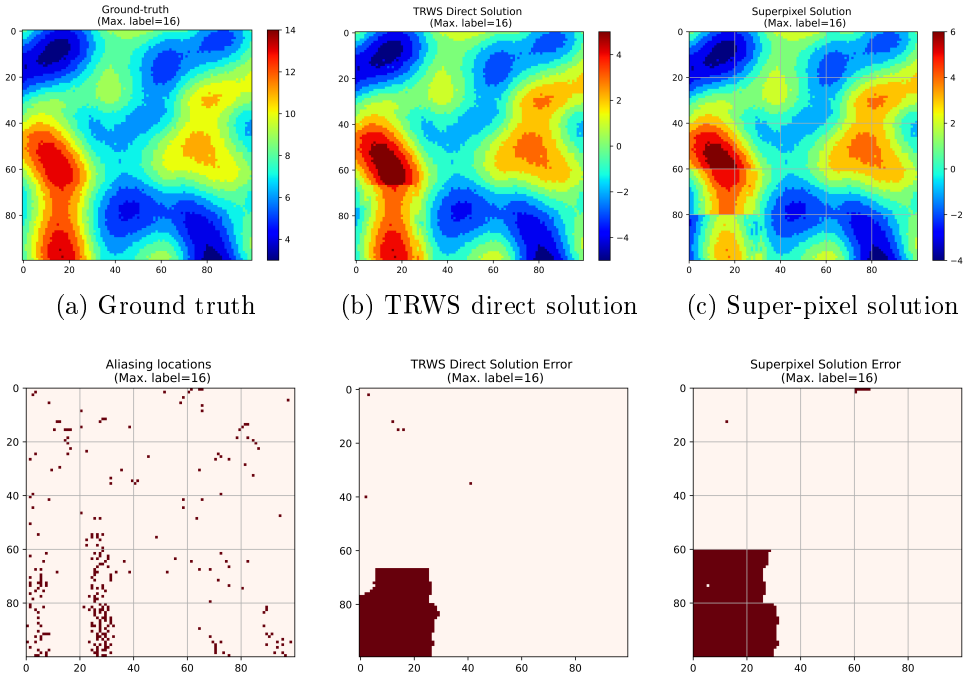


(a) Ground truth (b) TRWS direct solution (c) Super-pixel solution



(d) Aliasing locations (e) TRWS direct solution error (f) Super-pixel solution error

Figure E.3: The aliasing of a sample image with maximum label of 8



(a) Ground truth (b) TRWS direct solution (c) Super-pixel solution

(d) Aliasing locations (e) TRWS direct solution error (f) Super-pixel solution error

Figure E.4: The aliasing of a sample image with maximum label of 16

Appendix F

Problems Too Large for Current QUBO Solvers

As the size of labels increases, pixels require more qubits to encode such labels. This results in images with increased label size often require resources (e.g., number of qubits) that are not available. Since not all labels in an image have a maximum size, one can attempt to unwrap these images by encoding the range (delta) of the labels in the sub-images. The range of the labels is in general smaller than the label size, and as such, sub-images can be better accommodated by the available resources of the solvers. Yet, errors are introduced in the cases where the range exceeds the maximum label size used for the encoding, and this is shown in the experiments described in this section.

In this experiment set, we are studying the effect of the maximum label on the quality of the solution. Hence, we need images with different maximum labels. In addition to the image selected for the experiments in section 5.4.4.2 (the image with maximum label 4, high-complexity and high-noise), we randomly selected two more image with similar characteristics (the same complexity and noise level); however, with different maximum label (one with maximum label of 8 and another with maximum label of 16). However, these additional images, having large maximum labels, could not be fully accommodated by the solvers. As we shall discuss, this impacted the quality of the solution obtained.

Since most of the real images have maximum label of 4, with only 4 images with maximum label of 5, the effect of the limitation of the 4 labels is not evident in the real images results. Therefore, we are including only the synthetic images in this

Table F.1: The cases considered for experiment 1

Case	Max label	Complexity	Noise	Size	Sub-image size
0	4	High	High	120×120	10×10
1	8	High	High	120×120	10×10
2	16	High	High	120×120	10×10

experiment.

F.1 Experiment 1 (Images With Label Higher Than 4)

In this experiment, we tried three images with different maximum labels. These images were chosen at random from the set of synthetic images with high-noise content, high-complexity and different maximum label count. One of these images (the one with maximum label of 4) is already selected and processed in the previous experiments. Hence, some of these results might be used in this experiment.

