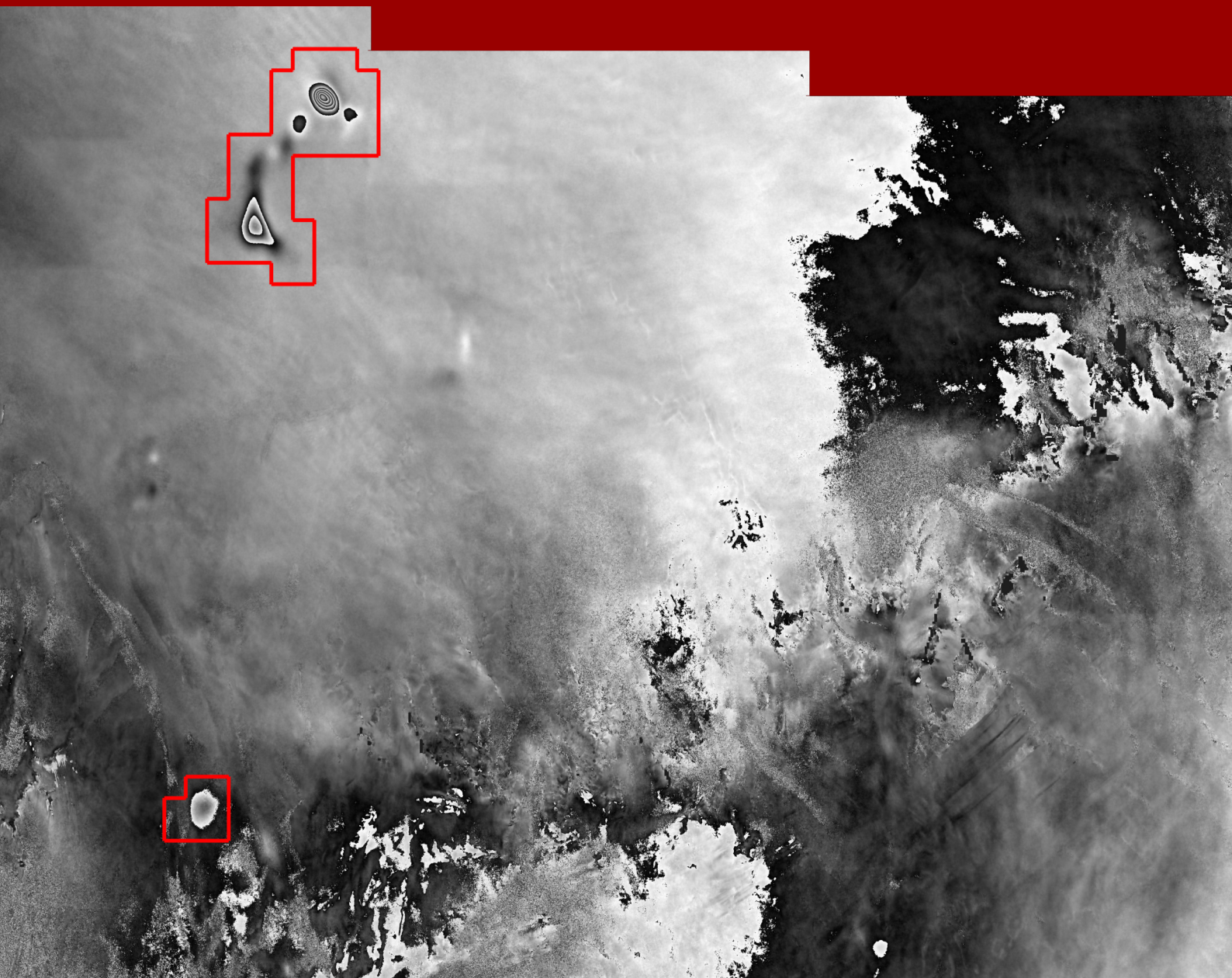
Investigating subglacial hydrology in Antarctica using InSAR measurements and machine learning methods

Master Thesis



Investigating subglacial hydrology in Antarctica using InSAR measurements and machine learning methods

Master Thesis
July, 2025

By
Niels Peter Kirkegaard Christensen

Cover image: Chunks detected by SGLNet at Jutulstraumen Glacier. The double difference phase interferogram is obtained using Sentinel-1 data from track 002 ascending, frame 925, covering dates 01/01/2020–13/01/2020–25/01/2020. The network configuration used is ViT-B/16 Phase/224.

Published by: DTU, Department of Space Research and Space Technology, Ørsteds Plads, Building 348, 2800 Kgs. Lyngby Denmark
www.space.dtu.dk

Approval

This thesis has been prepared over six months at the Division of Microwave and Remote Sensing, Department of Space Research and Technology, at the Technical University of Denmark, DTU, and in collaboration with the Department of Geosciences and Natural Resource Management, at the University of Copenhagen, KU, in partial fulfilment for the degree Master of Science in Engineering, MSc Eng.

Niels Peter Kirkegaard Christensen - s203980

.....

Signature

.....

Date

Abstract

In this study, we apply machine learning (ML) to Sentinel-1 interferometric synthetic aperture radar (InSAR) data to investigate subglacial hydrology in Antarctica. The dynamics of this system remain poorly understood and represent a significant source of uncertainty in glaciological models. We develop an algorithm called SGLNet to detect spatial phase patterns in interferograms arising from vertical surface displacements associated with transient subglacial water flow. SGLNet combines a vision transformer (ViT) backbone, pretrained using Self-Distillation with No Labels (DINO), with a custom multilayer perceptron (MLP) classification head. We evaluate 32 model configurations for binary classification, event detection, and semantic segmentation. Our best model achieves F_1-scores between 0.65 and 0.94, detecting 71% of 192 test events. A prototype attention-thresholded segmentation approach reaches a Dice coefficient of 0.39. SGLNet identifies over 2,000 events across selected regions of Antarctica and reveals that the subglacial hydrological system is highly dynamic near grounding lines. With SGLNet, large datasets can be processed and used to improve glaciological models, which is important for forecasting the evolution of Antarctica in the future.

Acknowledgements

John Peter Merryman Boncori, Head of Microwaves and Remote Sensing, DTU Space
Have been the primary supervisor for this thesis and have provided academic guidance throughout the project.

Jonas Kvist Andersen, Postdoc, KU IGN
Have been the host supervisor for this thesis and been incredibly helpful in providing technical and academic guidance on a near-daily basis.

Ander Bjorholm Dahl, Professor, DTU Compute
Have co-supervised this thesis and provided technical guidance relating to machine learning.

Morten Rieger Hannemose, Assistant Professor, DTU Compute
Have been helpful with insights and technical discussions relating to machine learning.

Contents

List of abbreviations

2D	Two-dimensional
ASF	Alaska Setellite Facility
CL	Chunk Loader
CNN	Convolutional Neural Network
CO_2eq	Carbon Dioxide Equivalent
CPU	Central Processing Unit
DD	Double-Difference
DEM	Digital Elevation Model
DDInSAR	Double-Difference Interferometric Synthetic Aperture Radar
DINO(v2)	self-DIstillation with NO labels (version 2)
DTU	Technical University of Denmark
EM	Electromagnetic
EMA	Exponentially Moving Average
EO	Earth Observation
ESA	European Space Agency
EW	Extra Wide swath mode
FCN	Fully Convolutional Network
FM	Foundational Model
FN	False Negative
FP	False Positive
GIM	Geophysical Inversion Module
GL	Grounding Line
GPU	Graphics Processing Unit
GT	Ground Truth
HPC	High Performance Computing
IFF	Interferogram Formation Module
InSAR	Interferometric Synthetic Aperture Radar
IGN	Department of Geosciences and Natural Resource Management
IPP	Interferometric Post-Processing chain
IW(1/2/3)	Interferometric Wide swath mode (subswath 1/2/3)
KU	Copenhagen University
LoS	Line of Sight
LOST	Localizing Objects with Self-Supervised Transformers and no Labels

ML	Machine Learning
MLP	Multilayer Perceptron
MSA	Multi-head Self Attention
NaN	Not a Number
NLP	Natural Language Processing
PA	Polar Amplification
PC	Principle Component
PCA	Principle Component Analysis
REMA	Reference Elevation Model of Antarctica
RES	Radar Echo Sounding
RGB	Red Green Blue
S1(A/B/C/D)	Sentinel-1(A/B/C/D)
SA	Self Attention
SAR	Synthetic Aperture Radar
SDPA	Scaled Dot-Product Attention
SGD	Stochastic Gradient Descent
SGL	Subglacial Lake
SHA	Subglacial Hydrological Activity
SLC	Single Look Complex
SLR	Sea Level Rise
TN	True Negative
TOPS	Terrain Observation by Progressive Scans
TP	True Positive
ViT(-S/B/L/G)	Vision Transformer (Small/Base/Large/Giant)

List of figures

List of tables

Chapter 1

Introduction

The significant of Antarctica's subglacial hydrology for glaciological modelling is briefly introduced in this chapter, together with motivations for using machine learning on satellite data to monitor the subglacial hydrological system.

Global mean sea level has risen steadily over recent decades and the trend is expected to continue as the climate warms in the future. One of the primary drivers of sea level rise (SLR) is melting of polar ice sheets. The Arctic is warming two to four times faster than the global average due to a process known as polar amplification (PA), which is driven by positive feedback mechanisms [1]. Although PA in Antarctica remains more uncertain due to limited observations, recent evidence suggests a twofold warming rate [2]. Moreover, studies indicate that PA may be underestimated in current models [1], [2], implying a greater vulnerability of polar ice caps to anthropogenic warming than previously assumed.

In recent years, glaciers in West Antarctica have contributed greatly to global SLR, and some glaciers in the previously stable East Antarctica, such as Totten Glacier, are undergoing change. These glaciers are projected to become major contributors to SLR over the coming century [3]. However, the exact rate of SLR depend on the stability of ice shelves and the flow dynamics of ice sheets—both of which are influenced by subglacial hydrology [4], [5]. Limited understanding of this hydrological system remains a source of uncertainty in SLR predictions [5], [6].

Thick ice covers make glacier beds inaccessible, limiting direct observations of basal conditions and subglacial hydrology. One possibility of observing effects of subglacial hydrology is subtle variations in surface elevation, caused by pressure changes at the base. These episodic surface signals can be detected with synthetic aperture radar (SAR) interferometry [7]–[10]. The European Space Agency's Sentinel-1 (S1) SAR satellite covers most of Antarctica, making it a prominent source of interferometric SAR (InSAR) observations. Unfortunately, manually identifying transient surface displacements is time-consuming and impractical at scale.

This study utilizes large pretrained neural networks—known as foundational models—to detect transient vertical displacements of the ice surface. The goal is to automate the processing pipeline so hydrological changes under the ice sheets can be mapped, and used to understand how subglacial hydrology influences ice mass loss in key areas such as Totten Glacier. This allows current assumptions about basal conditions to be investigated, and may ultimately improve our ability to predict the impact of climate change both in Antarctica and globally.

Chapter 2

Background and theory

This chapter explores the existing literature and theory related to the present work. The first section goes into subglacial hydrology, particularly how it modulates ice flow and interacts with ice sheets. After that comes a section presenting synthetic aperture radar and related techniques in technical details. The last section is about methods in machine learning for classification and segmentation, with the main focus being vision transformers.

2.1 Antarctica's subglacial hydrology

The Antarctic ice sheet is underlain by a network of meltwater channels and water reservoirs at the ice-bed interface. Subglacial water originates from geothermal and frictional heating, while surface meltwater—common in Greenland but rare in Antarctica—can reach the bed via crevasses or moulins. The subglacial water flow is governed by hydrological potential, determined by surface and bed topography (static), and variations in water pressure (dynamic). Altogether, the production, storage and drainage of subglacial water form a complex system that constitutes the subglacial hydrology [9].

Subglacial water reservoirs—known as subglacial lakes (SGL)— may be defined as *"any discrete water body at the base of an ice mass, without presuming a minimum area or depth"* [9]. They range in size from > 100 km across and hundreds of meters deep to just meters wide and centimeters deep. Lakes can be categorized as stable or active. Stable lakes are at equilibrium, with balanced inflow and outflow of water, and are often located close to the ice divides in Antarctica's warm-based interior [9]. Active lakes change in volume as they go through fill-drain cycles. Drainage occurs when the hydropotential fails to retain the stored water, or when leakage triggers syphoning. Drainage duration varies widely—from hours to years. Hydrologically connected lakes undergo cascading drainage, as water from upstream lakes triggers drainage in downstream lakes [8], [11]. Active lakes tend to form closer to the ice margins than stable lakes [9]. An overview of the subglacial hydrological system is shown in Figure 2.1.

Ice dynamics. Subglacial hydrology plays a key role in modulating ice dynamics. Lake drainage can affect basal water pressure in a large region and reduce basal traction, thus increasing the ice flow. This happens when water discharge exceeds the capacity of the drainage system [9]. As basal water pressure rises, the effective pressure (ice overburden pressure minus basal water pressure) decreases and may even become negative. This can trigger basal sliding, or even flotation [7].

Lakes beneath marine-terminating glaciers influence ice dynamics in another way. When basal meltwater reaches the grounding line[1], it forms buoyant freshwater plumes. These plumes rise to the ice-ocean interface, drawing warmer saline water into contact with the ice. This process stirs the stratified water column and disrupts the cold freshwater layer that insulates the floating ice. Known as *plume-driven frontal ablation*, this increases ice margin erosion and weakens shelf stability, [9]. Because ice shelves buttress inland glaciers,

[1] grounding line: the transition from grounded to floating ice. Detailed in next paragraph.

