
**Technische Universität Berlin**
**Institut für Luft- und Raumfahrt Fachgebiet Raumfahrttechnik**

**Fakultät V**
**Marchstraße 12-14**
**10587 Berlin**

# A Study on Co-registration of Synthetic Aperture Radar (SAR) images using Multiscale Attention U-Net Architecture

**Keshava Raaju Perumal**

**Matriculation Number: 465017**
**April 2025**

**Supervised by**
**Prof. Dr. Enrico Stoll**

**Assistant Supervisor**
**Dr. Francescopaolo Sica**

# Declaration of Authorship

I, Keshava Raaju Perumal certify that this thesis has been composed by me and is based on my own work, unless stated otherwise. In this thesis, no other work has been used without acknowledgment. All references and verbatim extracts have been quoted, and all sources of information, including graphs and data sets, have been specifically acknowledged.

02.05.2025

KESHAVA RAAJU PERUMAL

Berlin, Datum

# Agreement on Rights of Utilization

The Technische Universität Berlin, represented by the Chair of Space Technology, may use the results of the thesis at hand in education and research. It receives exclusive rights of utilization as according to 31 Abs. 2 Urheberrechtsgesetzt (Urhg). This right of utilization is unlimited and involves content of any kind (e.g. documentation, presentations, animations, photos, videos, equipment, parts, procedures, designs, drawings, software including source code, and similar).

Professor Dr.-Ing. Enrico Stoll
Head of the Chair of Space Technology

# Acknowledgements

The research described in this thesis was carried out at the Faculty of Aerospace Engineering in the Universität der Bundeswehr, München.

I would like to thank Prof. Dr. M. Schmitt, for giving me the opportunity to work in this field. Special thanks to my supervisor, Dr. Francescopaolo Sica, for his support and guidance throughout the duration of the thesis.

I would like to thank my family members, especially my parents and sister, who have constantly supported me throughout the duration of my thesis and encouraged me during the difficult times. I am grateful to my friends(Anagha, Bhavya, Naveen, Priya and Shruti) living in Koloniestr. 149, for being the pillar of support during the course of my thesis. I also would like to thank my best friend, Nita Prasad, who was always by my side during the same period.

# List of Figures

# List of Tables

# Symbols

| | |
|---|---|
| i | Sample |
| n | Total number of samples |
| p | Probability distribution |
| $\phi$ | Phase difference |
| $s_1$ | Primary complex signal |
| $s^*_2$ | Conjugate of Secondary complex signal |
| $\sum$ | Summation |
| x | Reference image |
| y | Secondary image |
| $y_i$ | Ground truth of the $i^{th}$ sample |
| $\hat{y}_i$ | Predicted value of the $i^{th}$ sample |

# Abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| ARES | Airborne Reflective Emissive Spectrometer |
| arg | Argument; extraction of phase angle |
| AZ | Azimuth |
| CC | Cross-correlation |
| CNN | Convolutional Neural Network |
| CPU | Central Processing Unit |
| CUDA | Compute Unified Device Architecture |
| Eg | Example |
| GB | Gigabytes |
| GPU | Graphical Processing Unit |
| HH | Transmit Horizontal, Receive Horizontal |
| HV | Transmit Horizontal, Receive Vertical |
| InSAR | Interferometric Synthetic Aperture Radar |
| MAE | Mean Absolute Error |
| MI | Mutual Information |
| MSE | Mean-Squared Error |
| NCC | Normalized Cross-Correlation |
| ONXX | Open Neural Network Exchange |
| OS | Operating System |
| PINN | Physics-Informed Neural Network |
| R | Correlation |
| Radar | Radio Detection and Ranging |
| RAM | Random Access Memory |
| RAR | Real Aperture Radar |
| RG | Range |
| SAR | Synthetic Aperture Radar |
| SE | Squeeze and Excitation |
| SIFT | Scale-Invariant Feature Transform |
| SURF | Speeded-Up Robust Features |

| | |
|---|---|
| VH | Transmit Vertical, Receive Horizontal |
| VRAM | Video Random Access Memory |
| VV | Transmit Vertical, Receive Vertical |

# Abstract

Synthetic Aperture Radar(SAR)[2] is a remote sensing technique that generates high-resolution images of the Earth's surface using microwave wavelengths, regardless the weather and illumination. It involves various phenomena such as distortions, registration and interferometry[3] of the Radar images of the same area at different times. Although it sounds simple, it is highly challenging in terms of atmospheric effects, grainy noise during interference[2], large computational resources for processing and analysis, interpretation, and extraction of meaningful information from images. Accurate registration[4] of SAR is crucial for time-series analysis and change detection (Eg. earthquakes studies). Traditional methods used in the image processing require prior information about the acquisition, which might not be always available or could be inaccurate. For processing SAR data, recent deep learning[5] methods have revolutionized geospatial image processing. This thesis focuses on co-registration of SAR images without prior knowledge of their acquisition geometry and other crucial information about acquisition. The proposed method learns to estimate the offsets directly from the pair of SAR images. The methodology uses a deep learning framework of the Attention[6] U-Net[7] architecture that automatically detects and corrects misalignment in complex radar images. The model processes paired SAR images to estimate the offsets in azimuth and range direction. The model is designed to handle high-dimensional data, noise, and complex spatial patterns. Incorporating the Attention mechanism enhances the model's ability to focus on relevant regions suppressing irrelevant background information.

Key contributions include:

1. Development of Attention-based U-Net architecture.
2. Creation of a custom data set containing normalized data and offset labels.
3. Implementation of the training strategy for optimal performance.
4. Evaluation of the model and visualization of the results.

Implementing Deep Learning methods for SAR image processing represents a cutting-edge solution for SAR image geometric correction in difficult scenarios where precise relative positions between acquisitions are not well known.

# Contents

# 1 Introduction

## 1.1 Background

### 1.1.1 SAR Imagery

**Synthetic aperture radar(SAR)** implements a simple method to capture images of various parts of the Earth's surface. It is not like the optical imagery method, which is a passive technique based on the energy emitted by the object/place. That means that the SAR images can be captured in a low-light condition, even at night, without the need for an external source of illumination. SAR imagery implements the use of high-energy radiations(radio waves), which interacts with physical objects like mountains, trees, etc., to create images. This concept is known as Radar(Radio detection and Ranging)[8], where the image is formed based upon the intensity of the reflected radio waves. The time taken by the wave to return to the antenna is also considered during the process. This feature of SAR is widely used in various applications, including disaster management such as oil spills, volcanic activity monitoring, landslide detection, and oceanography.

There are a lot of terminologies associated with SAR imaging. For SAR imaging, satellites and planes can be used to scan the area. For this thesis, consider the use of satellites alone. The satellite has an antenna and moves along a path called flight path. The antenna points at different angles from the nadir direction(perpendicularly downward) to capture various areas on the earth's surface. The look angle is the angle between the Nadir direction and the pointing angle of the antenna as illustrated in the Figure 1.1 below[9]. The image captured by SAR is usually an area of land whose width is called the swath. The area has 2 dimensions: one along the flight path called azimuth and the other one is the direction perpendicular to the flight path called range.

The main problem here is the antenna size, as the clarity of the image is directly proportional to the antenna size. Therefore, using a plane/satellite to capture images from afar will make the images blurry, without capturing much details. This phenomenon is Real aperture radar(RAR). In order to mitigate this problem, the antenna takes several tiny pictures as it moves along

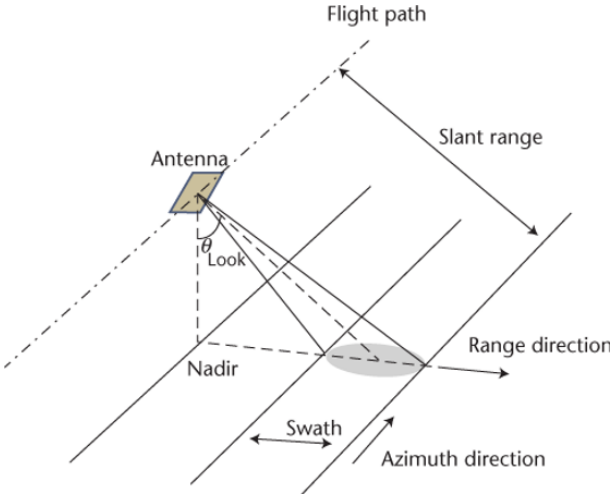



Figure 1.1: SAR Imaging

the azimuthal direction, and these pictures are combined to form a bigger and clearer image, simulating a longer antenna through synthetic aperture processing.

There are still many factors and phenomena that influence the final image obtained via SAR. As mentioned earlier, SAR uses high energy radiation(radio waves) to capture images. These waves have a wide range of frequencies, and different objects react differently with varying frequencies. These wavelengths are divided into different bands like X, C, L, etc. The choice of a band for SAR is highly crucial as it determines the energy associated with it and how much the wave can penetrate a particular object/medium like a tree, water, etc. Figure 1.2 [10]gives the overview of the SAR bands and their frequencies. In Figure 1.3, it can be seen that the bands

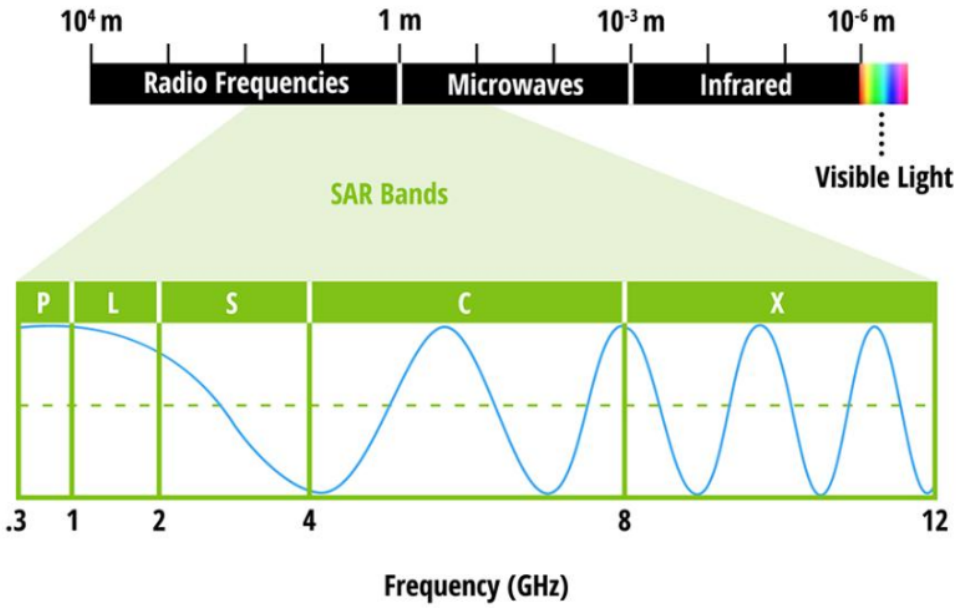



Figure 1.2: SAR bands

have different penetrating skills with the same object. While talking about the penetration of

the signal, we have to also talk about two other phenomena called scattering and polarization. **Scattering** refers to the way the radar signals react with the objects on the Earth's surface. These include surface, double bounce, volume and diffuse scattering in the order of decreasing intensity of the reflected wave. **Polarization** is the orientation of the plane in which the wave oscillates. Typically, the SAR sensors transmit linearly polarized waves. Otherwise, there can be horizontally(H) and vertically(V) polarized waves. The possible polarizations include:

1. Transmit vertical, receive vertical(VV),
2. Transmit vertical, receive horizontal(VH),
3. Transmit horizontal, receive vertical(HV),
4. Transmit horizontal, receive horizontal(HH).