Each of the 400×400 pixels images were cropped (at random) to a size of 120×120 pixels to limit the computational load of the limited-access QUBO solvers. For all cases, the sub-image size is 10×10 with zero overlapping margins. The matching fraction was obtained with reference to the Noisy Unwrapped Ground Truth. We used the two-pass super-pixel approach in this experiment set. Table F.1 summarizes the properties of the images.

F.1.1 Results

With the solvers' maximum label fixed to 4, Figures F.1 summarize the matching fraction for different solvers and different maximum labels. The format of the reported results has been discussed in Section 5.4.4.2.

Figure F.2 summarizes the matching of Direct TRWS, super-pixel TRWS, super-pixel D-Wave, super-pixel Simulated Annealing, and super-pixel Parallel Tempering.

For the D-Wave solver, the matching fraction decreases significantly when images have a maximum label of 16. Due to the limited number of qubits in the D-Wave solver (2041 qubits), the D-Wave solver cannot accommodate large images with large label

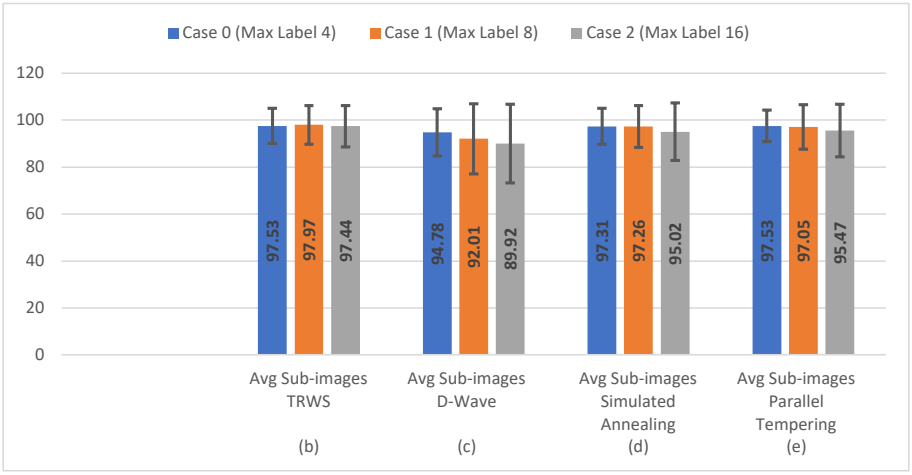


Figure F.1: The average sub-image matching fraction for the different sub-image sizes and different solvers (TRWS, D-Wave, Simulated Annealing, and Parallel Tempering) for experiment 1

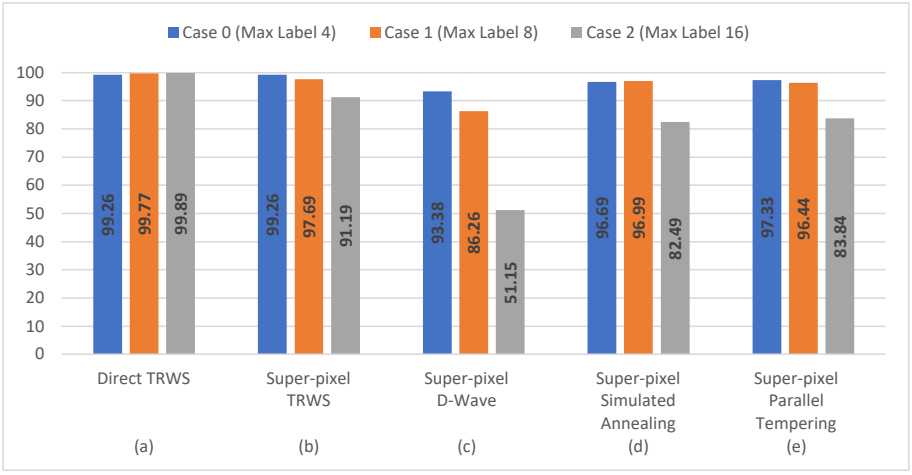


Figure F.2: The super-pixel matching fraction for the different sub-image sizes and different solvers (TRWS, D-Wave, Simulated Annealing, and Parallel Tempering) for experiment 1

sizes. The largest label-size of 10×10 pixel images is four. The experiments reported in this section unwrap images with a maximum label-size of 4. Since test images may include pixels with label-sizes larger than 4, the unwrapping of such images results in errors which are reflected in decreased matching fractions as evidenced in the results presented in the figures above and especially for the image with maximum labels of 16. In addition, this image has a high-noise content and high-complexity, factors that introduce additional errors (c.f. Section 5.4.4.2).