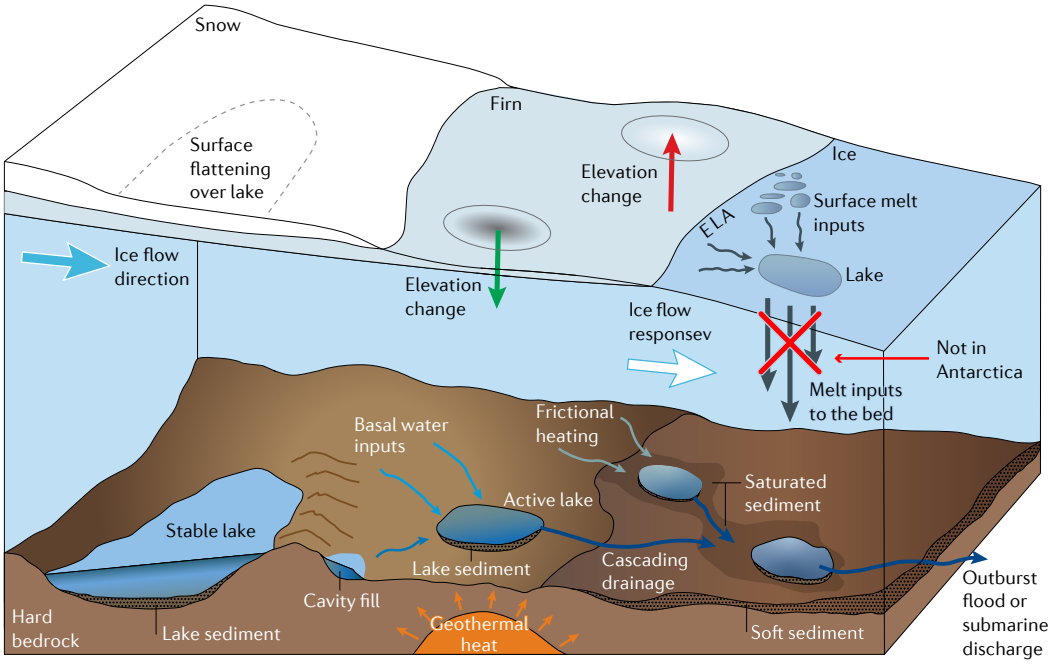


Figure 2.1: **The subglacial hydrological system.** Lakes form as meltwater collects in hydropotential wells. Melting occurs due to geothermal and frictional heating at the base. Active lakes fill and drain over time, sometimes as part of a cascading system of connected lakes. Illustration is from [9] and slightly altered.

their weakening or collapse can destabilize ice sheets. This accelerates ice flow towards the terminus and contributes to global SLR.

A cross-sectional view of the Antarctic ice cap is illustrated in Figure 2.2. It shows the distribution of lakes beneath the ice, and the particularly important influence of lakes near the margins, where drainage may lubricate the base and cause turbulence at the ice-ocean interface.

Grounding lines. Marine-terminating glaciers are defined by the presence of grounding lines. They mark the transition from grounded ice sheets to floating ice shelves, as illustrated in Figure 2.3a. Grounding lines[2] are dynamic and may retreat inland when ice shelves thin. As ice crosses the grounding line, reduced basal friction and buoyancy cause it to flex and accelerate.

Hydrological modelling. Subglacial hydrological models aim to describe the flow and storage of water beneath the ice sheets. Their accuracy is fundamental for projecting glacier dynamics and thus climate models. As shown in Figure 2.3b, the drainage system is commonly categorized into four types

[2]In reality, it is more of a grounding zone, as the exact point of grounding may vary on short time scales depending on tides etc.

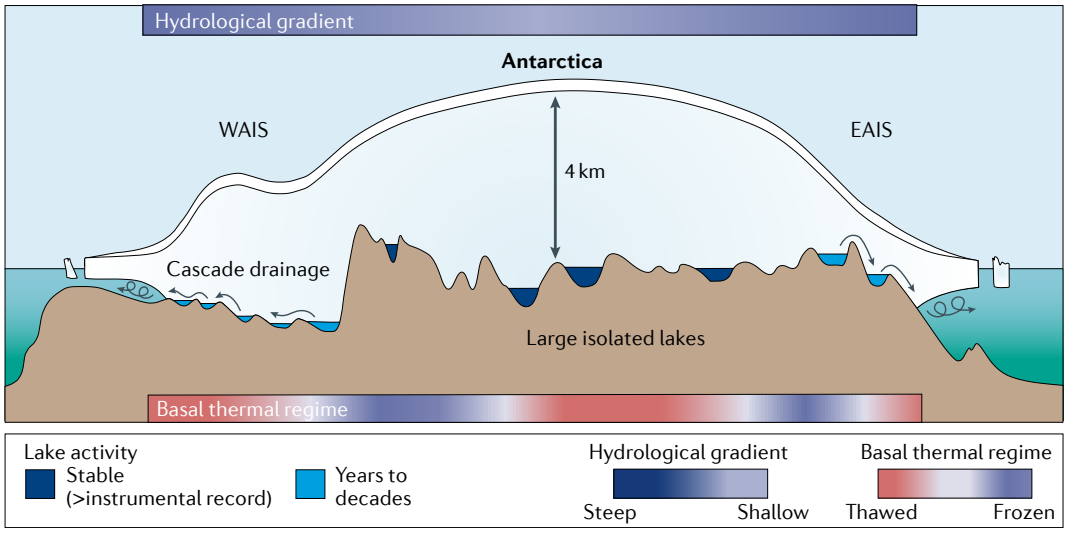


Figure 2.2: **Transect of Antarctica's ice cap.** Large stable lakes are found at the warm base underneath the ice divide, while smaller, active lakes a located towards the ice margins. Subglacial meltwater propagates along the hydrological gradient towards the margins, sometimes triggering cascade drainage. At the coast, subglacial water is discharged below the floating ice shelves and can cause plume-driven frontal ablation. Illustration is modified from [9].

based on bed hardness and drainage efficiency. Models generally assume a steady-state system with balanced water input, output and storage [12]. This assumption simplifies coupling with long-term ice flow models, where basal sliding depends on effective pressure. However, it overlooks the dynamic nature of subglacial hydrology. Episodic drainage can cause transient changes in effective pressure, while cavity formation and collapse may shift the system between states of inefficient (distributed) and efficient (channelized) drainage.

2.1.1 Remote sensing of subglacial hydrology

Seismic surveys and radar echo sounding (RES) measure the extents of both active and stable subglacial lakes [4], [9]. These methods directly measure the subglacial environment but require ground-based or airborne campaigns. For active lakes with episodic filling and draining, these methods become impractical at Antarctica's vast scale. This necessitates satellite measurements; but no current satellite can directly resolving the bed.[3] However, changes in subglacial water transport may alter basal water pressure, causing the overlying ice rise and fall. At the surface, this results in transient uplift and subsidence [7]–[9] (see Figure 2.3c). Surface deformations reveals active subglacial lake locations,

[3]ESA's BIOMASS satellite is the closest contender, but even this P-band radar will likely not be able to resolve basal conditions in Antarctica.

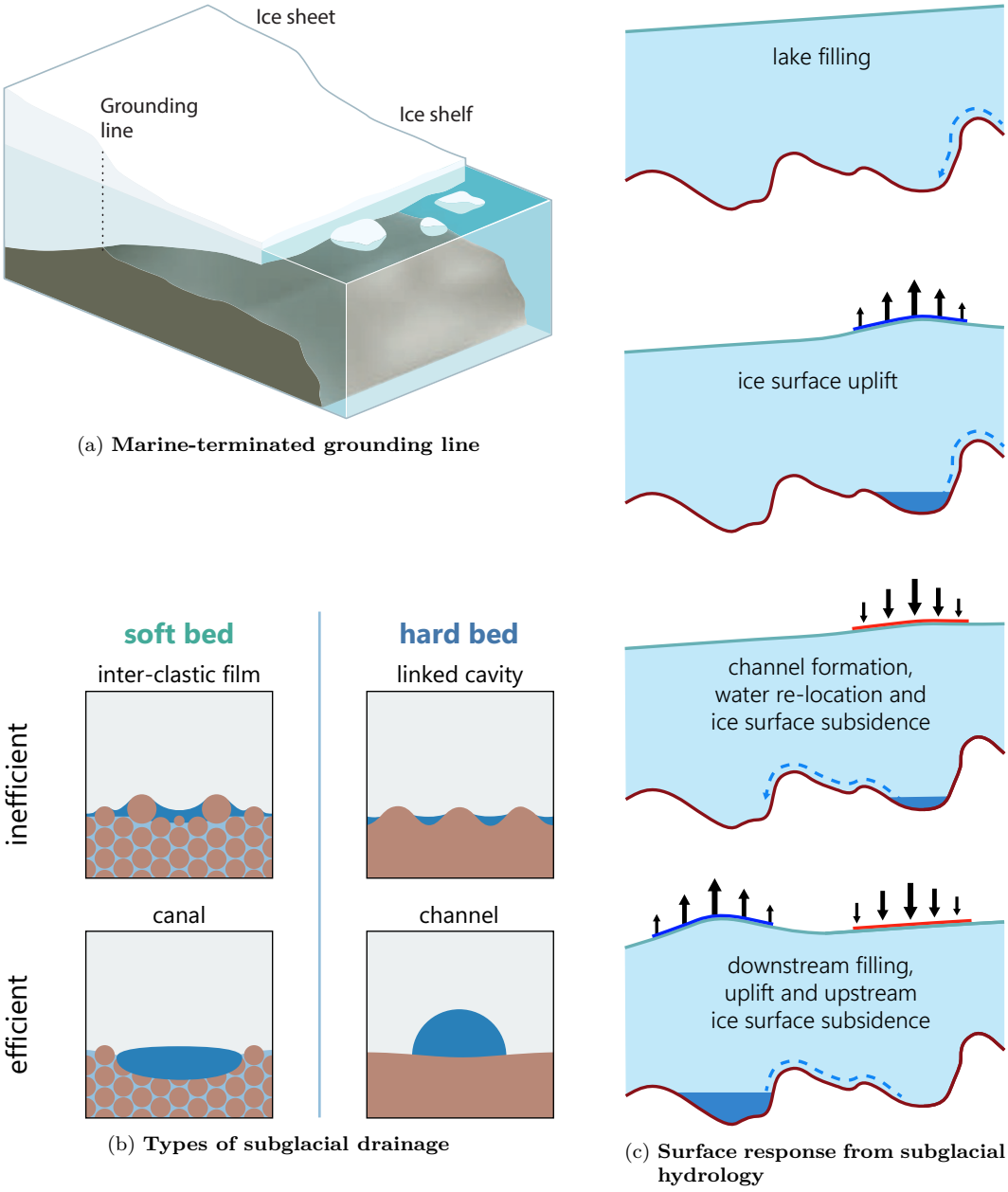
the spatial extent of change, and the magnitude of uplift or subsidence. This enables mapping of subglacial water pathways and volume estimates [8], [10].

Surface deformation can be observed using altimetry and differential synthetic aperture radar interferometry.[4] Consequently, operational satellites like ICESat-2, CryoSat-2, ALOS, and Sentinel-1 can be used [7]–[10], [13], [14], while ERS-1/2 and Radarsat have been used in the past [7], [15]. ICESat-2's laser altimeter offers high vertical precision but sparse spatial coverage, making it arguably better suited for validation purposes [8]. For regional to continental monitoring, Sentinel-1 provides decameter-scale spatial scale, centimeter-level vertical sensitivity, and six-day revisit times under nominal operation.[5]

Since these transient vertical displacements are indirect tracers of subglacial hydrological activity (SHA), no direct observations of subglacial lakes (SGL), this study refers to them as SHA events or simply SHA. Other studies use SGL similarly, but we prefer this clearer distinction.

[4] An explanation of Synthetic Aperture Radar in given in Section 2.2.

[5] Sentinel-1 data is described in more detail in Section 2.2.1.



(a) **Marine-terminated grounding line**

(b) **Types of subglacial drainage**

(c) **Surface response from subglacial hydrology**

Figure 2.3: (a) Simplified illustration of grounding line marking the point where ice become afloat and transition from ice sheet to ice shelf. Illustration is from [16]. (b) Four types of subglacial drainage, split between efficient/inefficient and soft/hard bed. Illustration is recreated from [12]. (c) Propagation of meltwater in the subglacial hydrological system, and the surface uplift and subsidence observed in response to the changing subglacial conditions. Figure is recreated from [8].

2.2 Synthetic aperture radar (SAR)

SAR is a remote sensing technique using radars mounted on moving platforms, typically airplanes or satellites. It achieves high spatial resolution over large areas, and can detect small displacements by combining multiple acquisitions. SAR is an active sensor, providing its own illumination independent of the Sun, and operates in the microwave spectrum, which penetrates clouds. These properties make SAR well suited for monitoring Antarctica's vast, remote regions.

The following sections briefly overview SAR data acquisition and processing; an in-depth description is available in [17].

2.2.1 Sentinel-1 acquisition geometry

The European Space Agency (ESA) operates two[6] Sentinel-1 (S1) satellites under the Copernicus Programme, distributing data free for public and private use [18]. The satellites are in near-polar orbit with a 12-day revisit time. With two satellites separated by 180°, any location on Earth is repeatedly measured along precise ground tracks every 6 days. Over Antarctica, S1's radar operates in so-called Interferometric Wide Swath (IW) mode [19].

Figure 2.4 illustrates S1 data acquisition. As the satellite flies along in the azimuth direction, the radar transmits microwave pulses in the range direction. Each pulse travels to the surface, scatters, and returns as faint echoes back at the radar. The surface terrain lies in the azimuth–ground range plane. A swath is the width of the terrain illuminated by the radar in the ground range direction. Sampling the return signal in discrete time intervals partitions the swath into smaller bins. Repeating this along the azimuth direction allows the instrument to resolve the scene into a sample grid [19].[7] Further details about IW acquisition and parameters are including in Appendix A.1 and Appendix A.2.