Hence, the examination of the signal strength from the different polarization and their scattering reveals crucial information about the target surfaces. Example:

- **Urban areas**: Double bounce occurs between the ground and building, giving a strong VV and HH signal.
- **Water**: reflects most of the energy, so it usually appears dark in VV and HH.

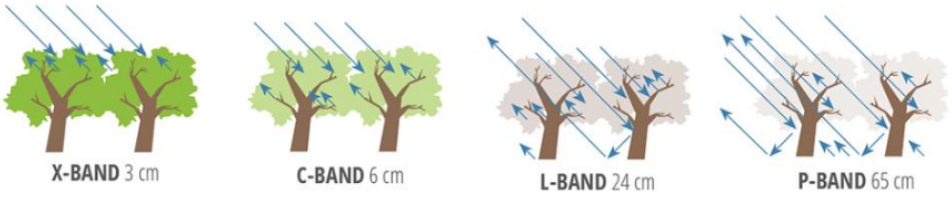



Figure 1.3: Comparison of different bands and their penetration ability [10]

Due to the side-looking nature of SAR imaging, three types of geometric distortions called foreshortening, layover and shadowing occurs. These happen in terrains whose surface has a high slope(deserts, mountains). **Layover** occurs when the top of a tall object appears closer to the radar than the base, making the image to "lean over". This occurs due to the signals reaching the peak of the object first and then the signal reaching the base of the object. This complicates the image overlapping, making it hard to interpret the shape of the object. **Foreshortening** is when the slope facing the radar appears compressed in the image, making it look shorter than the original length. This occurs when the wave from the slope reaches the radar first before the wave from the surface reaches back. Finally, **shadowing** occurs when a tall object blocks the signal from reaching the area behind the object, thereby creating a shadow where no signal is recorded. This makes the area appear dark or black, as there is no signal reflected from that area. The Figure 1.4 will give the comparison of these 3 phenomena.

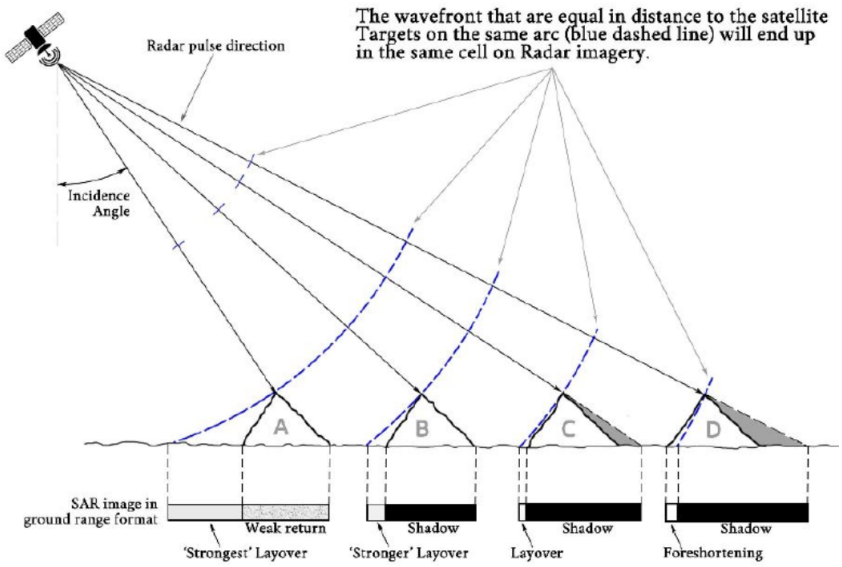



Figure 1.4: Geometric Distortions comparison
[11]

### 1.1.2 Speckle & Interferometry

With the introduction of scattering, another problem arises due to the scattering of the reflected wave. The waves don't reach back to the antenna, resulting in a noisy and grainy final image which is unclear. This grainy noise-like pattern is called **speckle**. This can occur not only due to scattering, but also due to coherent interference(which shall be explained later). It is not the actual noise but is a product of the surface object's reflectivity and properties. This reduces the quality and interoperability of the SAR images. To avoid this, de-speckling methods are implemented like spatial filtering, neural networks, etc.

With SAR images, there is another analysis method called interferometry, which, together with SAR, is called InSAR(Interferometric SAR). When two images are taken of the same area at different times and different angles, changes in the land topography can be detected up to millimeter precision, which are seen as ripples in the image.. This is referred to as the phase difference of the interferogram[10]. This has a wide range of applications, including monitoring of earthquakes[Figure 1.5], volcanoes, glaciers, etc. However, the challenges faced in InSAR are the complex data processing and the atmospheric effects.

### 1.1.3 Co-registration

Another important concept left in the introduction is the co-registration of two images. In order to have a clear image with pixel/sub-pixel accuracy, the primary and secondary images need to be properly aligned. This process is called co-registration. This step is crucial for InSAR

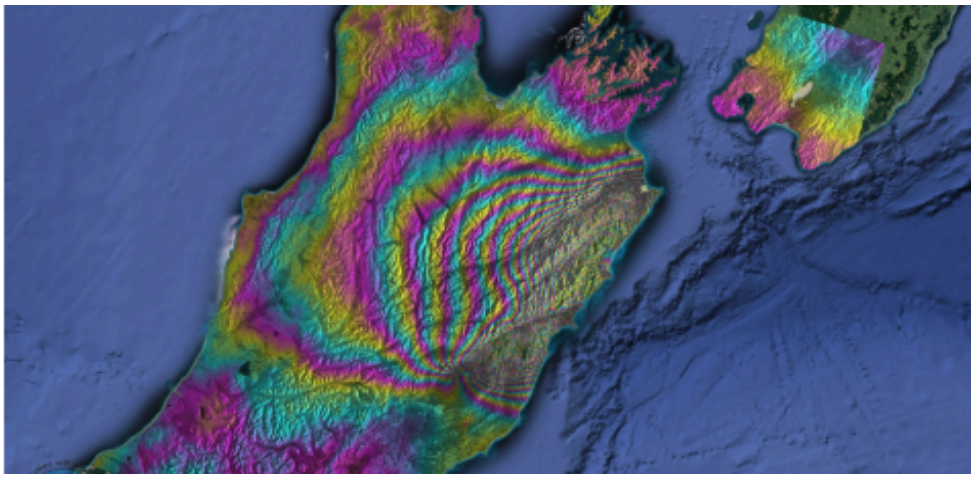


Figure 1.5: Interferogram after an earthquake in 2016 in New Zealand.
[12]

to create accurate interferograms. Without this, monitoring the earth's surface is almost an impossible task. The applications of co-registration include:

1. Noise reduction in SAR images,
2. Precise time-series analysis of SAR data,
3. Enabling coherent change detection(phase difference) between images.
4. Other multi-temporal and multi-sensor analysis.

As it is the most important step, this thesis shall focus solely on the co-registration process, its importance and the challenges faced during the processing of SAR images.

## 1.2 Motivation

In the process of co-registration, there is a crucial component called the offset maps(displacement maps). They represent the pixel displacements between two SAR images of the same area in range and azimuth directions. Meaning, they represent the shifts that are required to align one image over another. They are usually formed when a cross-corelation algorithm is applied onto a patch(small part of the image) that has strong features(buildings, roads) to estimate the local shifts between them. Their applications are very similar to the co-registration, which includes change detection, multi-sensor analysis, and deformation monitoring. The challenges faced in offset map creation include:

1. Temporal decorrelation: changes on the ground(human activity, vegetation) between image acquisition can make it difficult to estimate the offsets.

2. Speckle noise: This can degrade the quality of the image, making the offset estimation unreliable.

3. Large displacement: the large shifts due to rapid ground displacement between images due to earthquakes can cause the traditional correlation methods to fail.

4. Geometric distortions: SAR images suffer from distortions such as foreshortening and shadowing, which complicates further processing.

5. Computational complexity: Processing a pair of SAR images of high resolution can take many hours to even days.

6. Multi-sensor challenges: Aligning SAR images from different sensors is very difficult due to difference in wavelength, resolution and geometry.

7. Algorithm: the algorithms are sometimes not efficient in processing the highly complex data of SAR, making the accuracy at sub-pixel level very low and not reliable.

From the list of challenges, this thesis focuses on tackling the issues of speckle noise, algorithm and computational complexity to improve the efficiency of the final product of SAR.

### 1.2.1 Role of Deep learning in SAR processing

Recent advances in Deep Learning and Artificial Intelligence have shown that there is a high potential for its usage in processing SAR data processes like image feature extraction and pattern recognition. in self-supervised deep learning techniques, it is now possible to train large datasets without annotation and labeling. The U-Net architecture in deep learning with an attention mechanism can be an important solution to create offset maps with high reliability and accuracy. This makes deep learning a promising solution to overcome the challenges of traditional methods.

### 1.2.2 Problem Statement

To overcome the limitations and the shortcomings of the traditional SAR image processing methods, this research focuses completely to develop an automated, deep-learning driven solution that has :

1. **E**nhanced computational efficiency while maintaining high accuracy.

2. **Refined displacement map prediction** and speckle noise reduction.

3. **Sub-pixel accuracy** in both azimuth and range directions.

4. **Advanced feature extraction** globally and locally, with the use of advanced techniques of the residual network and the attention mechanism.

5. **Robust performance** with various kinds of datasets.

#### Research Focus

The main focus of this thesis is to research on developing an advanced attention-based U-Net model to generate SAR offset maps, integrating:

- Residual learning blocks to improve the model and to preserve relevant features.
- Multi-scale attention for capturing fine details in the image.
- Squeeze and Excitation blocks(SE) for a channel-wise architecture
- Data handling improvisation to optimize memory during the training,

The main goal is to create a more accurate, versatile and efficient solution for SAR offset estimation that surpasses the performance and scalability of the traditional methods.

## 1.3 Outline

This thesis is divided into 8 chapters. Each subsection is explained thoroughly in its respective chapter. This section provides an overall view of the chapters in the thesis with a short description and summary of the individual chapters.