For the Simulated Annealing and Parallel Tempering, even though the solvers can accept larger problems, the solvers converge to a solution only if the problem size is less than or equal to 20×20 with a maximum label of 4. Therefore, we choose to use 10×10 sub-images with maximum label 4 to be consistent with the results from D-Wave annealer.

The process of unwrapping is not a linear one, in the sense that an increased number of errors in unwrapping the sub-images affects the super-pixel method disproportionally.

In the following subsections, we present details of the composition of the images we have used, and the results we have obtained. As it can be seen in Table F.2 and Table F.3, where the number of sub-images with labels that are larger than 4 is small, as is the case 0 and 1, the quality of the solution for all solvers is high (a matching fraction greater than .8626). However, when the number of sub-images with labels that are larger than 4 is large, the quality of the solution is affected in a major way. This is shown in Table F.4 where more than 21% of the sub-images have labels that are larger than 4. The quality of the unwrapping deteriorates significantly with the D-Wave solver, achieving a matching fraction of only .5115.

F.1.2 Sub-images Average Matching

F.1.3 Discussion

Although the D-Wave has similar performance to the Simulated Annealing and Parallel Tempering for the sub-images with label size of less than 4 (the sub-images that fit) as per the Table F.3 and Table F.2, here, the D-Wave has a disproportionately lower performance as compared to the other solvers for the sub-images that have label sizes larger than 4 and hence cannot be accommodated directly on the solver.

In this experiment, the D-Wave solver has similar performance to the one we obtained in the experiment described in Section 5.4.4.2. However, these results are

Table F.2: Case 0 (Max label = 4, low-complexity, noise-free) results for experiment 1

	# sub-image	Avg matching			
		TRWS	D-Wave	Simulated Annealing	Parallel Tempering
Total sub-images	144	97.53	94.78	97.31	97.85
Super-pixel solution	-	99.26	93.38	96.69	97.33

Table F.3: Case 1 (Max label = 8, medium-complexity, medium-noise) results for experiment 1

	# sub-image	Avg matching			
		TRWS	D-Wave	Simulated Annealing	Parallel Tempering
Sub-images with label-size <= 4	125	99.51	94.6	99.59	99.59
Sub-images with label-size > 4	19	87.79	75.0	81.95	80.32
Total sub-images	144	97.97	92.01	97.26	97.05
Super-pixel solution	-	97.69	86.26	96.99	96.44

Table F.4: Case 2 (Max label = 16, high-complexity, high-noise) results for experiment 1

	# sub-image	Avg matching			
		TRWS	D-Wave	Simulated Annealing	Parallel Tempering
Sub-images with label-size <= 4	113	99.31	95.42	99.47	99.45
Sub-images with label-size > 4	31	90.65	69.87	78.8	80.97
Total sub-images	144	97.44	89.92	95.02	95.47
Super-pixel solution	-	91.19	51.15	82.49	83.84

Table F.5: The cases considered for experiment 2

Case	Max label	Complexity	Noise	Size	Sub-image size
0	16	High	High	120×120	10×10
1	16	High	High	120×120	8×8

based on a single image, and one may not draw statistical conclusions.

F.2 Experiment 2 (The Effect of Reducing the Sub-Image Size on the Maximum Label)

Reducing the sub-image size limits the range of the labels in the sub-image. Hence, reducing the sub-image size from 10×10 to 8×8 is expected to reduce the sub-images with maximum labels greater than 4 and improve the performance of the unwrapping. On the other hand, using smaller sub-images increases the total number of the sub-images and the probability of the boundary errors which negatively affects the performance of the unwrapping.

In this experiment, we compare the solution when a 10×10 sub-image size is used to when an 8×8 sub-image is used. In the matching calculation, we compare the results to the Noisy Unwrapped Ground Truth. Table F.5 summarizes the properties of the images.

Even though the images have a label size of 16, the sub-images generated during the super-pixel algorithm have a smaller label dynamic range.

F.2.1 Results

Figure F.3 summarizes the average sub-image unwrapping matching for the different sub-image sizes and different solvers (TRWS, D-Wave, Simulated annealing, and parallel tempering).