Figure 2.5 outlines S1 tracks over Antarctica. It shows the central part of Antarctica being uncovered. That is because the satellites orbit at inclinations of approximately 98°, never reaching latitudes beyond −82°. The rest of the

[6]The number of satellites vary. S1A launched April 2014, S1B launched April 2016, S1B failed December 2021, S1C launched December 2024. S1D is scheduled for the end of this year. This leaves a gap from 2021–2024 where only S1A flew. Nominal operation with two satellites in orbit has occurred from 2016–2021 and will resume in 2025.

[7]The given explanation is simplified in two parts. First, obtaining gridded samples requires careful signal processing to focus the signals in range and azimuth [20]. Second, in IW mode each swath is sampled in bursts. When bursting, the radar sweeps the beam along the swath in the azimuth direction [19]. This is not shown in the illustration.

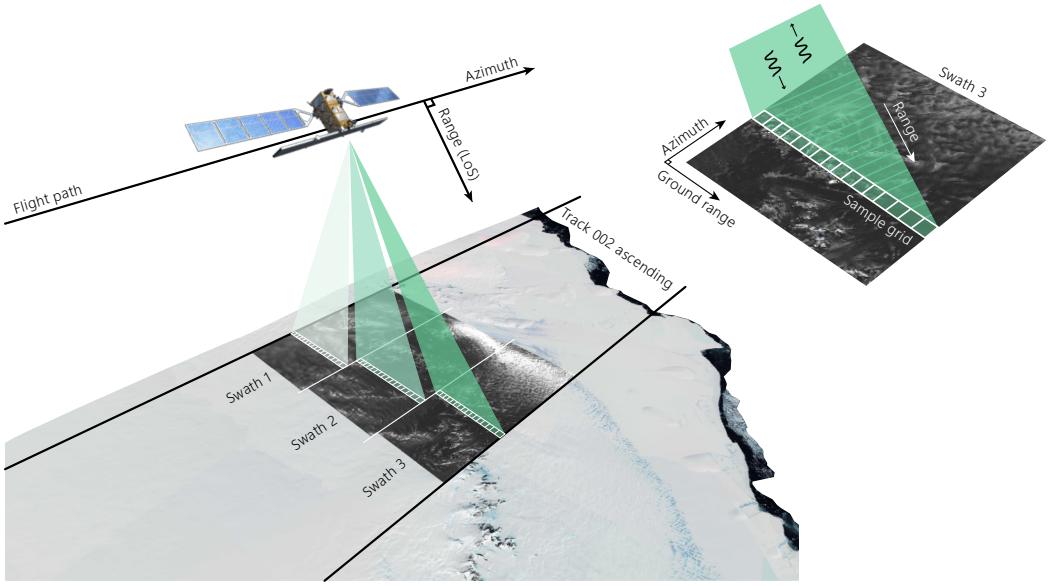


Figure 2.4: **Simplified acquisition geometry for S1 satellites**. The azimuth and range directions span the data space, while the azimuth and ground range directions span the image space. S1 measures tracks consisting of three sub-swaths. Each swath is sampled in range and azimuth to form gridded samples of the surface terrain.

landmass is covered by one or more tracks. S1 data are divided by track and categorized by direction: 'descending' when the satellite moves south and 'ascending' when it moves north again [19].

2.2.2 SAR imaging

Details on SAR signals and image formation are found in [17], [20]. Here is a brief summary of components relevant to this work.

SAR systems observe a scene (e.g., a glacier) by transmitting electromagnetic (EM) waves. These waves travel to and scatter at a position (y, x) on the Earth's surface. This interacting alters both the phase, $\phi(y, x)$, and amplitude, $a(y, x)$, of the EM wave. It also redirects part of the wave back towards the radar (backscatter). This is received as a complex signal

$$s = a\mathrm{e}^{j\phi} \tag{2.1}$$

where j is the imaginary unit. The radar resolves these signals to a certain resolution; a *resolution cell* denotes the smallest ground area the radar can distinguish. By transmitting EM waves and sampling their return as it moves,

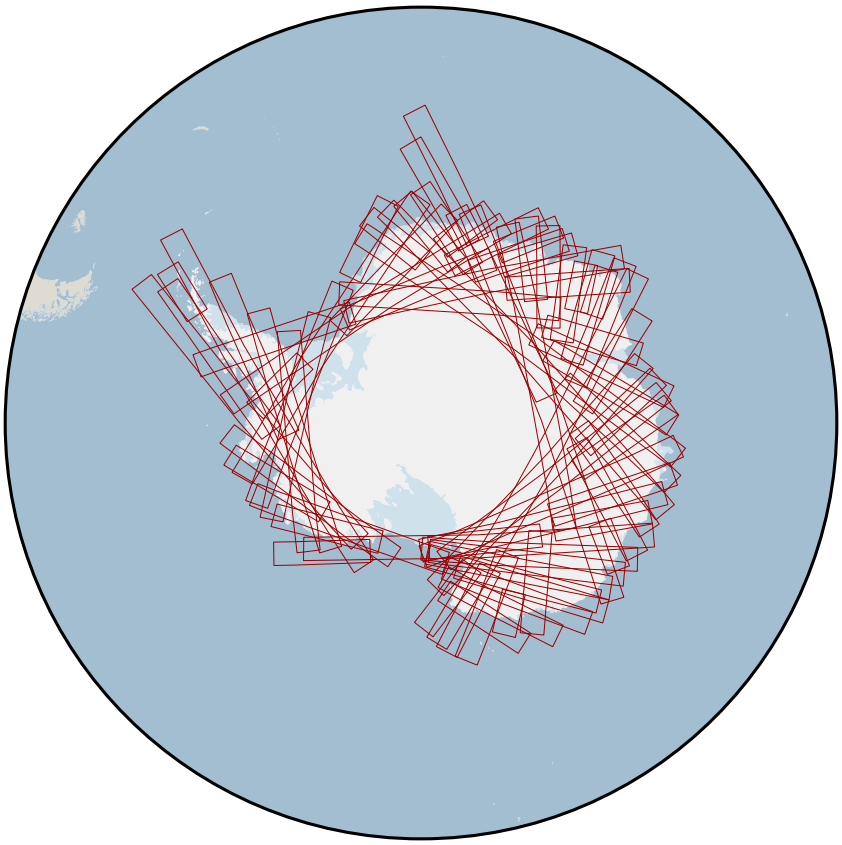


Figure 2.5: **Sentinel-1 acquisitions over Antarctica.** The figure shows tracks of S1 acquisitions obtained over Antarctica from 2015-2024. Illustration made using Qantarctica [21].

the SAR forms a complex-valued image[8], S, over all locations (y, x) in the scene. Formally,

$$S(y, x) : \Omega \subset \mathbb{R}^2 \to \mathbb{C} \tag{2.2}$$

Each coordinate (y, x) in the image domain Ω maps to a complex value $S(y, x)$, representing the received signal (2.1) from that range-azimuth sample. These images are often called single-look complex (SLC) images, as discussed in Section 2.2.3.1. SAR systems (e.g. S1) often oversample, so the pixel spacing (projected on the ground in Figure 2.4) is smaller than the resolution of the radar. This reduces aliasing but causes pixels to be correlated.

Equations (2.1) and (2.2) show that a SAR image captures the scattering properties of Earth's surface by measuring EM wave interactions. Importantly, each resolution cell is a distributed target; the EM wave interacts not with a single

[8]The term "image" is used in a broad sense of data arranged in a 2D grid, with each pixel having some value. However, SAR images are not directly displayable, owing to their complex-valued nature. Visualisation requires some manipulation of the complex data.

point but with an unknown number of discrete scatterers. Each scatterer contributes a backscattered wave with distinct phase and amplitude. The total signal from a resolution cell is the superposition of all such waves. Mathematically, this is a coherent sum over all scatterers:

$$s = a\mathrm{e}^{j\phi} = \sum_{k=1}^{N} a_k \mathrm{e}^{j\phi_k} \tag{2.3}$$

Here, N is the number of scatterers within the cell. Assuming many uncorrelated scatterers, the signal s can be modelled as a zero-mean circular complex Gaussian variable. Thus, the phase ϕ is uniformly distributed over $[0, 2\pi]$, while amplitude a reflects surface properties, varying due to how signals combine. This variation in phase and amplitude is called *speckle*, appearing as noise-like contributions in the SAR image $S(y, x)$. Note that speckle is not true noise but appears noise-like because individual scatterer contributions ($a_k \mathrm{e}^{j\phi_k}$) are unknown. This implies that imaging the same scene twice under identical conditions and viewing geometry yields an unchanged speckle pattern. Conversely, differences between two images over the same region indicate changes occurring between acquisitions. This property underlies SAR interferometry, described next.

2.2.3 SAR interferometry (InSAR)

Suppose two SAR images S_1 and S_2 capture the same scene at different times. For a given pixel location (y, x) each image contains a complex value, s_1 and s_2, representing the returned EM waves from the corresponding resolution cell (2.1). The complex product between these signals is the *interferometric signal*:

$$s_{int} = s_1 \cdot s_2^* = |s_1 \cdot s_2^*| \mathrm{e}^{j\phi_{int}} \tag{2.4}$$

where * is the complex conjugate, s_{int} is the complex interferometric value at (y, x), and ϕ_{int} is the interferometric phase. For the entire image domain, the *complex interferogram* I is defined as

$$I = S_1 \cdot S_2^* \tag{2.5}$$

Typically, only the phase of I, i.e., $\arg(I) \equiv \phi_{int}(y, x)$, is used to form an interferometric image. This phase-only image is often also called the *interferogram*, though it is no longer complex-valued. The time interval between the acquisitions S_1 and S_2 is referred to as the *temporal baseline* of the interferogram.

The interferometric phase ϕ_{int} encodes a mixture of effects causing differences in the phases of s_1 and s_2. These can be modelled as

$$\phi_{int} = (\phi_{flat} + \phi_{topo} + \phi_{defo} + \phi_{atmo} + \phi_{decorr})_{\mathrm{mod}\, 2\pi} \tag{2.6}$$

Notice that the combined phase is *wrapped* modulo 2π, so $\phi_{int} \in [0, 2\pi]$.

- Both ϕ_{flat} and ϕ_{topo} are known phases due to Earth's curvature and surface topography.
- ϕ_{atmo} accounts for temporal atmospheric changes that affect signal propagation, typically manifesting as a smooth, slowly varying phase screen across the interferogram.
- ϕ_{decorr} is an unknown noise term caused by *decorrelation*. Since the arrangement of scatterers generally changes between acquisitins, this term represents residual speckle noise. It can be mitigated by phase filtering.[9]
- ϕ_{defo} is the phase contribution from *surface deformation*. If this is isolated from other phase contributions, it enables us to quantify any surface deformation between two SAR acquisitions.

The deformation phase ϕ_{defo} occurs if surface deformation changes the range R_0 from the radar to the ground, as illustrated in Figure 2.6. The radar does not see the full deformation $\overrightarrow{D}$, only the component along the line of sight (LoS) direction

$$\overrightarrow{LoS} = \frac{\overrightarrow{R_0^1}}{|\overrightarrow{R_0^1}|} \tag{2.7}$$

The deformation phase is

$$\phi_{defo} = \frac{4\pi}{\lambda}\left(R_0^1 - R_0^2\right) = \frac{4\pi}{\lambda}\left(\overrightarrow{D} \cdot \overrightarrow{LoS}\right) \tag{2.8}$$

where λ is the radar wavelength, and R_0^1, R_0^2 are ranges at the two acquisition times. A deformation of $\lambda/2$ along the LoS corresponds to a full 2π phase cycle, called a phase *fringe*.

Assuming purely vertical displacement, the deformation phase relates to vertical displacement d_{vert} as

$$d_{vert} = \frac{\lambda\phi_{dd}}{4\pi\cos\theta_i} \tag{2.9}$$

where θ_i is the local incidence angle and ϕ_{dd} is the *unwrapped* (absolute) interferometric phase.

In the presence of a steady deformation rate v, then

$$\phi_{defo} = \frac{4\pi}{\lambda}T\left(\overrightarrow{v} \cdot \overrightarrow{LoS}\right) \tag{2.10}$$

where T is the temporal baseline.

In real scenarios, interferograms often contain both steady deformation and transient signals, especially over glaciers where localized uplift or subsidence is

[9] A more elaborate description of decorrelation and filtering is given in Section 2.2.3.1.