1. **Chapter 1**: Gives a detailed introduction of the topic. This includes all the important concepts in SAR imaging, along with the motivation and the problem statement.

2. **Chapter 2**: Contains the literature review of SAR and Co-registration. This introduces the traditional approaches in offset map estimation and the current trends of deep learning including the evaluation metrics.

3. **Chapter 3**: The methodology that is followed in the thesis is explained in this section. This involves data preparation, model architecture and the setup for training and validation.

4. **Chapter 4**: Gives the implementation of the code, including the description of the critical parts of the code, environment and the resources.

5. **Chapter 5**: Concerns with the results of the model. The qualitative and quantitative results are presented here and ending with the analysis and interpretation of the results.

6. **Chapter 6**: Includes the challenges and the limitations that were presented during the implementation of the thesis work.

7. **Chapter 7**: The chapter of the thesis deals with the conclusion that summarizes the research and highlighting the impact of the thesis work.

8. **Chapter 8**: Gives information about the future works and the ideas that can extend the research to incorporate additional features and tools.

# 2 Literature Survey

## 2.1 Traditional approaches for offset estimation

Several conventional approaches have been used to estimate offset maps. The traditional methods are broadly divided into intensity-based and phase-based methods. Here are the methods:

1. **Intensity-based methods**

   These methods rely on the amplitude (intensity) information of the SAR images to estimate the offsets. This is used when the phase information of the image is unreliable or unavailable.

   (a) **Cross correlation(CC)**: It's based on the principle of sliding one reference image over another secondary SAR image and calculating their similarity. This involves computing the spatial cross-correlation between 2 images. The position with the highest similarity is their offset.

   $$R_{xy}(m,n) = \sum_{i,j} x(i,j) \cdot y(i+m, j+n) \tag{2.1}$$

   R represents the correlation between the reference image(x) and the secondary image(y), with (m,n) representing their offsets. The offset is identified at the peak of the correlation function.

   **Advantages**: Simple and robust.

   **Limitations**: Sensitive to noise and decorrelation effects.

   (b) **Normalized cross-correlation(NCC)**: It is similar to CC, but the brightness in the image is normalized to make it less sensitive to changes in intensity. This normalization ensures that the patches are comparable despite differences in absolute intensity.

   $$NCC(m,n) = \frac{\sum_{i,j}(x(i,j) - \bar{x})(y(i+m, j+n) - \bar{y})}{\sqrt{\sum_{i,j}(x(i,j) - \bar{x})^2} \cdot \sqrt{\sum_{i,j}(y(i+m, j+n) - \bar{y})^2}} \tag{2.2}$$

**Advantages**: More robust to intensity changes than CC.

**Limitations**: It is slower and computationally expensive than CC.

(c) **Mutual information(MI)**: This method works on the principle of the measure of how much information one image gives about the other. It looks at how one image's pixel value relates to the pixel value of the secondary image, and then it finds the offset that makes the relationship the strongest.

$$MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left( \frac{p(x,y)}{p(x)p(y)} \right) \tag{2.3}$$

Here, p(x,y) refers to the probability distribution of the pixel intensities of both the images. p(x) and p(y) are the marginal distributions. The output peaks at the optimal alignment.

**Advantages**: Works well for multi-modal images.

**Limitations**: The images have to be very similar and are subject to noise.

(d) **Feature-based matching**[13]: In this method, key points are found in both images, and matching happens. This is done via algorithms of **SIFT**(Scale-Invariant Feature Transform) and **SURF**(Speeded-Up Robust Features).

**Advantages**: Robust to noise and geometric distortions.

**Limitations**: Requires sufficient features in the images.

2. **Phase-based methods**

(a) **Interferometric phase analysis**: This method uses the phased differences between two images to determine the offset. Initially, the co-registration of the images is done, followed by the subtraction of the phase of the images and finally concluding with the conversion of phase difference into physical offsets.[14]

$$\phi = \arg \left( s_1 \cdot s_2^* \right) \tag{2.4}$$

where $s_1$ and $s_2$ are complex SAR signals from the primary and secondary images. $s^*_2$ represents the complex conjugate of the SAR image.

**Advantages**: High precision for sub-pixel offsets.

**Limitations**: Require high coherence and susceptible to noise easily.

(b) **Phase correlation**: This method computes the phase difference in the frequency domain to produce the offsets. The method starts with implementing the Fourier transform on the patches, followed by comparing the images for the offset generation.

**Advantages**: Robust to noise and wavelength variations.

**Limitations**: It is limited to small offsets.

(c) **Coherence Optimization**: This method works with the principle of finding the offsets to make the two images similar in terms of phase. This method tries different offsets, checks the similarities and chooses the offset with the highest similarity between the images.

**Advantages**: Effective for highly noisy and decorrelated images.

**Limitations**: It is computationally expensive.

a) **Hybrid Methods**

Hybrid models[15] combine intensity-based and phase-based techniques for offset mapping, which is an improvement over the individual methods. These include:

- **Intensity-Phase Cross-Correlation**
- **Multi-Resolution Approaches**

These methods are highly complex, and proper handling is required as the chances of failure are high. Hence, high skill is needed for properly integrating the methods.

## 2.2 Challenges in traditional methods

- **Noise and Decorrelation**: SAR images are often affected by speckle noise and decorrelation(temporal and geometric), which degrades the quality of the offset determination.[8]
- **Computational Complexity**: Many methods(Example: Coherence estimation, NCC) are computationally intensive, especially for large SAR datasets. They require manual tuning of parameters for accurate performance.
- **Sub-pixel accuracy**: Achieving sub-pixel accuracy in a very low-coherence region or regions which lack features is a very challenging task.

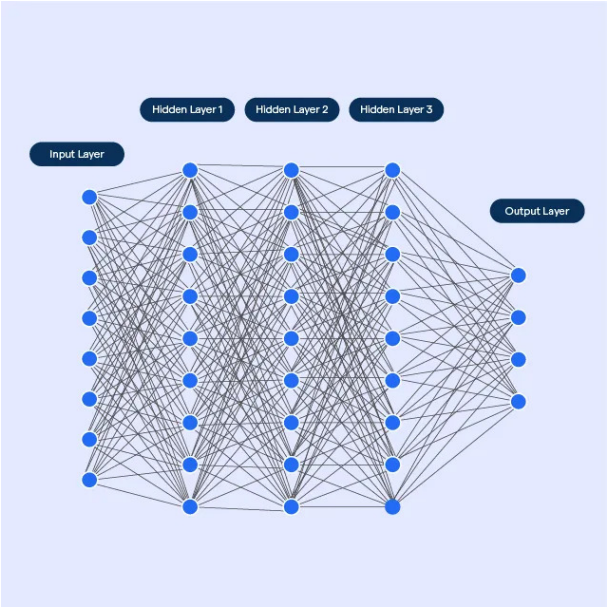



Figure 2.1: Neural Network architecture [1]

## 2.3 Deep Learning in Offset Estimation

The recent technological advancements have enabled machine learning and deep learning methods to have established roots across all technologies, including the fields of medicine and law and order. Especially, CNN have revolutionized image analysis with the implementation of feature extraction and learning to analyze better from the previous results. U-Net, a CNN-based architecture, has become a turning point in this analysis. It employs an encoder-decoder structure, and in-between, skip connections are employed to preserve important features globally and locally. Other advancements in the field include the concepts of residual learning and attention mechanisms, making it highly effective for complex tasks such as SAR image segmentation.

1. **U-Net and its variants**

   The original U-net architecture was designed for biomedical image[7] segmentation and introduced by Ronneberger in 2015. It enabled precise localization and segmentation of the objects with the help of a symmetric encoder-decoder structure. The encoder captures essential features by downsampling, and the decoder reconstructs these detailed features by upsampling. To preserve the fine details of the images, skip connections are introduced between the encoder-decoder structure, which enhances the model's learning ability. Thereby making it robust.

   To address the various limitations of the U-Net model, several studies have been conducted to design other variants. This led to the development of ResNets(2016)[16], incorporating residual blocks to tackle the issue of gradients. They also allow the model to train in deeper networks. Following the ResNets, Pyramid pooling network(2017)[17] was developed. It

was inspired by the U-Net model, which primarily focused on the semantic segmentation of images. During this time, the concept of attention networks gained high popularity, leading to the introduction of Attention U-Net[18] in 2018, implementing the attention gates and improving the model's accuracy.

2. **Attention mechanism and Multi-scale feature extraction**:

   As already mentioned, attention modules are very popular due to their high performance and efficiency in complex networks. These modules have become a powerful tool to improve CNN performance. The Squeeze and Excite (SE blocks) [19] method proposed in 2018 implements the attention module in the channel-wise feature response, which resulted in a powerful delivery model and is still trending.

   Multi-scale feature extraction is crucial for SAR image segmentation where both local and global features are important, thereby helping in giving fine-grained detail to the SAR images. In 2017, Chen introduced the atrous convolutions[19] to capture multi-scale information without losing the resolution of the images. As mentioned before, the pyramid pooling aggregates the multi-scale to capture global features. Hence, they have improved the efficiency of the segmentation.

3. **Phi-Net**:

   Recent advancements like **Phi-Net**[20] have proposed a novel architecture combining the efficient use of CNNs and attention-based transformers for the tasks of remote sensing. It involves a dual-axis mechanism where one axis works on extracting the local features using the CNNs, and then the other axis works on the extraction of global feature information using an attention mechanism. Phi-Net is highly modular, which enables easy integration with other existing SAR pipelines. Even though they pave the way for future works, the complex transformer-based methods are training-intensive and require extensive computational resources, which limits their practical application in SAR offset estimation.

Deep learning techniques have been applied in remote sensing for many years, even achieving state-of-the-art results in land cover classification(2018). Despite these advancements, multi-scale feature extraction in SAR image segmentation remains underexplored.

### 2.3.1 Research gap and Justification

While U-Net variants have shown remarkable success in SAR image segmentation, there are still some limitations that exist. Firstly, the attention gates in U-Net have been primarily focused on medical imaging, with limited exploration in remote sensing. Secondly, the existing studies

rely on single-feature extraction, which makes segmentation in SAR images highly challenging. Thirdly, a large annotated dataset of SAR is difficult to obtain and use in training models. Finally, the complexity of the transformer-based methods limits the application in real-time or large-scale application in image processing.

This research gap presents an opportunity to design an improved U-Net architecture specifically for image segmentation in remote sensing applications. This work focuses on enhancing U-Net, which offers a balanced trade-off between efficiency and performance. By integrating the SE blocks and multi-scale feature extraction in the model architecture, the accuracy of the SAR image analysis could be improved, which has always been a challenging task.