Figure F.4 summarizes the matching of Direct TRWS, super-pixel TRWS, super-pixel D-Wave, super-pixel Simulated Annealing, and super-pixel Parallel Tempering.

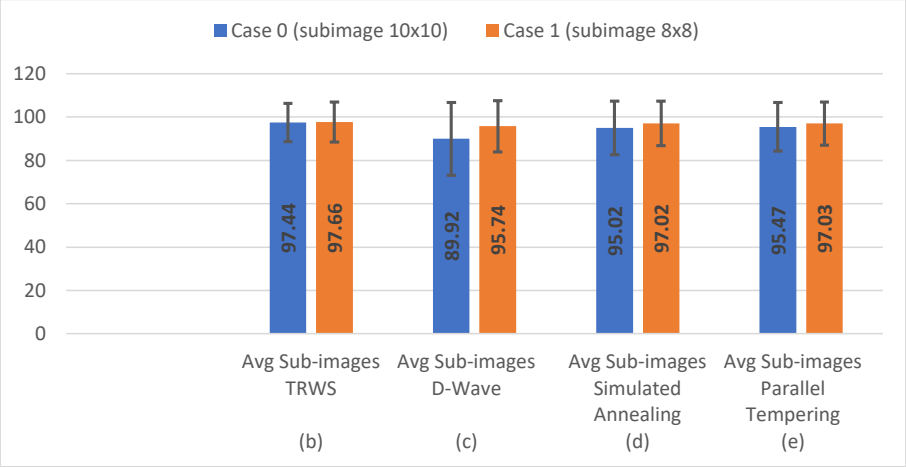


Figure F.3: The average sub-image matching fraction for the different sub-image sizes and different solvers (TRWS, D-Wave, Simulated Annealing, and Parallel Tempering) for experiment 2

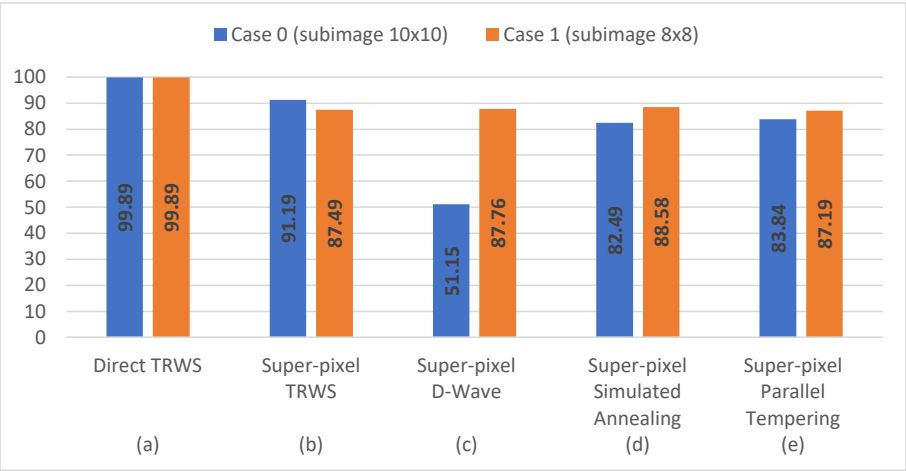


Figure F.4: The super-pixel matching fraction for the different sub-image sizes and different solvers (TRWS, D-Wave, Simulated Annealing, and Parallel Tempering) for experiment 2

	# sub-image	Avg matching			
		TRWS	D-Wave	Simulated Annealing	Parallel Tempering
Sub-images with label-size <= 4	113	99.31	95.42	99.47	99.45
Sub-images with label-size > 4	31	90.65	69.87	78.8	80.97
Total sub-images	144	97.44	89.92	95.02	95.47
Super-pixel solution	-	91.19	51.15	82.49	83.84

Table F.6: Case 0 (sub-image = 10×10) results for experiment 2

F.2.2 Sub-images Average Matching

Case 0 is identical to Table F.4. The results are added again here for the sake of results completeness.

F.2.3 Discussion

The D-Wave matching improves by reducing the size of the sub-image. This improvement is because the percentage of the sub-images with label sizes larger than 4 is reduced. Hence, the final solution is more influenced by the high matching of the sub-images with the maximum label that fits in the D-Wave. When the erroneous sub-images are less and their average match improves, then D-Wave yields results that are comparable to TRWS and the other QUBO solvers.