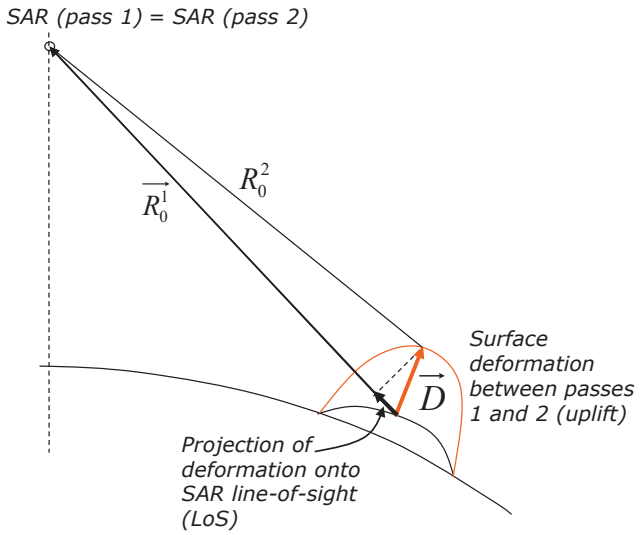


Figure 2.6: **Illustration of surface deformation measured via InSAR.** The deformation is given by the orange vector $\vec{D}$, which is projected onto the line-of-sight $\overrightarrow{LoS}$ to show the deformation along this line. Illustration is from [22].

superimposed on steady ice flow. These steady background deformation fields can be reduced by forming *double-difference (DD) interferogram* via differencing two single-difference interferograms

$$I_{dd} = I_1 \cdot I_2^* = |I_1 \cdot I_2^*| \mathrm{e}^{j(\arg I_1 - \arg I_2)} \tag{2.11}$$

The double-difference phase $\phi_{dd}(y, x)$ is

$$\phi_{dd}(y,x) = \arg I_{dd}(y,x) = \arg I_1(y,x) - \arg I_2(y,x) = \phi_{int,1}(y,x) - \phi_{int,2}(y,x) \tag{2.12}$$

Assuming a subset of pixels (y', x') in the first interferogram senses both steady and transient deformation, while the second interferogram contains steady deformation only:

$$\begin{aligned} \phi_{int,1}(y',x') &= \phi_{defo,1}(y',x') = \phi_{steady} + \phi_{transient} \\ \phi_{int,2}(y',x') &= \phi_{defo,2}(y',x') = \phi_{steady} \end{aligned} \tag{2.13}$$

then, from (2.12),

$$\phi_{dd}(y',x') = \phi_{steady} + \phi_{transient} - \phi_{steady} = \phi_{transient} \tag{2.14}$$

Thus, DD interferograms I_{dd} theoretically isolate transient deformation phase changes. This method has been successfully applied in interferometric glaciology studies [8], [10].

Figure 2.7 shows real examples of two single-difference interferograms I_1 and I_2, and their corresponding DD interferogram I_{dd}. Most of the background deformation field is removed by differencing, although a slowly varying atmospheric

phase screen ϕ_{atmo} remains. The figure also displays a prominent grounding line (bottom right) and smaller subglacial hydrological activity (SHA) patterns. The front-page image of this thesis is another example of a DD interferogram highlighting SHA patterns.

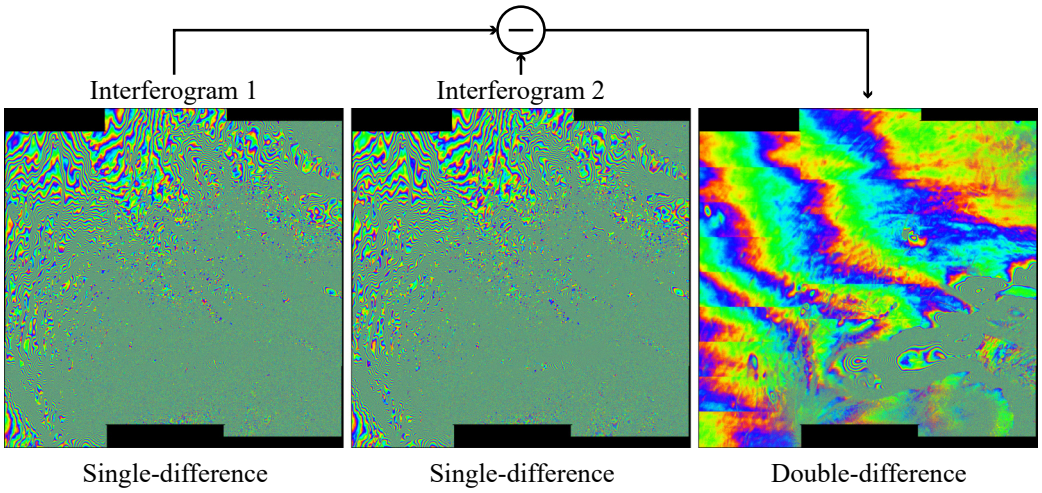


Figure 2.7: **Example of single- and double-difference interferograms.** The colormap is cyclical, with a full cycle corresponding to a 2π phase change. Data is from Sentinel-1, track 099 ascending. Interferogram 1 is formed between acquisitions from 2019-10-28 and 2019-11-03, and interferogram 2 is formed between acquisitions from 2019-11-03 and 2019-11-09, so each has a temporal baseline of 6 days and the combined temporal baseline is 12 days.

2.2.3.1 Decorrelation and coherence

In Figure 2.7, some regions of the DD interferogram appear notably noisy, particularly toward the bottom of the image. These noisy regions also appear in the corresponding single-difference interferograms. This noise is referred to as *phase noise* and originates from the ϕ_{decorr} term in (2.6). As explained in Section 2.2.3, this decorrelation arises when the backscatter signature changes between corresponding pixels in the two SAR images.

To reduce the impact of ϕ_{decorr}, it is common practice to *multi-look* the complex interferograms I. This involves averaging neighbouring pixels in both the range (x) and azimuth (y) directions. For example, with $N = 5$ range-looks and $M = 5$ azimuth-looks, each non-overlapping $N \times M = 5 \times 5$ neighbourhood—or "look"—is averaged to produce a single pixel.[10] The resulting multilooked interferogram I_{ML} has a *multilook factor* of

$$L = N \cdot M = 5 \cdot 5 = 25 \tag{2.15}$$

[10]For readers with a background in convolution neural networks, this is essentially the same as 5×5 average-pooling with stride 5, but with complex values.

As a side note, images are called *single-look* when no averaging is applied—hence, SAR images are often referred to as *single-look complex (SLC) images*.

To quantify the degree of similarity (correlation) between neighbourhoods in the two SAR images, the coherence γ is used. For a neighbourhood $\mathcal{W}_{N\times M}$ centered at pixel (y, x), coherence is estimated as

$$\gamma(y,x) \approx \frac{\sum\limits_{(i,j)\in\mathcal{W}_{N\times M}} s_1(i,j)\cdot s_2^*(i,j)}{\sqrt{\sum\limits_{(i,j)\in\mathcal{W}_{N\times M}} |s_1(i,j)|^2 \cdot \sum\limits_{(i,j)\in\mathcal{W}_{N\times M}} |s_2(i,j)|^2}} \tag{2.16}$$

where the sums are taken over non-overlapping neighborhoods $\mathcal{W}_{N\times M}$. Perfect correlation yield $\gamma = 1$, while complete decorrelation yields $\gamma = 0$.

The coherence γ is influenced by several factors (stated in Figure 2.8). Geometric and volume decorrelation occurs when the radar viewing angle changes between acquisitions, altering the apparent layout of scatterers. Temporal decorrelation is due to changes in the scatterers themselves over time. The last two terms are related to hardware noise and software artifacts.

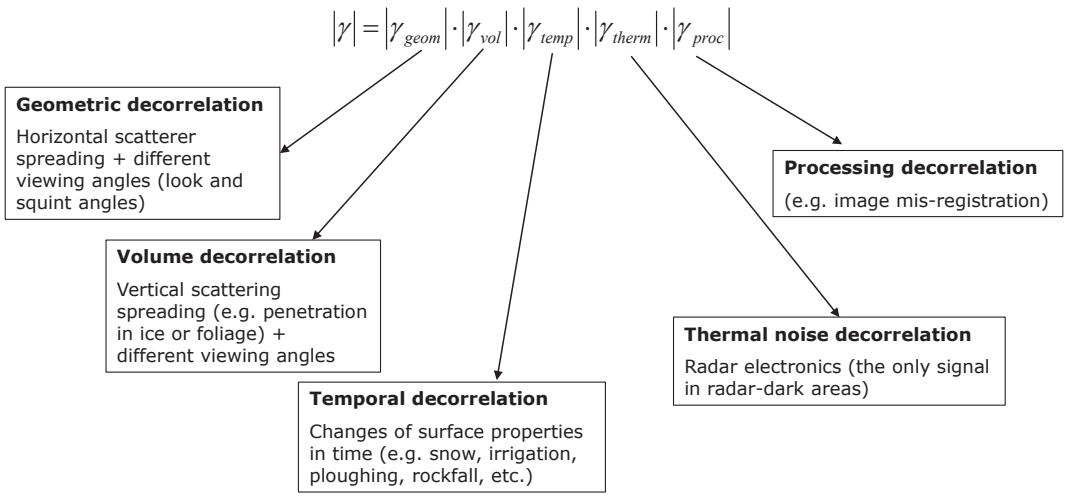


Figure 2.8: **Coherence decomposed into types of (de)correlation.** Illustration is from [22].

2.3 Machine learning (ML)

Double-Difference InSAR (DDInSAR) is an effective technique to produce interferograms in which transient surface deformation signatures are prominent. Nevertheless, such events are sparse in both space and time, interferograms are often noisy, and the Sentinel-1 (S1) data catalogue is massive, making manual

analysis impractical. ML has strong potential for automating the detection of such deformation signals. Unlike physical models, ML algorithms are generally effective at detecting patterns in data where unwanted influences, artifacts and noise sources are difficult to model. This is because ML algorithms learn to generalize from specific examples. In this work, we are particularly interested in methods for classification and segmentation. Both classification and segmentation are treated as binary tasks, where the goal is to detect the presence or absence of transient deformation in DDInSAR products. For segmentation, this corresponds to distinguishing regions undergoing transient deformation (foreground) from the surrounding background.

ML is a rapidly evolving field in which new methods are regularly developed. Since algorithms are tailored to specific tasks and datasets, it is difficult to predict their effectiveness in advance. This challenge also applied to DDInSAR analysis, where few ML models are explicitly designed for this specific application. Convolutional neural networks (CNNs) have previously been applied to similar tasks [23], [24], but this work employs a vision transformer instead. Details on the architecture, training, and evaluation are provided in the following subsections.

2.3.1 Vision transformers (ViT)

The ViT architecture, introduced in [25], is an adaptation of the transformer architecture originally developed for natural language processing (NLP) [26]. Transformers use a self-attention mechanism, which is perhaps best known for its use in large language models such as ChatGPT.

The general architecture of ViTs is shown in Figure 2.9. The network takes an image input $\mathbf{x} \in \mathbb{R}^{C \times H \times W}$, where $W \times H$ is the resolution and C is the number of colour channels. The image is divided into non-overlapping patches of size $P \times P$, which are then flattened into vectors $\mathbf{x}_p \in \mathbb{R}^{N_p \times (P^2 \cdot C)}$, where $N_p = HW/P^2$ is the total number of patches. Each patch is projected into a D-dimensional embedding space using a trainable linear layer. The resulting embedded vectors are referred to as *tokens*. An learnable `[class]` token is prepended to the sequence, before learnable position embeddings are added to the tokens. The sequence of tokens is passed through the transformer encoder composed of L identical blocks, each consisting of a multi-head self-attention (MSA) layer and a multilayer perceptron (MLP), with layer normalization (LN) applied before each block.

The transformer encoder iteratively updates each token's embedding to capture salient image features. These output tokens are often interpreted as feature vectors. The `[class]` token is particularly important, as it does not correspond to any specific patch. Instead, it learns to aggregate global information

from all patches, thus serving as a feature representation of the entire image. This makes the `[class]` token useful for downstream tasks such as image classification. As shown in Figure 2.9, the `[class]` token is passed through a final MLP head to produce class predictions.

A mathematical formulation of the ViT architecture up to the final MLP head is provided in [25]:

$$\mathbf{Z}_0 = \left[\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\, \cdots;\, \mathbf{x}_p^{N_p}\mathbf{E}\right] + \mathbf{E}_{\text{pos}}, \qquad \begin{matrix}\mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D} \\ \mathbf{E}_{\text{pos}} \in \mathbb{R}^{N\times D}\end{matrix} \tag{2.17}$$

$$\mathbf{Z}'_\ell = \text{MSA}\left(\text{LN}\left(\mathbf{Z}_{\ell-1}\right)\right) + \mathbf{Z}_{\ell-1}, \qquad \ell = 1 \ldots L \tag{2.18}$$

$$\mathbf{Z}_\ell = \text{MLP}\left(\text{LN}\left(\mathbf{Z}'_\ell\right)\right) + \mathbf{Z}'_\ell, \qquad \ell = 1 \ldots L \tag{2.19}$$

$$\mathbf{y} = \text{LN}\left(\mathbf{Z}_L^0\right) \tag{2.20}$$

Inputs consist of the class token $\mathbf{x}_{class}$ and patch tokens $\mathbf{x}_p$, where superscripts denote individual tokens. The sequence of tokens is denoted $\mathbf{Z}$ as they pass through layers 0 to L. The number of tokens (the length of the sequence) is $N = 1 + N_p$ due to the added `[class]` token. The final output $\mathbf{y}$ corresponds to the `[class]` token's embeddings after the final encoder, i.e., $\mathbf{Z}_L^{(0)}$). The embedding matrices $\mathbf{E}$ and $\mathbf{E}_{pos}$, along with the functions MSA, MLP and LN, are explained in more detail below.

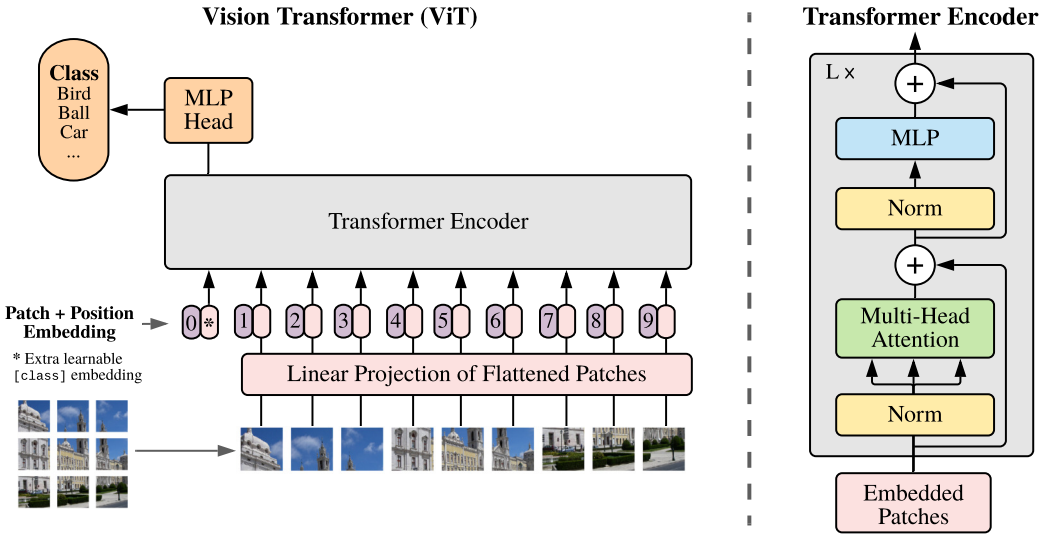


Figure 2.9: **Block diagram of ViT architecture.** Fix-resolution images are divided into patches and projected to an embedding space. The embedded patches pass through a number of transformer encoders from which the final output is obtained. Illustration is from [25].