Table 2.1: Comparison of Offset Estimation Methods in SAR Imagery

| Method | Advantages | Limitations |
|---|---|---|
| **Traditional Methods** | | |
| **Intensity-Based Methods** | | |
| Cross-Correlation (CC) | Simple, robust | Prone to noise, decorrelation |
| Normalized Cross-Correlation (NCC) | More robust than CC | Computationally expensive, slower than CC |
| Mutual Information (MI) | Effective in multi-modal images | Sensitive to noise, require similar images |
| Feature Matching (SIFT/SURF) | Robust to noise and distortions | Need rich features in image |
| **Phase-Based Methods** | | |
| Interferometric Phase Analysis | High sub-pixel precision | Needs high coherence, sensitive to noise |
| Phase Correlation | Robust to noise and wavelength changes | works well with small offsets only |
| Coherence Optimization | Effective in low coherence | Computationally expensive |
| **Hybrid Methods** | Balances robustness and precision | Complex implementation, need high skill to handle |
| **Deep Learning Approaches** | | |
| U-Net | works well with small datasets | needs skip connections, struggles with high-resolution images. |
| Residual U-Net | improved training stability, versatile | High computation cost |
| Attention U-Net | improves accuracy, focus on relevant data | increased complexity, need lots of data |
| Pyramid Pooling | Captures multi-scale global features | High computational cost, introduces redundancy |
| SE Blocks | improved performance in CNN, enhance feature recalibration | low performance in some cases, extra parameters |
| Atrous Convolution | good semantic segmentation | high memory usage |
| Phi-Net | modular, accurate and fast | Training-intensive and resource-heavy |
| **Proposed Model** | Combine U-Net with SE blocks, multi-scale Attention feature | Yet to be validated experimentally |

# 3 Methodology

The next part will be about the methodology implemented and the tools and techniques used in the research.

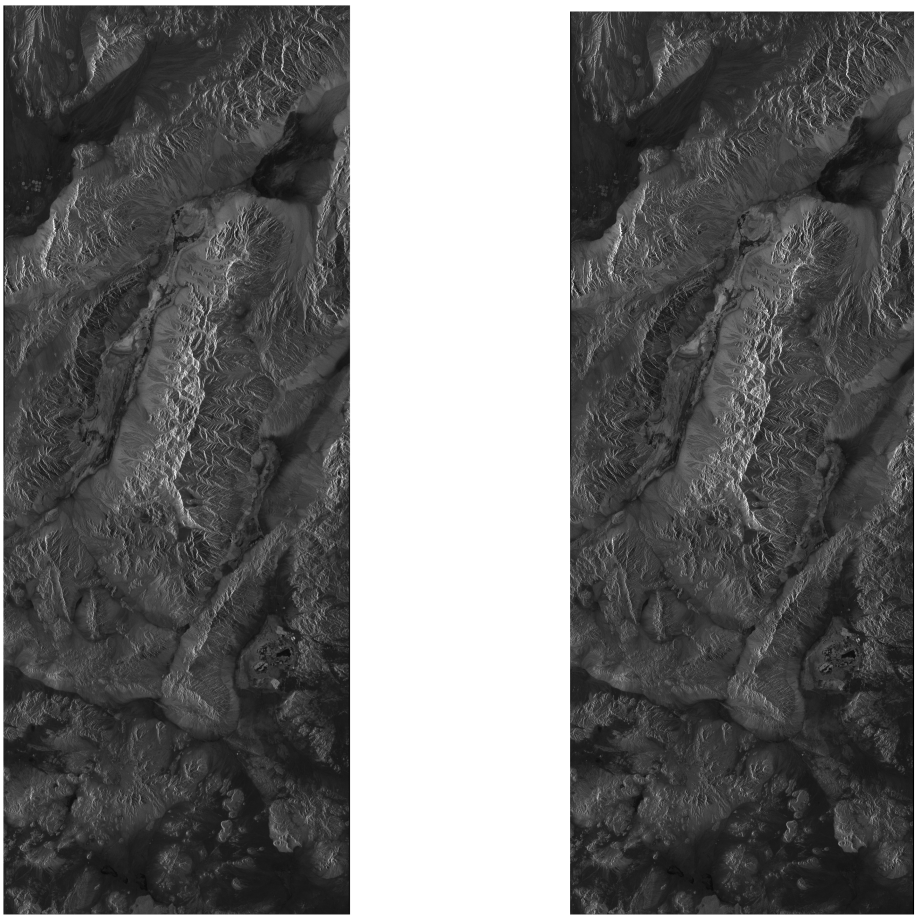
## 3.1 Overview of the approach

The main objective of the research is to develop an improvised U-Net architecture for the image segmentation of the SAR images. The architecture shall integrate techniques of SE blocks, attention blocks and multi-scale feature extraction. The proposed method addresses the limitations of SAR image analysis, including noise and complexity, by enhancing the efficiency of feature extraction and segmentation accuracy. The methodology involves the following steps:

- Preparation of the data,
- Design of the model Architecture,
- Training and Optimization of the model,
- Evaluation of the model and visualization,
- Implementation details.

### 3.1.1 Dataset Preparation

The dataset used in the study consists of SAR images from the **ARES**(Airborne Reflective Emissive Spectrometer), which receives the spectral reflectance signature of the Earth's surface that allows tasks such as mineral identification, monitoring vegetation mapping and assessing the water constituents. The dataset includes primary and secondary images [Figure 3.2], along with corresponding range(RG) and azimuth(AZ) offset maps. These offset maps are provided to give information about the ground truth data. The original data is in NumPy arrays format. The shape of the primary image is (48305, 18360), while the shape of the secondary image is (48292, 18360). Now this data undergoes several preprocessing steps before it is available for use in the training processes.

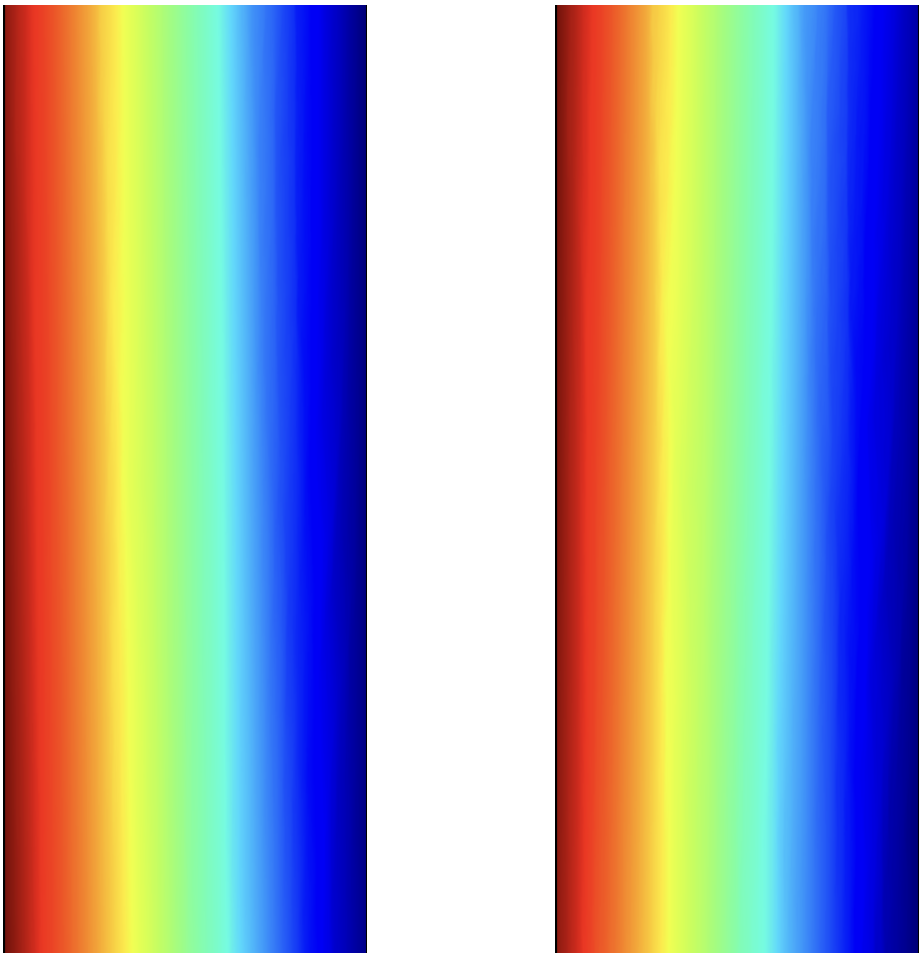


(a) Primary SAR Image (b) Secondary SAR Image

Figure 3.1: Original SAR Data

**Preprocessing steps**

1. **Cropping**:
    - As mentioned earlier, the primary and secondary images are of different shapes, so it is necessary to match their shapes to maintain their compatibility.
    - In this, the minimum row and minimum column sizes are maintained across all the input arrays.
2. **Magnitude Determination**:
    - The input data is filled with complex data, consisting of imaginary and real parts.
    - We need to convert this complex data for the magnitude representation(Figure 3.2).
3. **Normalization**:
    - This method is used to scale down the data to [0,1] for easier understanding and computation.



(a) Azimuth Offset map (b) Range Offset map

Figure 3.2: Original Offset maps

- The original minimum and maximum values are stored for later de-normalization purposes.

4. **Data Conversion**:
   - The data needs to be converted to **float32** datatype for compatibility with the TensorFlow API.
5. **Resizing**:
   - Since the data is large and with a random shape, it is highly necessary to resize the data to a standard size.
   - All the images are resized to the target size of **128*128** pixels.
6. **Channel stacking**:
   - In this step, the magnitudes of primary and secondary images are concatenated to form the input channel.

- The offset maps of azimuth and range are concatenated similarly to form the output channel, thus making a **two-channel model**.

7. **Dataset creation**:

   - The TensorFlow API is used here to create the dataset by initially converting the arrays into TensorFlow tensors.
   - Then it applies the shuffling with the buffer size of 1000.
   - Batch the data with a batch size of 32.
   - Enabling of prefetching for optimized performance.

### 3.1.2 Model Architecture Design

- **Encoder**: Initially, the data needs to be downsampled. For this purpose, three residual layers are implemented, followed by max pooling layers. The residual blocks implement the method of normalizing the data in batches and ReLU activation. The **skip connections** are added to mitigate the vanishing gradient problem. Each level of the encoder is followed by an increasing number of filters, i.e., from 32 to 64 and to 128 in the successive layers. This allows the model to capture complex features as it passes through many layers.
- **Bridge**: The bridge section of the model is used to connect the encoder to the decoder. The bridge consists of residual blocks with 256 filters. It is followed by a **dropout layer** with a value of 0.5. This means that half the output of the encoder is converted to 0, and this creates a blank area, which forces the model to learn better from the iterations.
- **Decoder**: As the data was downsampled previously, it needs to be upsampled to bring it back to the original form. For this purpose, there are similar number of layers in the decoder as there are in the encoder layer(3 layers). Each level employs transposed convolution layers to increase spatial dimension. It is then followed by the **multi-scale attention module**, which is concatenated with an improved residual block. This attention module is the key innovation in the model, which allows the model to concentrate on the most relevant features for accurate offset map determination. This layer applies 1*1, 3*3 and 5*5 convolutions with different dilation rates. Finally, it creates offset maps using the sigmoid function.
- **Output layer**: The final layer is a convolution layer that consist of 2 filters and linear activation. This produces the estimated offset maps in the azimuth and range directions.