Similarly, for the Simulated Annealing and the Parallel Tempering the super-pixel matching is enhanced as the average sub-image matching is improved.

In conclusion, reducing the sub-image size reduces the number of the sub-images that do not fit in the solver. Hence, improving the average matching for the sub-images and the super-pixel solution.

	# sub-image	Avg matching			
		TRWS	D-Wave	Simulated Annealing	Parallel Tempering
Sub-images with label-size <= 4	200	99.49	98.52	99.77	99.67
Sub-images with label-size > 4	25	83.00	73.56	75	75.88
Total sub-images	225	97.66	95.74	97.02	97.03
Super-pixel solution	-	87.49	87.76	88.58	87.19

Table F.7: Case 1 (sub-image $= 8 \times 8$) results for experiment 2

Appendix G

Sub-image Size Effect on the Unwrapping Quality

In this section, we test the effect of the size of the sub-image selected and the choice of the one-pass or multi-pass super-pixel algorithm on the quality of the unwrapping solution. The whole synthetic dataset 1 is tested, with TRWS as a sub-image solver with different sub-image sizes. In this study, we examine the effects of the noise on the quality of the solution. We average the obtained matching fraction over all images in the dataset irrespective of their complexity. Since all the images are of the size 400×400, the images with sub-image size less than 20×20 are unwrapped using the two-pass super-pixel algorithm. If the sub-image size is 20×20 or larger, the one-pass super-pixel algorithm is used. In addition, to study the effectiveness of the two-pass super-pixel algorithm as compared to the one-pass one, we have also obtained the respective performance of selected sub-image sizes under the one and two-pass super-pixel algorithms.

For this section, we use the Noisy Unwrapped Ground Truth to compare the results of our approach and obtain the matching fractions.

G.1 Results

G.1.1 Maximum Label = 4

Figure G.1 shows the effect of the size of the sub-image on the matching fraction for the maximum label of 4.

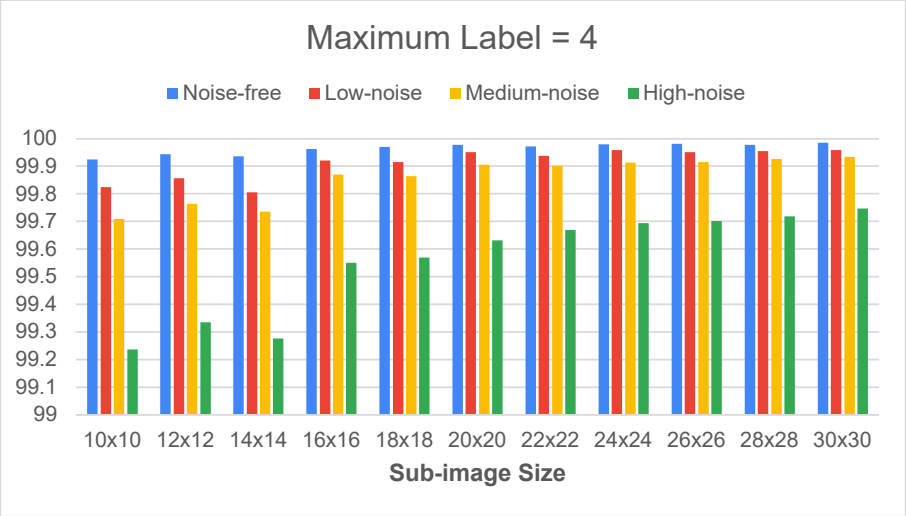


Figure G.1: Average matching fraction results over different noise content for different sub-image sizes. The maximum label is 4 and the images are compared to the Noisy Unwrapped Ground Truth

G.1.2 Maximum Label = 8

Figure G.2 shows the effect of the size of the sub-image on the matching fraction for the maximum label of 8.