Embeddings. The embedding matrices in (2.17), $\mathbf{E}$ and $\mathbf{E}_{pos}$, contain learnable parameters, also called *weights*. The matrix $\mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D}$ linearly

projects each flattened patch $\mathbf{x}_p^i \in \mathbb{R}^{P^2 \cdot C}$ into a D-dimensional embedding space. Subsequently, the matrix $\mathbf{E}_{pos} \in \mathbb{R}^{N \times D}$ is added to the sequence to inject position information into each of the N token.

Instead of flattening and linear projection, more recent ViTs use a 2D convolution layer (Conv2D) to split the image, $\mathbf{x} \in \mathbf{R}^{C \times W \times H}$, directly into D-dimensional patches. In this case, the Conv2D layer with kernel size and stride equal to the patch size (P, P) produces non-overlapping patch embeddings of dimension D. The output of this operation is $\text{Conv2D}(\mathbf{x}) \in \mathbb{R}^{D \times \frac{H}{P} \times \frac{W}{P}}$, which is reshaped into a sequence of tokens before prepending the `[class]` token and position embedding $\mathbf{E}_{pos}$.

In both approaches, the embedding layer produces a sequence of N tokens as described in (2.17), where each token is a vector of dimension D in the latent feature space. Figure 2.10 provides visual examples illustrating the similarity between positional embedding $\mathbf{E}_{pos}$ associated with different patches.

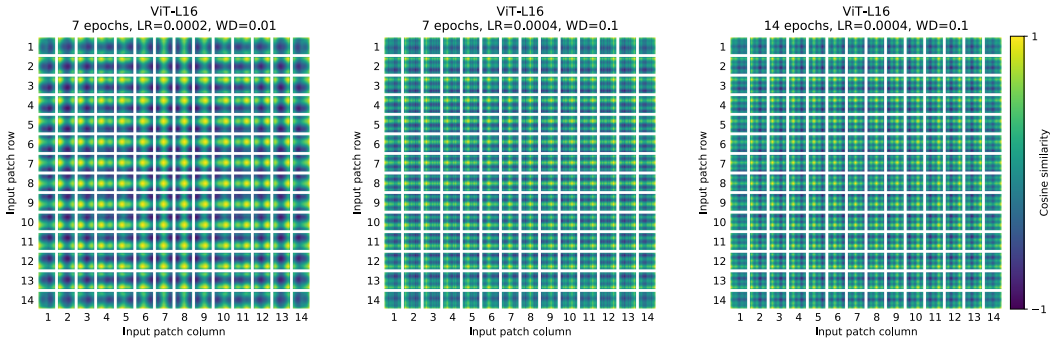


Figure 2.10: **Position embedding similarity depending on training hyperparameters.**[11] Input image has a resolution of (224, 224) px with a patch size of (16, 16) px which correspond to (14, 14) embedded patches. Illustration is from [25].

Multi-head self-attention (MSA). A central component of a transformer is the *attention mechanism*. It can appear quite convoluted as first—especially since multiple attention functions can run in parallel, known as *multi-head attention*. However, since each attention head operates independently, it's helpful to start by examining a single head.

Within a single attention head, the sequence of tokens $\mathbf{Z}_\ell$ is linearly projected into three distinct spaces—queries, keys, and values—using learnable weight matrices

$$\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{D \times D_h} \tag{2.21}$$

[11] More about hyperparameters in Section 2.3.2.

This yields three new sequences of tokens:

$$\begin{aligned}\mathbf{Q} &= \left[\mathbf{q}_0; \mathbf{q}_1; \cdots; \mathbf{q}_{N_p}\right] = \mathbf{Z}\mathbf{W}^Q \\ \mathbf{K} &= \left[\mathbf{k}_0; \mathbf{k}_1; \cdots; \mathbf{k}_{N_p}\right] = \mathbf{Z}\mathbf{W}^K \\ \mathbf{V} &= \left[\mathbf{v}_0; \mathbf{v}_1; \cdots; \mathbf{v}_{N_p}\right] = \mathbf{Z}\mathbf{W}^V\end{aligned} \tag{2.22}$$

where the N tokens in each sequence each has a dimension of D_h. For efficiency, this is often computed in a single operation:

$$[\mathbf{Q}, \mathbf{K}, \mathbf{V}] = \mathbf{Z}\mathbf{W}^{QKV}, \quad \mathbf{W}^{QKV} \in \mathbb{R}^{D \times 3D_h} \tag{2.23}$$

The attention mechanism uses $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ to aggregate contextual information across the sequence. The attention head uses *scaled dot-product attention (SDPA)* to compare each query vector with all key vectors to compute *attention scores*, which reflect the importance of each token with respect to other. The attention scores are scaled by $\sqrt{D_h}$ and normalized using the softmax function:

$$\mathbf{A} = \text{SoftMax}\left(\frac{\mathbf{Q}\mathbf{K}^\mathsf{T}}{\sqrt{D_h}}\right) \in \mathbb{R}^{N \times N} \tag{2.24}$$

These scores are used to compute the *self attention (SA)*—a weighted sum over the value vectors:

$$\text{SA}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = \mathbf{A}\mathbf{V} \tag{2.25}$$

This is shown schematically in Figure 2.11.[12]

To better understand attention, consider its origins in natural language processing. In NLP, a sentences is split into words, which are embedded as vectors and passed through attention layers. Each layer computes relationships between words based on meaning—e.g. how adjectives relate to nouns or verbs.[13] A vision transformer (ViT) works analogously: an image (a sentence) is split into patches (words), with each patch being some collection of pixels (letters). The patches (words) are vectorized and passed through attention layers, where relations between patches (words) are learned. This leads to an intuitive interpretation of the $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ matrices:

1. **Query (Q):** What am I looking for?
2. **Key (K):** What information do I contain that others may look for?

[12] Figure 2.11 (right) shows three separate linear projections, but more recent works use the single combined projection.

[13] This is very simplified and strictly speaking not what the attention layer does, as it is not familiar with concepts such as adjectives, verbs and nouns. Additionally, each input token does not represent entire words, only short strings of characters.

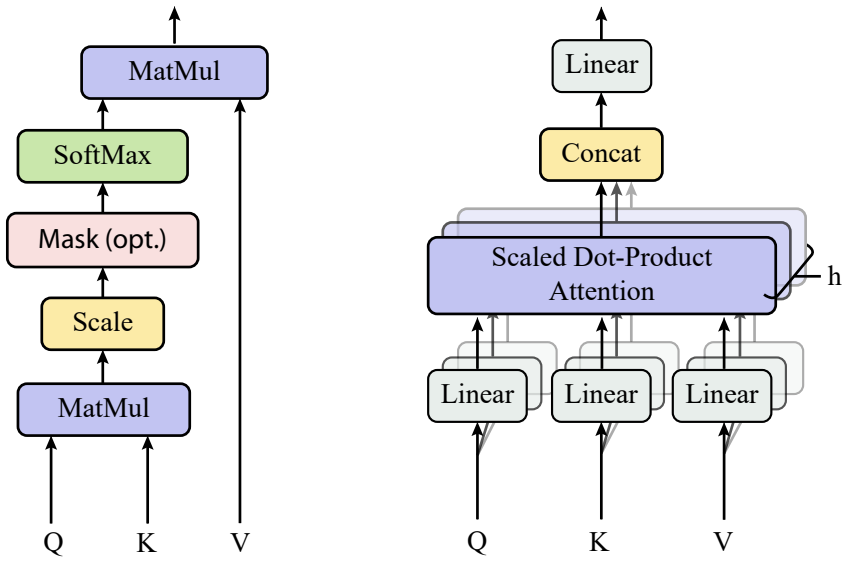


Figure 2.11: (left) **Scaled Dot-Product Attention.** (right) **Multi-Head Attention block.** These consist of several attention layers running in parallel. Illustration is recreated from [26].

3. **Value (V):** What information do I provide to the final representation

— Adapted from [27]

In a sense, we are determining how much **attention** our query should be paying to each key-value pair based on *meaning*. The amount of "attention" is represented as a decimal percentage, called an **attention score**. Mathematically, we can define our output as a simple weighted sum:

$$\sum_i \alpha_i v_i \tag{2.26}$$

where α_i is our attention score for the ith kv pair and v_i is the ith value.

— From [28]

Richer relationships between tokens are established by running h attention heads in parallel, referred to as *multi-head self-attention (MSA)*. The tokens outputted by each head is concatenated along the embedding dimension, and the sequence is then projected to the original D-dimensional embedding space:

$$\mathrm{MSA}\left(\mathbf{Z}\right) = \left[\mathrm{SA}_1\left(\mathbf{Z}\right); \mathrm{SA}\left(\mathbf{Z}\right); \cdots; \mathrm{SA}_h\left(\mathbf{Z}\right)\right] \mathbf{W}^{MSA}, \quad \mathbf{W}^{MSA} \in \mathbb{R}^{k \cdot D_h \times D} \tag{2.27}$$

To keep compute and number of parameters constant when changing h, it is typical to set $D_h = D/k$.

Multi-layer perceptron (MLP). MLPs are found in many ML networks. They consist of layers of interconnected units, or *neurons*, that learn mappings between a set of inputs and outputs. Layers between the input and output layers are called *hidden layers*. Each layer contains a number of neurons, and each neuron is connected to all neurons in the previous layer via learnable weights. Typically, layers also include a *bias unit*, except the last layer. These weights and biases are updated during training. An example MLP is illustrated in Figure 2.12, and we will use the notation from this figure in the following discussion.

An MLP takes as input a vector $\mathbf{x} = [x_1, \ldots, x_D]$ and prepends an additional bias unit $x_0 = 1$. The full input becomes:

$$\mathbf{x} = [x_0, \ldots, x_D] \tag{2.28}$$

The first layer computes M linear combinations of the input values. For each neuron $i = 1, \ldots, M$:

$$a_i = \sum_{d=0}^{D} w_{id}^{(1)} x_d \tag{2.29}$$

Thus, each element in the new vector $\mathbf{a}$ is a weighted sum over all units in the previous layer. The superscript (1) indicates that the parameters are in the first layer of the network. Weights $w_{id}^{(1)}$ are all the weights connecting units in layer 0 and 1, which form a matrix:

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{10}^{(1)} & w_{11}^{(1)} & \cdots & w_{1D}^{(1)} \\ w_{20}^{(1)} & w_{21}^{(1)} & \cdots & w_{2D}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M0}^{(1)} & w_{M1}^{(1)} & \cdots & w_{MD}^{(1)} \end{bmatrix} \tag{2.30}$$

Equation (2.29) can be written as a matrix product

$$\mathbf{a} = \mathbf{W}^{(1)}\mathbf{x} \tag{2.31}$$

To introduce non-linearity, each element a_i is passed through an activation function $\phi(\cdot)$ to give:[14]

$$z_i = \phi(a_i) \tag{2.32}$$

[14] In [29] the activation function is denoted $h(\cdot)$, but to avoid confusion with the notion of "heads" also used in this work, we use ϕ instead. Though ϕ is also used to represent phase in Section 2.2, it should be easier to distinguish the two depending on the context of the matter (ML vs SAR).

Common activation functions also used in this work include:

$$\text{ReLU}\left(a_i\right) = \max\left(0, a_i\right) \tag{2.33}$$

$$\text{Tanh}(a_i) = \tanh\left(a_i\right) \tag{2.34}$$

$$\text{GELU}\left(a_i\right) = a_i \cdot \Phi\left(a_i\right), \qquad \Phi\left(a_i\right) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{a_i}{\sqrt{2}}\right)\right) \tag{2.35}$$

where $\text{erf}\left(\cdot\right)$ is the error function.

Referring to the network in Figure 2.12, a new bias unit is added to the hidden layer output, and the units z_i are combined via another set of weights to produce output activations:

$$a_j = \sum_{i=0}^{M} w_{ji}^{(2)} z_i, \qquad j = 1, \ldots, K \tag{2.36}$$

Depending on the task, the output layer may or may not include an activation function. In an MLP used for classification, it is common to use Sigmoid activation for binary classification, and SoftMax for multi-label classification:

$$\text{Sigmoid}\left(a_j\right) = \sigma\left(a_j\right) = \frac{1}{1 + \exp\left(-a_j\right)} \tag{2.37}$$

$$\text{SoftMax}(a_j) = \frac{\exp\left(a_j\right)}{\sum_{k=1}^{K} \exp\left(a_k\right)} \tag{2.38}$$

For example, an MLP using sigmoid activation would have output units

$$y_j = \sigma\left(a_j\right) \tag{2.39}$$

and the whole network in Figure 2.12 from input to output unit would be

$$y_j = \sigma\left(\sum_{i=0}^{M} w_{ji}^{(2)} \cdot \phi\left(\sum_{d=0}^{D} w_{id}^{(1)} x_d\right)\right) \tag{2.40}$$

This is a *shallow network* with one hidden layer. A *deep MLP* contains multiple hidden layers. If there are no hidden layers, the MLP reduces to a linear model. With a single output and a sigmoid activation, $\sigma\left(\cdot\right)$, the model becomes *logistic regression*:

$$y\left(\mathbf{w}, \mathbf{x}\right) = \sigma\left(\mathbf{w} \cdot \mathbf{x}\right) \tag{2.41}$$

where $\mathbf{x} \in \mathbb{R}^{D+1}$ includes the bias unit, and $\mathbf{w} \in \mathbb{R}^{D+1}$ contains the learnable weights. This is often used for *binary classification*.