**Other Custom Components**

- **PadToEven layer**: This layer ensures that the input dimensions are even for the upcoming processes like upsampling and downsampling.
- **SE Block(Squeeze and Excitation)**: This layer is applied to implement channel-wise attention. This uses the global average pooling to squeeze the spatial information into a channel descriptor. Two layers with ReLU and sigmoid functions help the model to learn and emphasize informative features and suppress the less useful ones.
- **Dimension matching**: A 1*1 layer of convolution is added to the residual connection if the number of input channels do not match with the number of output channels.

### 3.1.3 Training and Optimization

It is crucial for the model to provide accurate results. For this purpose, the model's training should be effective with a high learning capacity and with the help of the right optimization technique.

1. **Loss function**: The model is trained using the Mean Squared Error(MSE) loss function. This equation calculates the error between the predicted and actual value, and then it is squared. This punishes large errors more severely. Then, the average is taken for its final value. The Mean Squared Error (MSE) is calculated as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

   Where:

   - i is the sample
   - $n$ is the total number of samples in a batch,
   - $y_i$ is the actual value/ground truth,
   - $\hat{y}_i$ is the predicted value.

2. **Optimizer**: The model is optimized using the AdamW optimizer for the model. It combines AdaGrad and RMSProp, thereby giving a better adaptability to the learning rate. The optimizer has the following hyperparameter:

   - Learning rate

- Weight decay
- Batch size

3. **Training Procedure**: The dataset is split into training(80%) and validation(20%) sets. The model is run for 20 epochs, during which the model processes a batch on 32 samples, calculates the loss and the weights are updated through backpropagation. A callback is implemented to monitor the validation loss and to halt the training if no progress is observed for 10 consecutive epochs. This is called early stopping. The training process employs TensorFlow API, which provides distributed training, potentially utilizing the GPUs if available.

### 3.1.4 Evaluation and Visualization of the Model

Once the training of the entire dataset is completed, the model must be evaluated to determine its performance. Multiple approaches are implemented for this purpose:

1. **Loss metrics**: Once the training set is tested, the validation dataset is analyzed for its overall performance in the task of offset determination. This SAR dataset is analyzed by the MSE and MAE parameters. MAE(Mean Absolute Error) measures just the average of the difference between the predicted and actual values(ground truth), while the MSE squares the difference.

2. **Colormap evaluation**: To evaluate the performance of the model firstly, the primary and secondary images of the SAR need to be visualized. The actual offset map and the predicted maps are displayed next to each other to highlight the key differences between the images. These differences are not only examined visually but also quantitatively.

3. **Training History analysis**: To analyze the degree of convergence between the images, the loss curves are plotted. These include both validation and training plots. The MAE trend is analyzed to understand the stability of the model and its consistency over epochs.

The visualization function takes the data of the trained model, input labels and the number of samples to save an image which gives a general depiction of the model's performance. This also saves the original minimum and maximum values for denormalization, adds color bands before saving the file. This enables the user to analyse crucial data by identifying patterns, strengths, and potential areas needing improvement for a better model.

The above-mentioned methodology is implemented to ensure that the deep learning model is robust, scalable and efficient in the offset map prediction with regards to the large-scale SAR image processing application. This integration of complicated deep learning methods ensures that the model is highly capable of generalizing unseen SAR data, thereby enhancing its practical

usability in real-world scenarios. The flowchart on the workflow of the model is given in Figure 3.3.

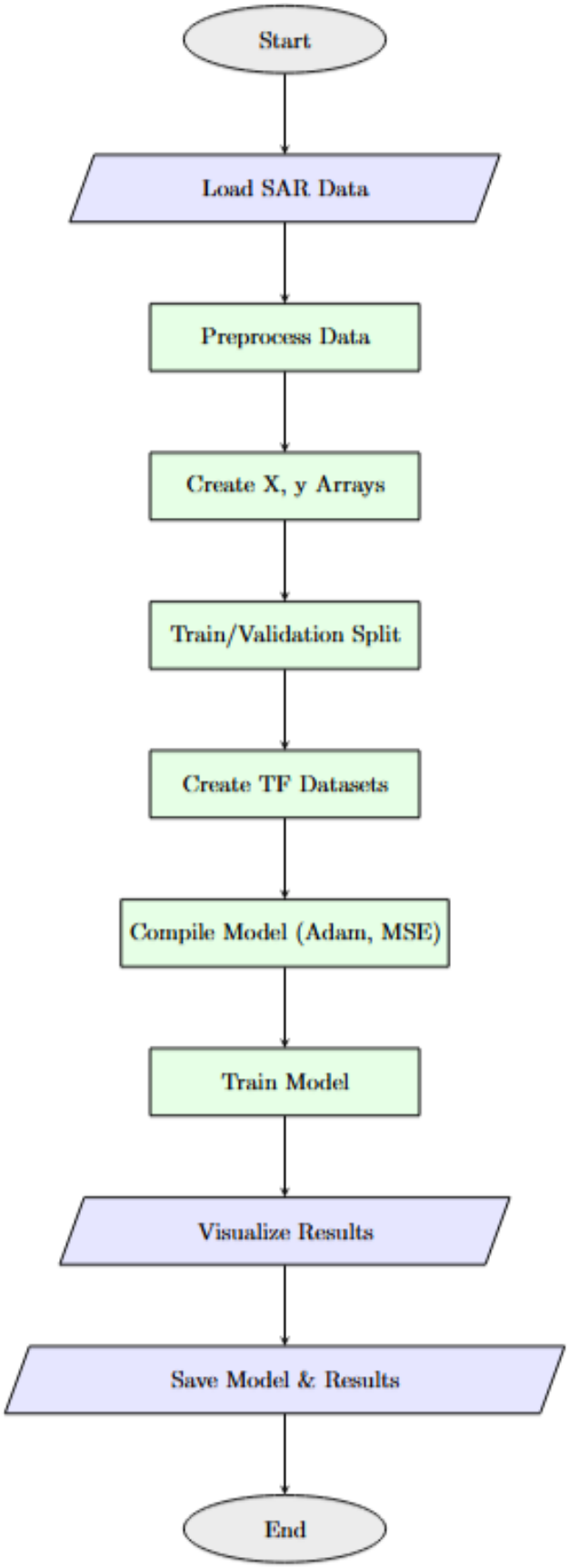



Figure 3.3: Flowchart of the model

# 4 Implementation of the Model

## 4.1 Environment Details:

The thesis research was modeled on a Linux-based server running on Ubuntu 24.04.1 LTS, equipped with an AMD Ryzen 9 5950X 16-Core Processor with 24GB VRAM and CUDA version 12.2. The model uses python programming language with the implementation on the TensorFlow, Keras API framework in a virtual environment. Docker Engine was used to create containers and for easy portability between various Operating Systems(OS). The following is the general summary.

**Tools and Framework**:

- **Operating system**: Linux, Ubuntu 24.04.
- **Programming Language**: Python.
- **Deep learning Framework**: TensorFlow 2.10.0, Keras API.
- **Editor Application**: Visual Studio Code.
- **Hardware details**: AMD Ryzen 9 5950X 16-Core Processor with RAM of 64GB.
- **Other**: Docker Engine 27.3.1.

In the Docker engine, several packages and libraries were installed with a separate requirement.txt file. Given below are the details:

**Installed Libraries**:

- **keras-tuner**: Helps the model to automatically find the right hyperparameter. It also provide various serach algorithms to search for the right hyperparameter.
- **tensorflow-addons**: It includes Advanced layers and activation functions that are not available in the core TensorFlow . It also provides additional evaluation metrics and optimizers(F1score, AdamW).

- **scikit-image**: Applicable in image processing tasks such as filtering, segmentation and data augmentation processes.
- **psutil**: It focuses on retrieving information on the running processes and utilization of system resources like CPU, memory).
- **NumPy**: Most important package for numerical computation with large collection of mathematical functions.
- **Matplotlib**: Data Visualization and plots.
- **OpenCV**: Image processing processes similar to scikit-image and also very efficient in terms of real-time applications.
- **scikit-learn**: Machine learning algorithms and data processing.

## 4.2 Model Architecture

Each methodology mentioned in the previous chapter is implemented using the Keras API. The main components of the model were implemented in different Python scripts. These files namely code.py, Dockerfile and requirements.txt. The main input files were stored in a separate folder in.npy format. These included:

- primary_image.npy
- secondary_image.npy
- offset_az.npy
- offset_rg.npy.

Initially, the model started with a simple U-Net with residual block, followed by constant upgrades leading to the Multiscale-Attention U-Net model.

The main model can be divided into 3 parts:

1. Data handling
2. U-Net modeling
3. training process

### 4.2.1 Data Handling

This part deals with converting the raw data of the SAR data into a suitable form that can be used by TensorFlow and the Keras API.

**Algorithm 1** Data Handling Pipeline

```
Function load_data from the folder
Function normalize(image)
    Compute mean and std of image
    return image
Function slice_to_patches(image, patch_size, stride)
    create empty list of patches
    Extract patch of size
    Append patches
    return patches
Function create_batches(pri, sec, rg, az, batch_size)
    create empty list of batches
    Extract batch of size
    Append batches
    return batches
Function reconstruct_from_patches(patches, shape, patch_size, stride)
    return reconstructed_image
```

#### Explanation of Data Handling Pipeline

Initially, the load_data function loads the data into the model from the given path to the folder. It is then followed by the normalize function which takes one .npy file at a time and normalize the data using the Min-Max method. The Min-Max normalization technique was applied to the data stored in the '.npy' file. Each data point $x_i$ in the original array $X$ was transformed to $x_{i,normalized}$ using the following formula:

$$x_{i,normalized} = \frac{x_i - \min(X)}{\max(X) - \min(X)}$$

where $\min(X)$ and $\max(X)$ represent the minimum and maximum values in the original NumPy array $X$, respectively. The resulting normalized data, where all values fall within the range $[0, 1]$, was then saved to a new '.npy' file.