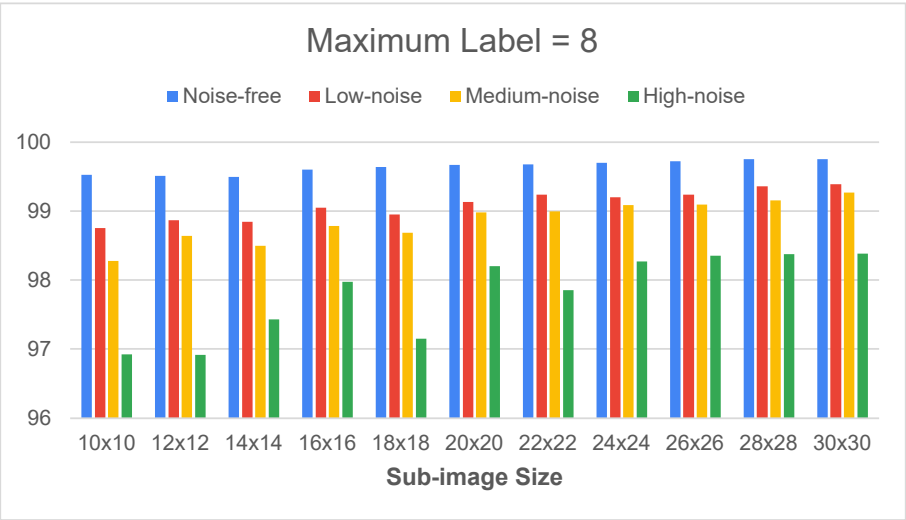


Figure G.2: Average matching fraction results over different noise content for different sub-image sizes. The maximum label is 8 and the images are compared to the Noisy Unwrapped Ground Truth

G.1.3 Maximum Label = 16

Figure G.3 shows the effect of the size of the sub-image on the matching fraction for the maximum label of 4.

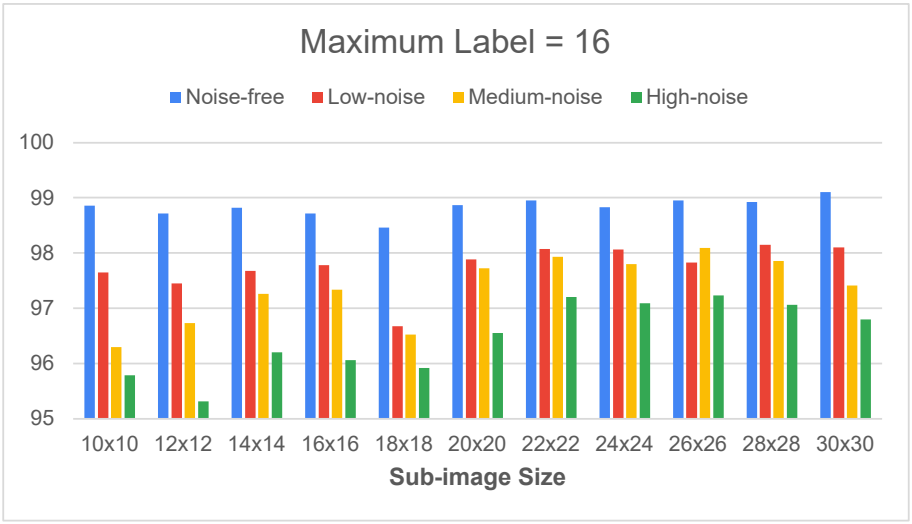


Figure G.3: Average matching fraction results over different noise content for different sub-image sizes. The maximum label is 16 and the images are compared to the Noisy Unwrapped Ground Truth

G.2 Discussion

Increasing the sub-image size improves the matching. This improvement is more evident with the images with high noise than the images with less noise. Reducing the sub-image size has a negative effect on the solution. Reducing the sub-image size increases the boundaries between the sub-images, which increases the boundary errors.

Appendix H

List of Publications

- Khaled A Helal, Bardia Barabadi, Amirali Baniasadi, and Nikitas Dimopoulos. Scale invariant super-resolutions methods with application to InSAR images. In *2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, 2019.

- Khaled A Helal Kelany, Amirali Baniasadi, Nikitas Dimopoulos, and Matthew Gara. Improving InSAR image quality and Co-registration through CNN-based Super-Resolution. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020.

- Khaled A Helal Kelany, Nikitas Dimopoulos, Clemens P. J Adolphs, Bardia Barabadi, and Amirali Baniasadi. Quantum annealing approaches to the phase-unwrapping problem in synthetic-aperture radar imaging. In *2020 IEEE International Conference on Quantum Computing & Engineering (QCE)*. pages 120-129, 2020.

- Khaled A Helal Kelany, Nikitas Dimopoulos, Clemens P. J Adolphs, and Amirali Baniasadi. Quantum Annealing Methods and Experimental Evaluation to the Phase-Unwrapping Problem in Synthetic-Aperture Radar Imaging. *IEEE Transaction on Quantum Engineering*, 2021, under revision.