Returning to the transformer encoder in Figure 2.9, we observe that each MSA layer (2.18) is followed by an MLP layer (2.19). Conceptually, this design serves a meaningful purpose:

- The MSA layer routes contextual information between tokens.
- The MLP layer then processes each token independently, transforming the attended information into a form more suitable for the task.

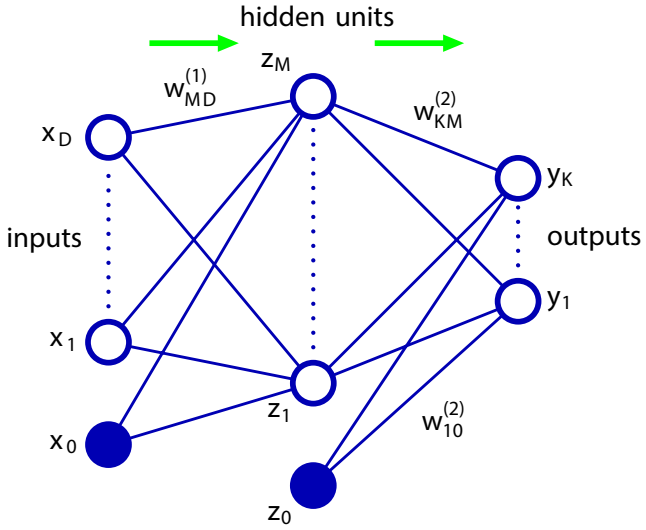


Figure 2.12: **Multi-layer perceptron structure.** Neurons in input-, output- and hidden layers are connected via weights. Illustration is from [29].

Layer normalization (LN). Layer normalization is a technique used to normalize each token according to some learnable parameters

$$\mathrm{LN}\left(\mathbf{z}^{(i)}\right) = \frac{\mathbf{z}^{(i)} - \mathrm{E}\left[\mathbf{z}^{(i)}\right]}{\sqrt{\mathrm{Var}\left[\mathbf{z}^{(i)}\right] + \epsilon}} \cdot \boldsymbol{\gamma} + \boldsymbol{\beta} \tag{2.42}$$

where $\mathbf{z}^{(i)} \in \mathbb{R}^D$ is the embedding of the ith token. $\mathrm{E}[\cdot]$ is the expectation value and $\mathrm{Var}[\cdot]$ is the variance. ϵ is a small constant added for numerical stability, and $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^D$ are learnable parameters for scale and shift. LN is used to help the network train in a more stable and efficient way.

2.3.2 Network training

(This section is based on [29], [30].)

Training a neural network involves adjusting its parameters so that it performs better at a specific task. The performance is measured using a *loss function*, denoted $\mathcal{L}(\mathbf{W})$, which quantifies how far the network's prediction deviates from the ground truth. Classification networks often use the cross-entropy loss function

$$\mathcal{L}_{multi} = -\sum_{k=1}^{K} t_k \log y_k \tag{2.43}$$

where each element in the output vector, $\mathbf{y}$, and the target vector, $\mathbf{t}$, correspond to the probability of the kth class label with $k = 1, \ldots, K$ classes in total. A binary classifier (2.41) with a singular output unit, y, and target value, $t \in \{0, 1\}$, uses the binary cross-entropy loss function

$$\mathcal{L}_{binary} = -t \log y - (1 - t) \log (1 - y) \tag{2.44}$$

Minimizing the loss means finding the weights $\mathbf{W}$ that zeros the gradient

$$\nabla \mathcal{L} (\mathbf{W}) = \frac{\partial \mathcal{L}}{\partial \mathbf{W}} = 0 \tag{2.45}$$

This is done by iteratively taking small step in the direction of $-\nabla \mathcal{L}$ in weight space, as visualized in Figure 2.13 for a very simple 2D weight space. The gradient expresses how much each weight affects the loss, and weights are updates in iterations τ with gradient descent:

$$\mathbf{W}^{(\tau+1)} = \mathbf{W}^{(\tau)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(\tau)}} \tag{2.46}$$

where η is the learning rate.

Evaluating the gradient over the full dataset is computationally expensive, and using a single data sample leads to noisy updates. Therefore, data is randomly divided into smaller batches and the gradient of each batch is used to update the weights. This method is called *stochastic gradient descent (SGD)*, and one full pass over all batches is called an *epoch*.

Note that batch size, number of epochs, and learning rate η in (2.46) are *hyperparameters*. They are used to configure the training process and model configuration. The number of hidden layers, number of units in each layer, input shape, and activation functions are other examples of hyperparameters. They affect performance but are not updated via SGD.

In a multilayer neural network, each weight is updated by computing the gradient of the loss function with respect to that particular weight.

$$w_{ji}^{(\tau+1)} = w_{ji}^{(\tau)} - \eta \frac{\partial \mathcal{L}}{w_{ji}^{(\tau)}} \tag{2.47}$$

Using the chain rule the gradient of the loss function for a given weight in layer ℓ within the network is

$$\frac{\partial \mathcal{L}}{\partial w_{ji}^{(\ell)}} = \frac{\partial \mathcal{L}}{\partial a_j^{(\ell)}} \frac{\partial a_j^{(\ell)}}{\partial w_{ji}^{(\ell)}} = \delta_j^{(\ell)} \cdot z_i^{(\ell-1)} \tag{2.48}$$

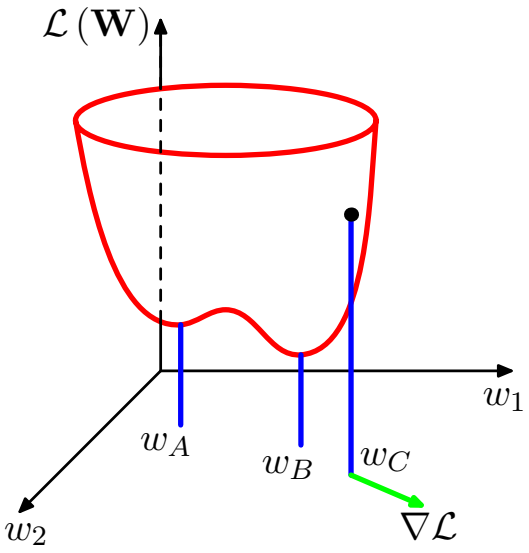


Figure 2.13: **Visualization of gradient descent.** Shown for a loss function $\mathcal{L}(\mathbf{W})$ depending on two parameters w_1 and w_2. At a given point in weight space $\mathbf{W}_C$ the gradient is $\nabla\mathcal{L}$. Taking steps towards $-\nabla\mathcal{L}$ moves loss function towards one of it's local minima, $\mathbf{W}_A$ or $\mathbf{W}_B$. Figure is modified from [29].

where $z_i^{(\ell-1)}$ is the output of unit i in previous layer $(\ell-1)$ and δ_j is referred to as the error signal, given by

$$\delta_j^{(\ell)} = h'\left(a_j^{(\ell)}\right) \cdot \sum_k^K w_{kj}^{(\ell+1)} \delta_k^{(\ell+1)} \tag{2.49}$$

where $h'\left(a_j^{(\ell)}\right)$ is the derivative of the activation function $h(\cdot)$ for pre-activation unit a_j in layer (ℓ), and the sum is over all $k = 1, \ldots, K$ error signals δ_k from units in the next layer $(\ell+1)$ multiplied by the weights w_{kj} that connect the current unit to next layer's units. This is illustrated in Figure 2.14.

Equation (2.49) is recursive, since each $\delta^{(\ell)}$ depends on all $\delta^{(\ell+1)}$ from the next layer and so on. Updating a given weight w_{ji} in layer (ℓ) requires knowledge of the error signals δ from all downstream layers $(\ell+1), (\ell+2), \ldots, (L)$. This is not surprising, as any change to w_{ji} will propagate through the subsequent layers, thereby influencing the loss $\mathcal{L}$. Only at the last layer, L, can the error signals $\delta_j^{(L)}$ be calculated explicitly from $\partial L / \partial a_j^{(L)}$. To update each weight in the network, we therefore start at the last layer and work backwards towards the first layer. This scheme is called *backpropagation*.

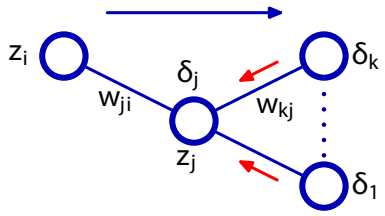


Figure 2.14: **Illustration of how δ_j for unit j is calculated.** The δs from units k are backprojected to j. The blue arrow denotes forward direction of the network (inference direction) and the red arrows indicate the flow of error information during backpropagation. The illustration is from [29].

2.3.3 Self-distillation with no labels (DINO)

As noted in [25], vision transformers (ViTs) exhibit far less inductive bias than convolutional neural networks (CNNs). A CNN considers neighbouring pixels to be related (locality) and is translationally equivalent. In a ViT, only the MLP blocks are local and translationally equivariant, while attention-blocks are global (any token can attend to all other tokens). This makes ViTs more data-driven, being less constrained by assumptions baked into the model. They are generally better at modelling long-range dependencies between patches, given their global receptive field from the very first layer—unlike a CNN that requires multiple layers to relate distant pixels—meaning ViTs are potentially better at capturing patterns that require global context-awareness. However, the reduced inductive bias means that ViTs do not generalize well when trained on small datasets.

Training a ViT from scratch on DDInSAR data to detect surface deformation related to subglacial hydrological changes is not realistic for the present work. Collecting sufficient data and training the network would be time-consuming, and may prove entirely unfeasible. This issue is avoided altogether by using a pretrained foundational models. In particular, a model known as *self-distillation with no labels* (DINO) is used.

DINO is described in [31] and the structure is shown in Figure 2.15. DINO is used for image classification and is trained in a self-supervised way where, during training, different views $\mathbf{x}_1$ and $\mathbf{x}_2$ of an image $\mathbf{x}$ are passed through two separate networks, a student g_{θ_s} and a teacher g_{θ_t}, which use the same architecture, g. The networks, g, are composed of a backbone f (e.g. a ViT) and of a projection head h (MLP), so that $g(\mathbf{x}) = h(f(\mathbf{x}))$ as shown in Figure 2.9 with the MLP head added onto the transformer. Both the teacher and the student network output a probability distribution function (pdf), p_1 and p_2, representing the probability of each label (class). The model learns from the cross-entropy between the teacher and the student. If both networks output similar pdfs, p_1 and p_2, the cross-entropy is small and parameters are

only updated slightly, whereas completely different pdfs result in a large cross-entropy and thus a larger change in network parameters. The key here is that the teacher and the student makes predictions independent of each-other, so to minimize the cross-entropy between their outputs, they must learn to classify each image based on some inherent/latent features contained in each image.

DINO makes this framework work by using the teacher as a stable reference for the student network. Therefore, the cross-entropy is only used to update the weights of the student network, θ_s, via stochastic gradient descent (SGD). The weights in the teacher network, θ_t, are not updated with SGD, but instead with an exponential moving average (EMA) of the student network weights. In this way, the framework is distilling knowledge from the teacher onto the student, with the teacher itself being an aggregate of previous students.

Self-supervised models are prone to collapse, which is when both networks defaults to a trivial solution. DINO can collapse in two ways: if both networks learn to always predict a uniform pdf (all classes are equally likely) or a pdf where one particular class or cluster always dominates. In both cases, the cross-entropy is low, but the model has achieved this by circumventing the intended behaviour. DINO applies centering and sharpening to avoid collapse. The specific details are not relevant to this work, but a very rough example of their effect on the output pdf is shown in Figure 2.15 (right). Sharpening is also applied to the student (though it is less than for the teacher) before the cross entropy between p_1 and p_2 is computed.

2.3.4 Performance evaluation

Common evaluation metrics are summarized in [32]. For binary tasks, both predictions and targets are either positive (1) or negative (0). This yields four possible outcomes:

true positive (TP)	correctly predicted positive outcome
true negative (TN)	correctly predicted negative outcome
false positive (FP)	positive instance predicted to be negative
false negative (FN)	negative instance predicted to be positive

These outcomes form the basis for a set of standard performance metrics.