In the same function, the mean and standard deviation are also measured and stored for future requirements. The mean value tells about the average value of the entire data, while the standard deviation tells about how much a particular value deviates from the mean value. The sample

mean is mathematically defined as:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

where,
$n$ is the number of data points in the matrix $X$,
$x_i$ represents each individual data point,
($\bar{x}$) is the mean of the sample,
($s$) represents the standard deviation.

The sample standard deviation is mathematically defined as:

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

where,
$n$ is the number of data points in the sample,
$x_i$ is each individual data point, and
$\bar{x}$ is the sample mean,
$s$ is the standard deviation.

After this step, the large images are divided into small patches of size 128*128 to maintain a standard pixel size. This makes processing easier and reduce the stress on system resources. Initially, an empty list is made and a slide window moves across the image that slices the large image into small patches and then concatenation occurs in the empty list, returning the created patches. After this, similar steps are implemented for the creation of batches. Finally, the final map is created by merging the created patches and batches using the weighted-average method(some data-points are given more preference).

### 4.2.2 Model Training

#### Explanation of Model Training pipeline

Firstly, the created patches, batches, epoch, optimizer and loss functions are loaded into the function as inputs. Then the looping starts, wherein every batch is unpacked(primary, secondary, rg and az). These unpacked data are passed through the created model to predict the offset maps. After this, the loss is computed and back-propagated, to update the parameters of the model for the next iteration. After the training is done, looping through validation set to determine the validation loss is implemented(without updating the weights). Optionally, the logs of the model and the losses are saved in the end.

**Algorithm 2** Model Training Pipeline

```
Function train_model(train_batches, val_batches, model, epochs, optimizer, loss_fn)
for epoch = 1 to epochs do
    Initialize train_loss = 0
    for each batch in train_batches do
        Unpack batch into primary, secondary, rg, az
        prediction = model(primary, secondary)
        loss_rg = loss_fn(prediction.rg, rg)
        loss_az = loss_fn(prediction.az, az)
        total_loss = loss_rg + loss_az
        Backpropagate total_loss
        Update model weights with optimizer
        Add total_loss to train_loss
    end for
    Compute average train_loss
    Initialize val_loss = 0
    for each batch in val_batches do
        Predict using model
        Compute validation loss
        Add to val_loss
    end for
    Compute average val_loss
    Log or save model and losses
end for
return trained model
```

### 4.2.3 Offset Map Reconstruction

**Algorithm 3** Inference and Offset Map Reconstruction

1: **Function** `predict_offsets`(test_image_pri, test_image_sec, model, patch_size)
2: Initialize `full_rg_map` and `full_az_map` with zeros
3: Divide input images into patches of size `patch_size`
4: **for** each patch_pri, patch_sec at position (i, j) **do**
5: `pred_rg, pred_az = model(patch_pri, patch_sec)`
6: Place `pred_rg, pred_az` into corresponding position in `full_rg_map` and `full_az_map`
7: **end for**
8: **return** `full_rg_map`, `full_az_map`

**Explanation of the Inference and offset map reconstruction algorithm**

The function first takes the trained model, patches and normalized images as inputs. To solve the issues of memory, both the images are divided into smaller patches. Once the patches are created, the loop is started to stack them into channel-wise inputs for the model. They are groped into batches to predict the offset maps in mini-batches. For each mini-batch, predictions are made to determine the maps and are temporarily stored. After the small predicted offsets are made, these are merged back together to form the complete offset map of the input images. The output gives the full-sized offsets and these files are saved in the folder for future analysis.

### 4.2.4 Evaluation and visualization of Offsets

After inference, the model's predictions are evaluated using the metrics, followed by the visualization plots to compare the results with the ground truth. This give the crucial information about the models performance and the ways in which the model can be improved.

**Algorithm 4** Evaluate and Visualize Offset Maps

1: **Function** `evaluate_predictions`(pred_rg, pred_az, gt_rg, gt_az)
2: Compute `MAE_rg` = Mean Absolute Error between `gt_rg` and `pred_rg`
3: Compute `MAE_az` = Mean Absolute Error between `gt_az` and `pred_az`
4: Compute `RMSE_rg` = Root Mean Squared Error between `gt_rg` and `pred_rg`
5: Compute `RMSE_az` = Root Mean Squared Error between `gt_az` and `pred_az`
6: Plot predicted vs. ground truth maps for RG and AZ offsets
7: **return** `MAE_rg`, `MAE_az`, `RMSE_rg`, `RMSE_az`

**Explanation of the Evaluate and Visualize Offset maps algorithm**

The function takes the predicted values and the ground truths as inputs. Then the function computes the losses(RMSE and MAE) between the predicted offsets and the ground truth. After the analysis, the plots are created and saved for visualization and future analysis.

# 5 Results and Analysis

## 5.1 Data Statistics

The data used in the thesis research includes two primary and secondary images with their respective offset maps in range and azimuth direction. They are processed to match the shaped for easier processing. The details of the input data are given in Table 5.1.

| **Metric** | **Value** |
|---|---|
| Primary Images Shape | (48305, 18360) |
| Secondary Images Shape | (48292, 18360) |
| RG Offsets Shape | (48305, 18360) |
| AZI Offsets Shape | (48305, 18360) |
| Cropping Dimensions | (48292, 18360) |

Table 5.1: Raw Data Statistics

The range of values that these input files contain is given in Table 5.2. These values should be normalized between the range [0,1] for easy processing and analysis.

| **Metric** | **Range** |
|---|---|
| Primary Images | [0.0000, 3274.3396] |
| Secondary Images | [0.0000, 3469.2307] |
| RG Offset | [-18.5886, -16.9046] |
| AZI Offset | [24.6866, 25.0061] |

Table 5.2: Normalization Ranges

After this process, various processing tasks occur, including cropping, data augmentation, channel-wise concatenation, splitting the data and batch-wise concatenation. The values of the data after these processes are given in Table 5.3. In this table, the **cardinality** represents the number of batches in the given dataset. This table also displays the data values after splitting the data with 80% of training and 20% of testing. Finally, show the shape and value range of the batches.

The value (48292,128,128,2) in processed X shape infers that there are a total of 48292 samples present, each having a dimension of 128*128 shape and are in 2 channels, i.e., primary and

Table 5.3: Processed Data Statistics

| **Metric** | **Value** |
|---|---|
| Processed X Shape | (48292, 128, 128, 2) |
| Processed y Shape | (48292, 128, 128, 2) |
| X Range | [0.0000, 0.2911] |
| y Range | [0.0027, 0.9962] |
| Training Samples | 38633 |
| Validation Samples | 9659 |
| Training Dataset Cardinality | 1208 |
| Validation Dataset Cardinality | 302 |
| Approximate Training Samples | 38656 |
| Approximate Validation Samples | 9664 |
| Batch X Range | [0.0000, 0.1051] |
| Batch y Range | [0.0027, 0.9962] |
| Batch X Shape | (32, 128, 128, 2) |
| Batch y Shape | (32, 128, 128, 2) |

secondary. In the batch shape, we can see that the entire dataset is divided into 32 samples per batch, and each having a shape of 128*128 with 2 channels.

## 5.2 Model Performance

The analysis of the model performance is described in detail in this section.

### 5.2.1 Visualization and Quantitative Analysis

In the Figure 5.1, we can see that the prediction was giving accurate predictions for the offset, within a very close range of the ground truth(0.03-1.39). The final validation loss was 0.003 and the training loss was close to 0.0001. This shows that the model has learned accurately in the training. The MAE for the final training set is 0.0033 and the MAE for the final validation is 0.0079. This low error shows that the model has a strong generalization performance. As mentioned earlier, there is no standard for the metrics used to measure the Deep Learning models, it is more of the visual comparison with predicted image versus the ground truth. The overall execution of the model took about 9 hours, 24 minutes, and 10.5 seconds. Each epoch took an average of 3050 seconds with 3 seconds/step in each epoch. The final plots of the model took 399 milliseconds to be generated.

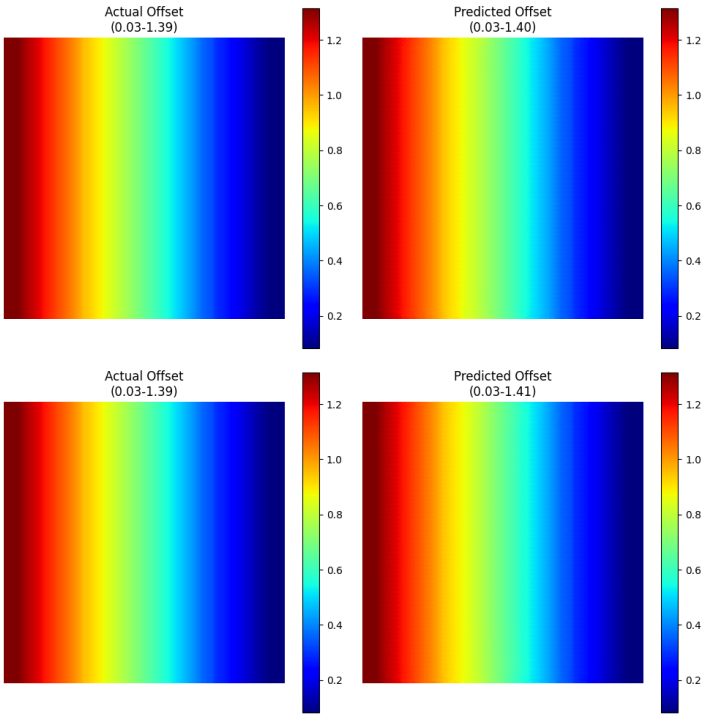



Figure 5.1: Comparison plot of the model

### 5.2.2 Training History and Convergence

The training diagram is depicted in Figure 5.2 shows a consistent decrease in training and validation losses over the epochs. The model was trained for a maximum of 20 epochs, but due to stagnation in validation performance, the early termination function was triggered. In the left subplot, the training loss (blue plot line) shows that the model effectively learns the patterns with the training data. In contrast to the previous part, the validation loss (orange plot drama) shows greater instability and plateaus, indicating a divergence in performance between seen and unseen data. In the right subplot, the MAE trend shows that the model has improved prediction accuracy on the training data (blue line), but the validation set has problems due to the complexity of the unseen data.