Binary classification metrics. Accuracy measures the fraction of true predictions to the total number of predictions

$$\text{Acc.} = \frac{TP + TN}{TP + TN + FP + FN} \in [0, 1] \tag{2.50}$$

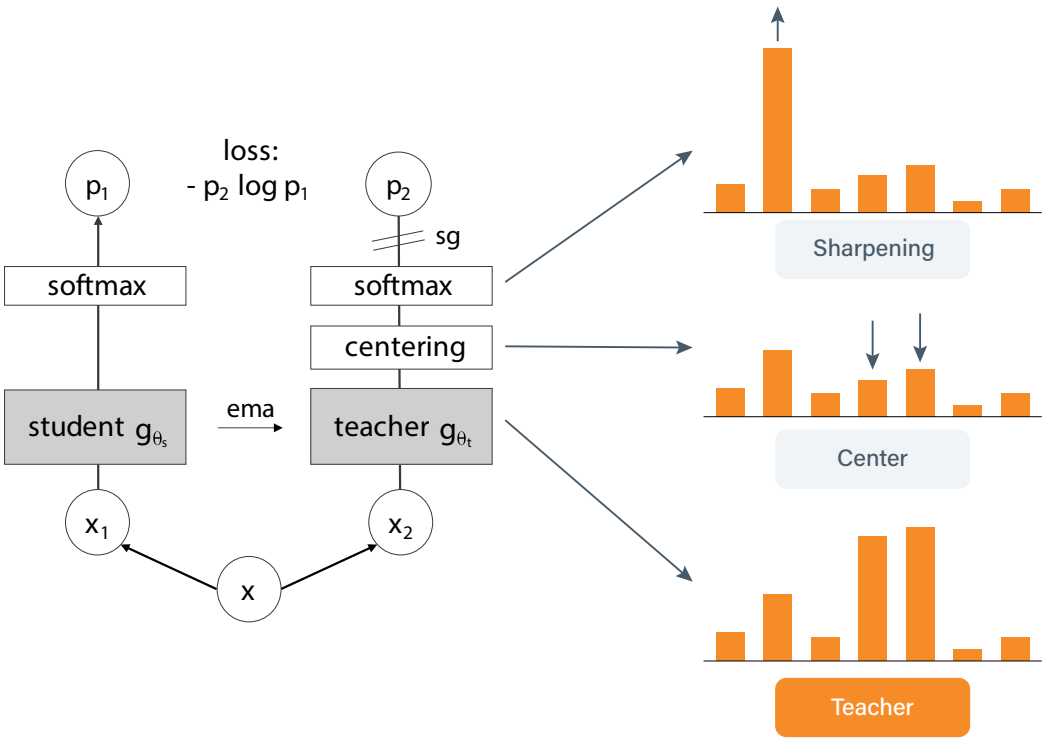


Figure 2.15: (left) **High-level architecture of DINO network.** Originally from [31]. (right) **Effect of centering and sharpening on teacher pdf.** This is an oversimplification to show that centering helps to remove persistently dominating classes (exemplified by the two large components in the pdf), while sharpening amplifies the relative size of the dominant class after centering.

Recall measures how many of all the positive outcomes that were correctly predicted to be positive

$$\text{Rec.} = \frac{TP}{TP + FN} \in [0, 1] \tag{2.51}$$

Specificity measure how many of all the negative outcomes that were correctly predicted to be negative

$$\text{Spec.} = \frac{TN}{TN + FP} \in [0, 1] \tag{2.52}$$

Precision is the fraction of correct positive predictions out of all positive predictions

$$\text{Prec.} = \frac{TP}{TP + FP} \in [0, 1] \tag{2.53}$$

Both recall and precision are especially useful in cases of class imbalance, where one class (typically the negatives) significantly outnumbers the other. These two metrics often trade off against each other. Increasing recall (by making the

model more prone to positive predictions) often lead to a decrease in precision (as negatives are falsely predicted as positives), and vice versa. To strike a balance between recall and precision, the harmonic mean can be used

$$F_1 = \frac{2 \cdot \text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \in [0, 1] \tag{2.54}$$

which is referred to as the F_1-score. It favours recall and precision equally. It is possible to favour one over the other using a similar F_β-score, but this is not used in this work.

Binary segmentation metrics. Binary segmentation can be evaluated analogously to classification, except that predictions are made per pixel instead of per instance. The predicted and ground truth labels are represented as segmentation masks or arrays, with each pixel labelled as positive (1) or negative (0).

Let X be the set of positive pixels in the predicted mask and Y the set of positive pixels in the ground truth mask. Similar to $F1$-score, the Dice coefficient balance the number of false positives and false negatives:

$$\text{Dice} = \frac{2 \cdot |X \cap Y|}{|X| + |Y|} \in [0, 1] \tag{2.55}$$

where $X \cap Y$ is the overlap between set X and Y, and $|\cdot|$ denotes the number of pixels in the set.

Note that $|X \cap Y| = TP$, $|X| = TP + FP$ and $|Y| = TP + FN$, which can be substituted into (2.55) to obtain the same expression as in (2.54). Thus, the Dice coefficient can be interpreted as the harmonic mean for binary segmentation.

Chapter 3

Algorithm implementation and processing chain

This work implements a ML algorithm to automatically detect SHA from DDInSAR data. The developed algorithm is referred to as SGLNet - a name chosen to draw connections between subglacial lakes (SGL) and neural networks.[1] SGLNet is created as a component of a larger project at KU called DynaMelt. This project makes use of software built by DTU for InSAR processing, called the Interferometric Post-Processing Chain (IPP). The implementation of SGLNet therefore has to consider the processing chain of DynaMelt, which has important implications for inputs, outputs and design choices.

This chapter will explain the processing chain including a brief overview of the IPP software. It also specifies model implementation and design choices made during development. Lastly, some notes about performance testing are added.

The algorithm and a user guide can be found at
https://github.com/NielsPeterChristensen/SGLNet

[1] As mentioned in Section 2.1, the abbreviation SHA is preferable over SGL for the type of events that this work is concerned with. Nevertheless, the implemented algorithm includes SGL in its name, because it is a recognizable abbreviation within the field of subglacial hydrology, and it is not uncommon to use it broadly.

3.1 Processing chain

A high-lever overview of the processing pipeline of Project DynaMelt is shown in Figure 3.1. Boxes in green represent processing introduces in this work, while yellow boxes represent IPP modules. Blue boxes are additional processing steps used by Project DynaMelt. The double differencing module was not originally part of the IPP, but has been incorporated since, hence the striped colouring. From end to end, the processing chain takes Sentinel-1 SLC products and outputs shapefiles with outlines of SHA events. SGLNet is situated in the middle of the chain. It receives phase images, $\phi_{dd}(y, x)$, equivalent to (2.12), from the double differencing module and produces txt-files with SHA event coordinates. These coordinates are used to filter and crop interferograms, so that the geophysical inversion module only receives interferometric data from detected SHA event. An optional segmentation routine based on the attention mechanism has also been implemented. This offers an alternative processing chain towards the final event shapefiles. This functionality was never the main priority, so it is only at an experimental stage. For the same reason, no method has been developed to transform the attention segmentations into geocoded shapefiles in this project.

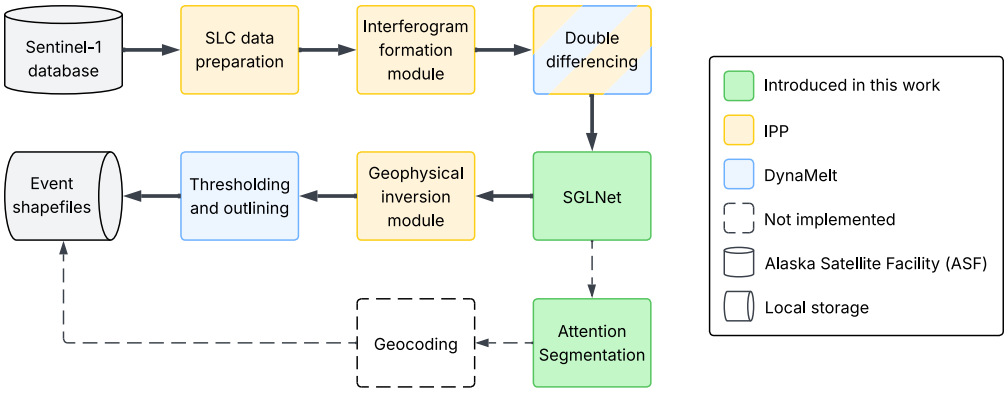


Figure 3.1: **High-level processing chain of Project DynaMelt.** The pipeline takes Sentinel-1 SLC products and process them to produce outlines of SHA events as shapefiles. The pipeline uses the IPP to generate DD interferograms, and to unwrap detected regions of interest. Some processes in the pipeline are DynaMelt-specific (with 'double differencing' being in-between, as it has been incorporated into the IPP). SGLNet detects SHA events which are passes on. It can also optionally be used to do semantic segmentation.

The processing chain of SGLNet specifically, is illustrated in Figure 3.2. It can be divided into two main components. First, the input interferogram $\phi_{dd}(y, x)$ must be transformed such that it can be given as input to DINO. The interferogram is in the form of a 2D matrix with upwards of 6000 elements

in each dimension, while DINO receives 3-channel (RGB) images of resolution (224 px, 224 px) or (448 px, 448 px). Second, the DINO ViT runs inference on the RGB images to extract feature vectors (tokens). The `[class]` token is then passed to a linear classifier to detect SHA events.[2]

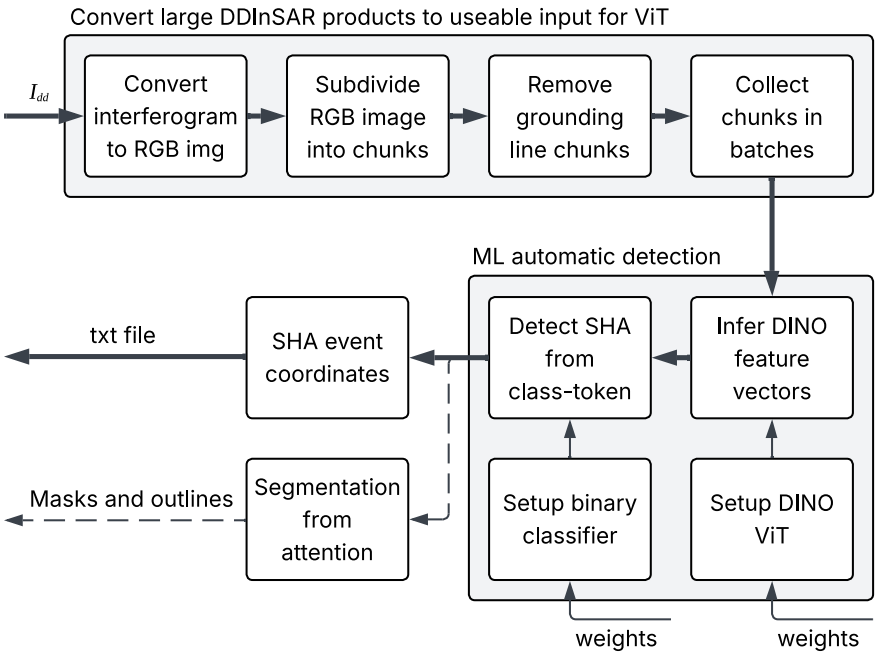


Figure 3.2: **Block-diagram of SGLNet dataflow.** Interferograms first go through a series of operations to go from 2D gridded data to batches of multiple smaller 3-channel RGB images (chunks). DINO runs inference on chunk batches and passes output to another network that uses the `[class]` feature vectors to detect SHA. Coordinates of chunks where SHA are detected are collected into a txt file. Optionally, attention segmentation can be performed on positively detected chunks, which outputs binary masks and outlines.

The following two subsections will give more detail on the IPP and the implementation of SGLNet.

3.1.1 Interferometric post-processing chain (IPP)

The IPP is responsible for all interferometric processing. It is coded in C with Python compatibility and designed to run through Linux command lines. The IPP tools and modules relevant to this work are

- SLC data preparation

[2]See Section 2.3.1 for more about feature vectors and `class`-tokens.

- Interferogram formation module
- Double differencing
- Geophysical inversion module

and are documented in the IPP user guide [33]. The following paragraphs offer a briefly introduction to each module.

SLC data preparation. Sentinel-1 data products are downloaded from the Alaska Satellite Facility (ASF) database [34]. Each SLC product covers approximately 170 km × 250 km and is supplied in a zippped .SAFE directory. These files are unzipped and unpacked to prepare them for interferometric processing. When multiple products from the same track are available, it also combines these products into a single composite product. The preparation process requires precise orbit files, which are provided by the ESA from various sources.

Interferogram formation module (IFF). The IFF produces interferograms by coherently combining pairs of SLC products. This is not trivial, owing to how S1 samples data with overlapping bursts along three parallel subswaths.[3] The IFF can be configured in various ways. It is supplied with a digital elevation model (DEM) [35] and an external ice velocity map [36]. By default it uses 10 range looks and 2 azimuth looks[4], and a temporal baseline within 6-12 days.

Double differencing. The double differencing module takes two interferograms and produce a DD interferogram according to (2.11). By default, it uses interferograms with a temporal baseline of 6 days. It can be configured to compute all DD interferograms with respect to a common reference, or to use a running reference. The running reference option is used for this work. It is also possible to apply phase filtering - this reduces phase noise, but increases computation time, so it is not used for this work.

Geophysical inversion module (GIM). The GIM converts the interferometric phase to displacements via a phase unwrapping. The displacements are then calibrated and geocoded. While it is technically possible to pass an entire DD interferogram through the GIM, it is not practical for real applications. It is both incredibly slow over large areas and unable to resolve noisy regions of the interferogram. This is because the unwrapping algorithm has to reconstruct the absolute phase (continuous) from the wrapped interferometric phase

[3] See Section 2.2.1.
[4] See Section 2.2.3.1.

(discontinuous), which means solving a huge system of modulo-2π constraints across the given interferogram. This means that computation time generally scales super-linearly with interferogram size. This is also one of the main reasons for why SGLNet is placed between the IFF and GIM module—it is simply not realistic to run large quantities of data through the GIM.Ib this way, we also decrease the likelihood of unwrapping errors, which are generally difficult to detect. Consequently, SGLNet has to operate on the wrapped phases, so that only regions of interest (i.e. where SHA events occur) are passed to the GIM.

3.1.2 SGLNet implementation

The block diagram from Figure 3.2 gives an idea of the processing chain, but does not provide an intuitive understanding of how the algorithm was implemented. For this, Figure 3.3 offers a more visual representation of the algorithm. The implementation is divided into three modules:

- **Chunk Loader:** Transforms large phase interferograms into small, digestible RGB images. It also applies masking (removing specific chunks) and collects chunks into batches.
- **DINO ViT:** The main workhorse of the algorithm, which transforms each chunk to a latent feature space, represented in the form of a high-dimension feature vector.
- **Binary Classification:** The feature vector corresponding to the `[class]` token is passed through a binary linear classifier to predict if the chunk is 'Positive' or 'Negative' in terms of SHA event occurrence.