## 5.3 Ablation studies

In this section, the model is analyzed by removing different parts of the model and comparing the results with the other models. This shows how each part affects the overall performance of the model. The crucial parts of the model includes:

- U-Net
- Residual Block,

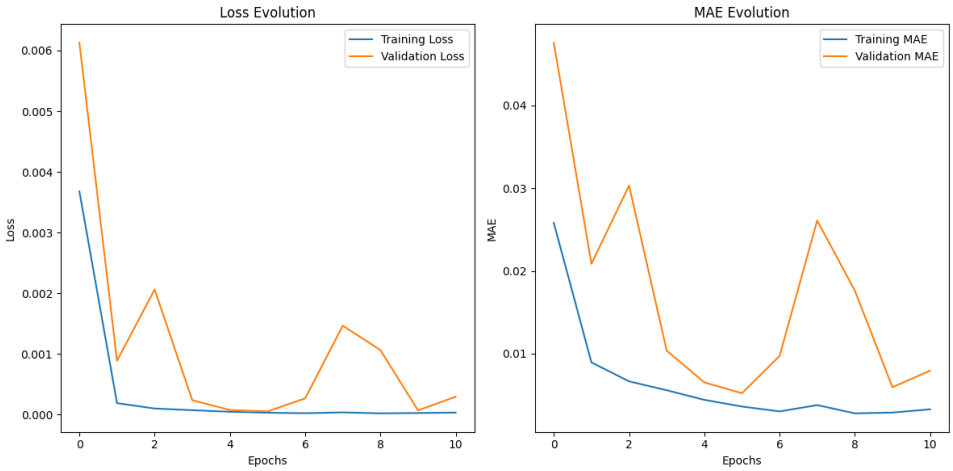



Figure 5.2: Training history of the model

- SE block,
- Multi-scale attention unit.

### 5.3.1 Simple U-Net model

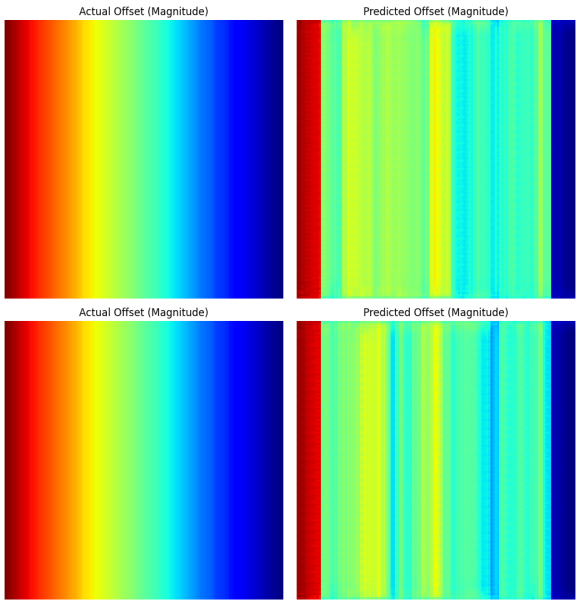



Figure 5.3: Comparison plot of Simple U-Net

In this model, a simple U-Net model is implemented without any residual blocks and attention blocks. The training plot is visualized in Figure 5.4, and the comparison plot between the ground truth and the predicted offset map is given in Figure 5.3. In the Figure 5.4, we can see that the model follows the same trend as the original model, where the training curve shows constant decrease while the validation has distortion and shows instability. The model ran for 7 epochs

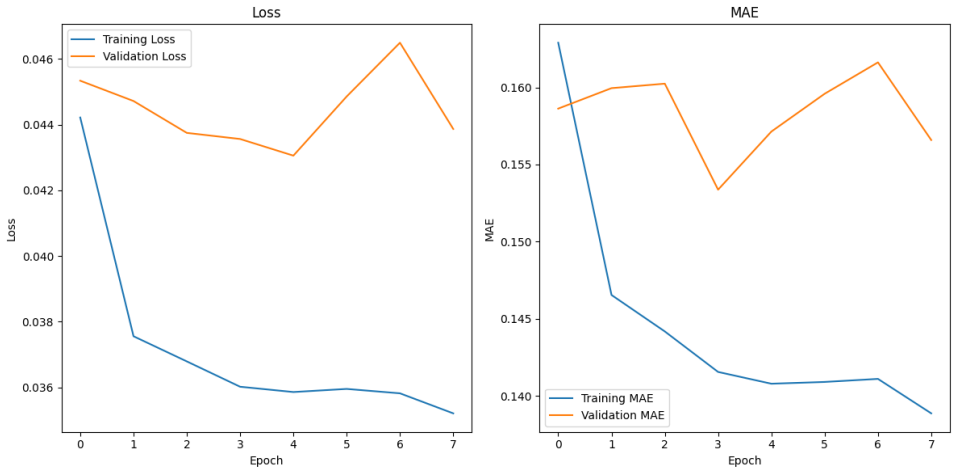



Figure 5.4: Training history of Simple U-Net

before the early callback function was triggered. In Figure 5.3, we can see that the predicted offset is showing poor convergence and having no clear similarity with the ground truth maps. Even though the metrics show that the model has a good learning curve, it is not being able to identify crucial features of the ground truth, giving bad predictions. The number of trainable parameters were 530,274, with the allocation of 5063704576*4 bytes ~4.74 GB * 4 = 18.96 GB using almost 40% of the free system memory. The model took approximately 2200 seconds per epoch with each step taking 2 seconds. The entire model ran for 4 hours 42 minutes and 32 seconds to run.

### 5.3.2 U-Net with Residual block

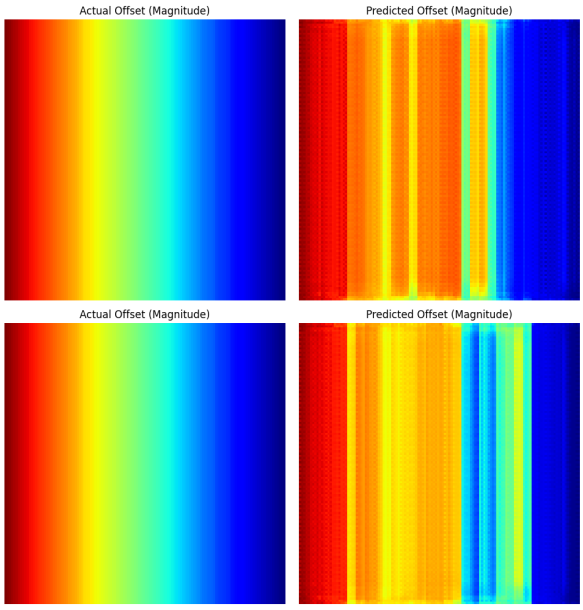



Figure 5.5: Comparison plot of U-Net with Residual Block

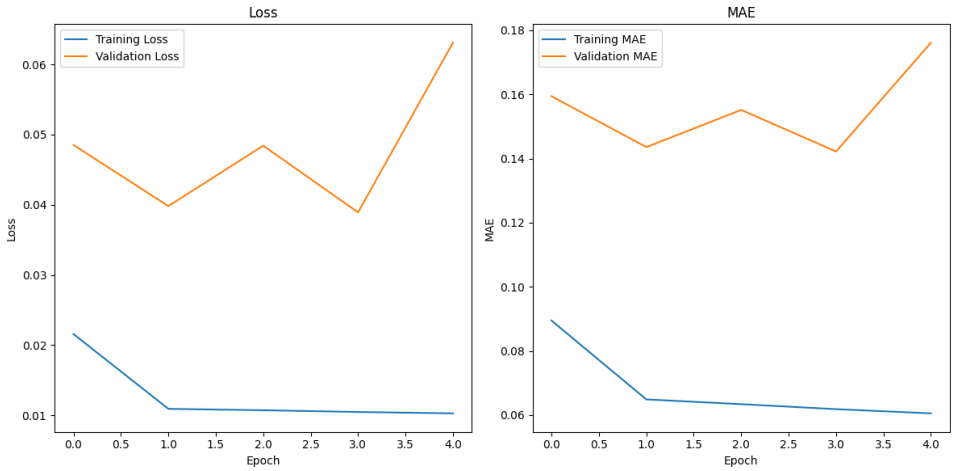



Figure 5.6: Training History plot of model with Residual block

In this model, the U-Net is integrated with Residual blocks to compare the results with other models. This model includes residual block layers with a dropout of 0.2. In the Figure 5.6, the training curve on both the sub-plots almost reaches plateauing in the end which shows that it has reached a global or local minimum and there is no room for learning and improvement.

In the visualization plot in Figure 5.5, there is some similarity showing that the model has good learning and is able to read important features from the source image. But still the model is not able to capture all the relevant features needed to accurately predict the offset.

### 5.3.3 Model without Multi-Scale Attention Module

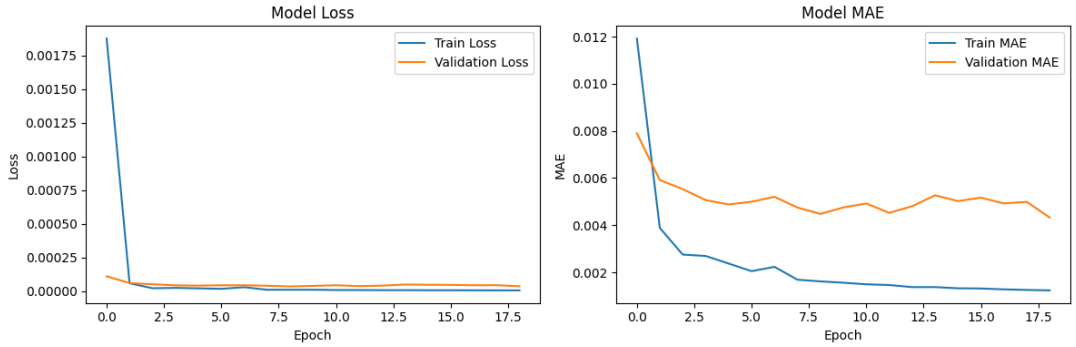



Figure 5.7: Training history of Model without Multi-Scale Attention module

In this part, the SE blocks were removed from the main model. In Figure 5.7 in the left subplot, the training loss was drastically reduced during the second epoch and then the loss remained constant and plateaued, indicating that the model learned effectively. Similar results were also seen with the validation loss plot, indicating a good generalization. In the right subplot, MAE was similar to the model loss but with a few fluctuations and finally plateauing, indicating a gradual learning process in the training curve.

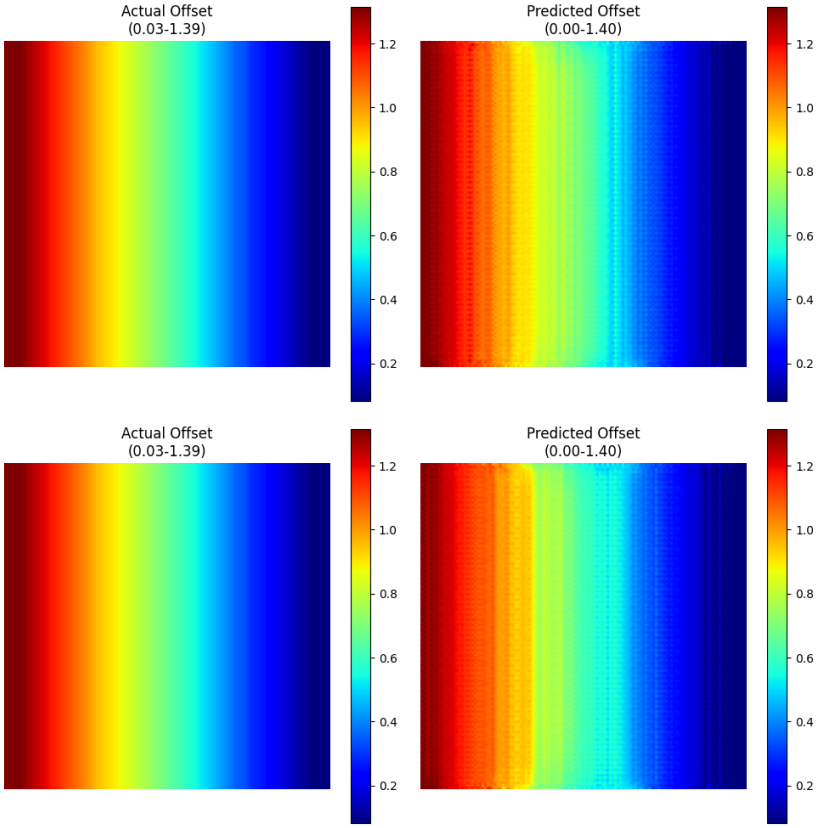
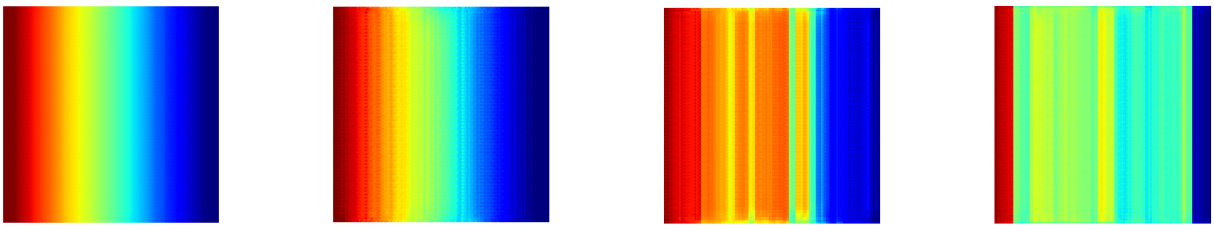



Figure 5.8: Final results of the model without Multi-Scale Attention module

In Figure 5.8, we can see that the visualization was good. The model was able to capture crucial information and features from the ground truth. The colors were predicted correctly, and offset maps were generated with an accuracy of 99%. The only difference is the rough lines in the predicted offset map, whereas the ground truth has fine lines with a smooth contrast between different colors.

Figure 5.9 shows an overview of the ablation studies of different models in the order of their decreasing complexity.



(a) (b) (c) (d)

Figure 5.9: Overall Results of the ablation studies,
(a):*Multi-Scale Attention model*, (b):*Model without Multi-Scale module*, (c):*U-Net with Residual Blocks*, (d):*Simple U-Net model*

# 6 Challenges and Limitation

The design and implementation of the research faced many challenges. The details are given below.

## 6.1 Data-Related Challenges:

1. **Noise and Data quality**:
    - Inherently, SAR data contains noise. This complicates the process of determining the offsets.
    - The biggest challenge was the lack of labeled data for the implementation of supervised learning, leading to semi-supervised or self-supervised learning.
    - Decorrelation of the data with temporal and geometry can lead to the misalignment of the input images.
    - Ignores the needs of atmospheric corrections and physical constraints.
2. **Challenges in Preprocessing:**
    - Several techniques are needed to convert the complex SAR data consisting of real and imaginary parts to a suitable format that can be used in deep learning models.
    - Although normalization is a crucial step, this step removes the phase information in complex SAR data, which is relevant for displacement analysis.
    - Other processes of cropping, alignment, and handling varying resolution of the input data was challenging. Any misalignment can lead to poor performance of the model.
    - Robust scaling methods are needed when there are extreme values in the offset maps.
3. **Memory and Computational issues:**
    - Processing the entire data set at once was not a feasible task due to the high-resolution SAR images.

- The system frequently encountered **out-of-memory** errors during the Attention U-Net training.
- To counteract memory issues, the entire SAR dataset was divided patch-wise, then processing with batch-wise training strategy.
- Batch processing also leads to the loss of finer details. Hence, fine-tuning the parameters was time-consuming.

4. **Evaluation Metrics and Validation:**
   - Choosing the correct evaluation metric for SAR offset determination was difficult, as there are no standardized metrics.
   - MSE/MAE metrics do not reflect the perceptual quality of the offsets.
   - Validating the model's effectiveness in the real- world applications is challenging.

## 6.2 Model Architecture Limitations:

1. **Attention Mechanism bottleneck**:
   - Attention gates are connected to only the adjacent encoder/decoder, and there is no cross-scale interactions.
   - Simple attention gates may not capture large-scale deformation patterns in large images.
   - 3-level depth provides only 8x downsampling.
   - Focus per channel leads to a lack of spatial attention.
2. **Generalization to other datasets:**
   - Trained model usually fails in unseen terrains(urban vs agriculture).
   - Seasonal changes degrade the model's performance.
3. **Patch-wise processing:**
   - Conversion of entire data to 128*128 pixels made the computation efficient, but may lead to the loss of contextual data at a broader scale.
   - Estimated offsets could be locally accurate but can lack global coherance.
   - A batch size of 32 with 128*128 requires at least 12 GB VRAM.

4. **Limited use of Phase Informtion:**

   - This work is only concerned with the co-registration based on amplitude due to the complexity of handling phase discontinuities and noise.

Table 6.1: Categorized Challenges in SAR Offset Estimation

| Level | Challenge |
|---|---|
| **Critical** | • Out-of-memory errors during training.<br>• Lack of labeled SAR data.<br>• Absence of a standard metric for offset map determination.<br>• Loss of information due to patch-wise processing.<br>• No-Inclusion of phase information. |
| **Moderate** | • Generalization issues.<br>• Variable results due to seasonal and acquisition differences.<br>• Attention bottlenecks.<br>• Challenges related to normalization, resizing, and phase loss |
| **Minor** | • Limited data augmentation techniques.<br>• Instability in the training.<br>• Complexity in Docker setup. |

# 7 Conclusion

This thesis presented an innovative approach in Deep Learning-based technology for the application in Co-registration of SAR images. This method provided a practical solution without the need of prior acquisition geometry. The novel approach of Multiscale Attention U-Net surpasses the traditional and baseline approaches by a huge margin. With the integration of residual blocks, SE blocks and multi-scale attention modules, the model showed significant performance in the estimation of offset maps in azimuth and range direction. Despite the challenges of limited labeled data and handling large SAR data, the model demonstrated high robustness and generalization. Beyond achieving high accuracy, this research opens doors for next generation SAR image analysis for faster, scalable, autonomous systems, and real-time analysis.

# 8 Future Scope

## 8.1 Architectural improvements:

- Incorporation of phase information in the determination of offset maps.
- Hybrid CNN-transformer models for advanced feature learning.
- Implement multi-task learning to determine offset and coherence maps.
- Implementation of physics-informed Neural Networks(PINNs) accounting orbital parameters, atmospherics delays.

## 8.2 Advanced training strategies:

- Future works can completely focus on fully-unsupervised learning.
- Transfer learning and domain expansion to learn from one dataset to another, thereby improving generalization.
- Implement active learning so that the model requires less labelled dataset for accurate estimation of offsets.

## 8.3 Expansion of the Application domain:

- Rapid SAR data processing for monitoring real-time displacement estimates for disaster response like infrastructure, climate change, etc.

## 8.4 Model Optimization and Edge deployment:

- Quantized and a compact model for onboard processing in satellites and other autonomous systems.
- Optimizing the network with pruning techniques of ONXX, TensorRT for better model performance and accurate results.

Table 8.1: Future Work: Short-Term vs Long-Term Roadmap

| **Timeframe** | **Future Scope Points** |
|---|---|
| **Short-Term** | • Integration of phase information into the model.<br>• Attention modules with spatial perception and cross-scale.<br>• Application of transfer learning across different data sets.<br>• Application of advanced augmentation.<br>• Use synthetic data sets for more robust performance |
| **Long-Term** | • Deployment of lightweight models on edge devices.<br>• Exploration of fully unsupervised/self-supervised pipelines.<br>• Implementation of CNN-transformer hybrid architectures<br>• Implementation of multi-task learning to predict offset and coherence maps.<br>• Implementation of constraints via Physics- Informed Neural Networks. |

# Bibliography


[1] V. Special, "Synthetic aperture radar (sar)." Accessed April 23, 2025.

[2] Y. K. Chan and V. Koo, "An introduction to synthetic aperture radar (sar)," *Progress In Electromagnetics Research B*, vol. 2, pp. 27–60, 2008.

[3] R. Gens and J. L. Van Genderen, "Review article sar interferometry—issues, techniques, applications," *International journal of remote sensing*, vol. 17, no. 10, pp. 1803–1835, 1996.

[4] F. Ayoub, S. Leprince, and J.-P. Avouac, "Co-registration and correlation of aerial photographs for ground deformation measurements," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, no. 6, pp. 551–560, 2009.

[5] I. Goodfellow, "Deep learning," 2016.

[6] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015.

[8] R. F. Hanssen, *Radar interferometry: Data interpretation and error analysis*. Springer Science & Business Media, 2001.

[9] C. Space, "Sar 101: An introduction to synthetic aperture radar," 2020.

[10] N. Earthdata, "Sar (synthetic aperture radar) [earth observing system]."

[11] Y. Qin, E. Hoppe, and D. Perissin, "Slope hazard monitoring using high-resolution satellite remote sensing: Lessons learned from a case study," *ISPRS International Journal of Geo-Information*, vol. 9, p. 131, 02 2020.

[12] COMET, NERC, "Licsar service: Maps of deadly earthquake zones," 2023.

[13] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[14] H. A. Zebker and R. M. Goldstein, “Topographic mapping from interferometric synthetic aperture radar observations,” *Journal of Geophysical Research: Solid Earth*, vol. 91, no. B5, pp. 4993–4999, 1986.

[15] G. Fornaro and G. Franceschetti, “Image registration in interferometric sar processing,” *IEE Proceedings-Radar, Sonar and Navigation*, vol. 146, no. 6, pp. 331–340, 1999.

[16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[18] O. Oktay, J. Schlemper, L. F. Folgoc, M. Lee, M. Heinrich, K. Misawa, *et al.*, “Attention u-net: Learning where to look for the pancreas,” *arXiv preprint arXiv:1804.03999*, 2018.

[19] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[20] F. Sica, G. Gobbi, P. Rizzoli, and L. Bruzzone, “-net: Deep residual learning for insar parameters estimation,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 5, pp. 3917–3941, 2021.