Each module is described in further detail in the following paragraphs.

Chunk Loader (CL). Each interferogram is passed to the CL. Here, the phase interferogram is first transformed to a suitable RGB representation. Four different representations have been implemented, and are shown in Figure 3.4. These four representations are given as follows:

$$\begin{aligned}
&\text{Phase:} && (R,G,B) = (\arg I_{dd},\ \arg I_{dd},\ \arg I_{dd}) \\
&\text{Polar:} && (R,G,B) = (|I_{dd}|,\ \arg I_{dd},\ 0) \\
&\text{Recta:} && (R,G,B) = (\cos(\arg I_{dd}),\ \sin(\arg I_{dd}),\ 0) \\
&\text{Blend:} && (R,G,B) = \text{IPP pseudocolor image}
\end{aligned} \tag{3.1}$$

The 'Phase' representation is simply the interferometric phase $\arg I_{dd}$ in all three channels. The 'Polar' format mimics the polar form of complex numbers, with magnitude $|I_{dd}|$ in the first channel, phase in the second channel, and zeros

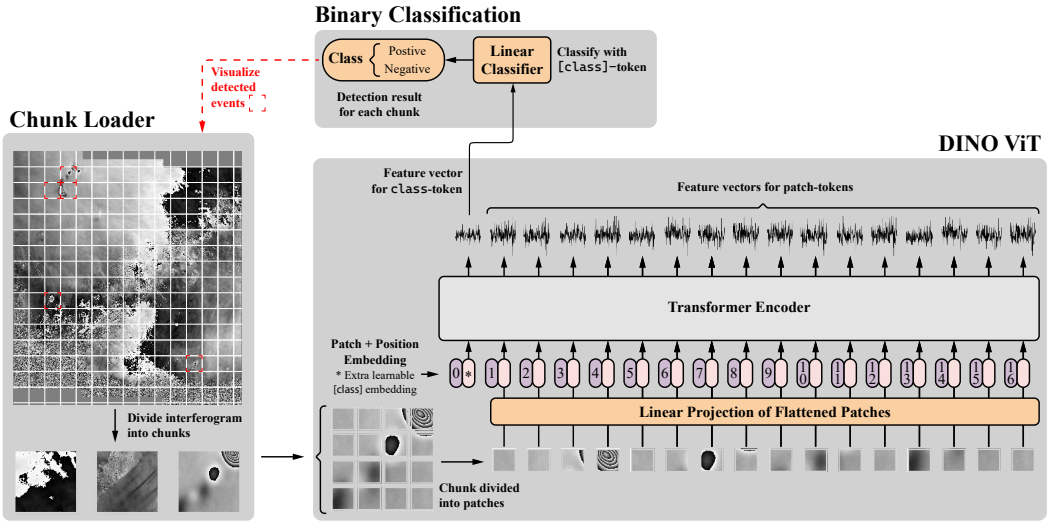


Figure 3.3: **Illustration of SGLNet implementation.** The "Chunk Loader" takes the entire interferogram and transforms it into smaller 3-channel images (chunks). "DINO ViT" takes batches of chunks and infers feature vectors for both the `[class]` token and the patch tokens. The `[class]` feature vector is passes on to "Binary Classification" where a linear network predicts SHA events in the chunks (positive/negative). Coordinates of positive chunks are collected to map regions of interest (visualized as red highlights in the figure).

in the third. 'Recta' is given from the unit rectangular coordinates in the first two channels, i.e $\cos(\cdot)$ and $\sin(\cdot)$ of the phase, and zeros in the third channel. The last format is named 'Blend' and is created by the IPP with a pseudocolor palette that blends phase and magnitude together - the interferometric phase determines hue, the magnitude controls brightness, and saturation is fixed at 0.75.[5]

The interferogram is then sampled with a moving window to divide into chunks. This is sketched in Figure 3.5. The window has a size of (M_{cs}, M_{cs}) where M_{cs} is the chunk size and uses a stride of $M_{cs}/2$. Thus, the interferogram is divided into chunks, with each chunk being an RGB image $\in \mathbb{R}^{3\times M_{cs}\times M_{cs}}$. The interferogram is padded so that the dimensions of the interferogram are divisible by $M_{cs}/2$.

The CL also filters the chunks. This is mainly used in regions where grounding lines are present, in which case a mask of the grounding line can be supplied to discard chunks overlapping the mask. This improves performance and ensures that grounding lines are not mistaken to be SHA events.[6]

[5] In the actual Python code, the format 'Blend' is referred to as 'Blend'. This name was chosen early during development, but have been changed in this report to avoid confusion.

[6] It is also used during training to exclude ambiguous chunks - see Section 3.3.

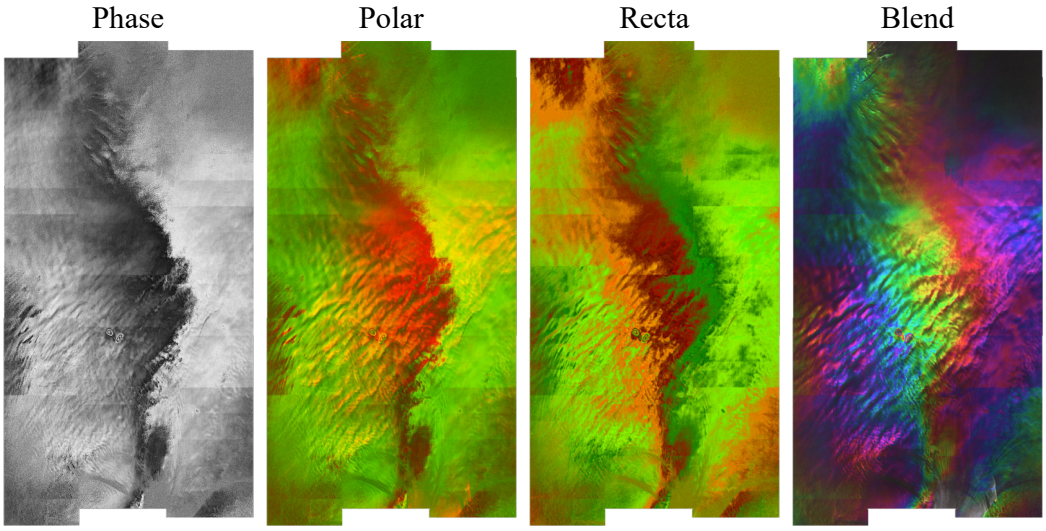


Figure 3.4: **Four ways that interferograms are represented as RGB images.** They are referred to as 'Phase', 'Polar', 'Recta', and 'Blend'.

The remaining chunks are collected into batches to be passed to the DINO ViT.

DINO ViT. The DINO network allows for some variability in it's architecture, with each change requiring a different set of pretrained weights. The authors of DINO [31] have published weights for four different configurations of their ViT network.[7] These four configurations are ViT-S/16, ViT-S/8, ViT-B/16, and ViT-B/8. The 'ViT' denotes that it is for their Vision Transformer model. The 'S' and 'B' distinguished between the size of the model, being 'Small' and 'Base', respectively. The model size determined the embedding dimension and the number of parallel attention heads. The '8' and '16' refer to the patch size, M_{ps}, that each chunk is divided into before embedding. Each patch is thus (M_{ps}, M_{ps}). Smaller patches means higher spatial resolution, but also more tokens to pass through the network.

All DINO configurations was trained by [31] on images with a resolution of 224^2. Nevertheless, DINO does also work for larger images. Larger images means more patches $\rightarrow$ more tokens in each dimension. This does not break the transformer architecture itself, but the position embeddings that the network learned during training are no longer meaningful. Therefore, DINO interpolates position embeddings to accommodate large images. In this work, both $M_{cs} = 224$ and $M_{cs} = 448$ have been used.

[7] https://github.com/facebookresearch/dino

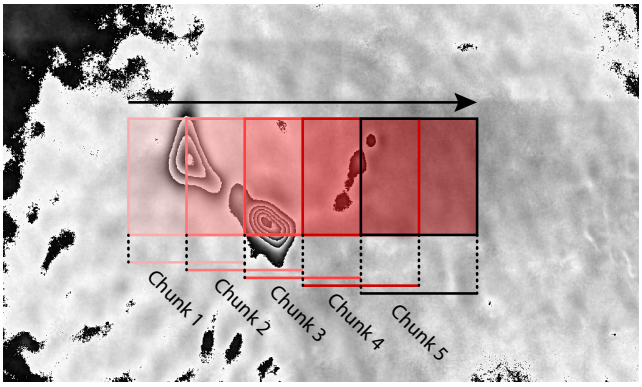


Figure 3.5: **Moving window sampling of interferograms.** The Chunk Loader samples the interferogram with a moving window and a 50% overlap between each sample in both directions.

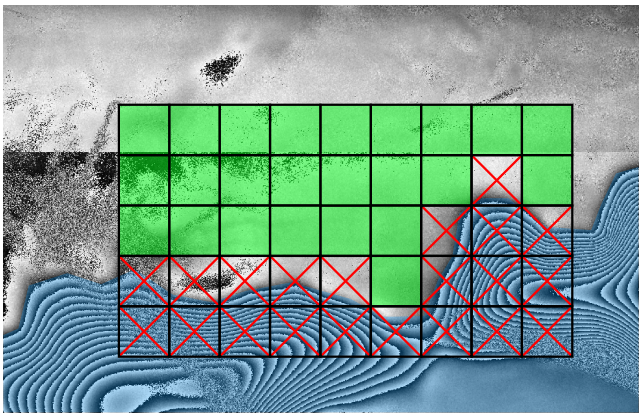


Figure 3.6: **Chunk filtering with mask.** The chunk loader can filter out specific chunks using a binary mask. Here is shown which chunks would be kept (green fill), and which would be discarded (red cross) when supplied with a grounding line mask (blue polygon).

A summary of DINO network configurations are given in Table 3.1. It is clear that the choice of configuration will influence the performance of the network. Consider for example the configuration with ViT-S/16 and chunk size 224, compared to ViT-B/8 and chunk size 448. We may expect the first configuration to do worse in terms of inferring useful features, as it has only half the embedding dimension, half the number of attention heads, and a larger patch size (coarser "resolution"). On the other hand, the latter configuration is more computationally demanding. This configuration has a sequence length that is 16 times larger and uses four times as many parameters. Testing all configurations to determine optimal trade-off therefore seems relevant.

The inner workings of ViTs and the DINO network is explained in Section 2.3.1 and Section 2.3.3 and is not expanded upon here. It is relevant to add, however,

model	depth	dim	heads	#tokens $M_{cs} = 224$	#tokens $M_{cs} = 448$	#params
ViT-S/16	12	384	6	197	785	21M
ViT-S/8	12	384	6	785	3137	21M
ViT-B/16	12	768	12	197	785	85M
ViT-B/8	12	768	12	785	3137	85M

Table 3.1: **DINO network configuration.** "Depth" is the number of Transformer blocks, "dim" is the embedding dimension and "head" is the number of parallel heads in the multi-head attention. "# tokens" is the length of the token sequence for input chunk (M_{cs}, M_{cs}). "# params" is the total number of parameters (weights) in the model.

that DINO does not just output the `[class]` token directly (which would be a vector with 384 elements for ViT-S and 768 elements for ViT-B). Instead, when ViT-S is used, the network concatenated the `[class]` token from the last four Transformer blocks to form an output feature vector with $4 \times 384 = 1536$ elements. When ViT-B is used, the network takes all tokens (both `[class]` and patch tokens) from the last transformer layer and concatenated the `[class]` token with an average over all the patch tokens, resulting in an output feature vector of $768 + 768 = 1536$ elements. This was determined to be the optimal output format by the DINO authors [31].

Classifier head. The 1536-dimensional feature vectors from DINO are passed to a binary classifier. This is a separate network consisting of a single fully connected layer. It resembles an MLP[8], but in this implementation does not have any hidden layers. The output is a single neuron with sigmoid activation (2.37) to obtain binary predictions. The structure is shown in Figure 3.7.

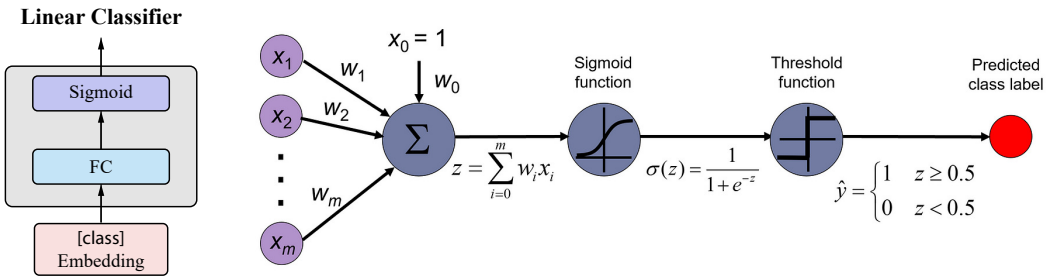


Figure 3.7: **Binary classifier network.** The fully connected (FC) layer is a single-output linear layer with sigmoid activation. Binary predictions are obtained via thresholding. The network illustration on the right is from [37].

[8] See Section 2.3.1

3.2 Semantic segmentation

As mentioned in Section 2.3.1, a well-trained ViT can capture global contextual information through the attention mechanism, and encode salient image features into patch embeddings. This opens the door for semantic segmentation, which is an area where DINO has great potential. One study [38] even found that the DINO model by Caron et. al. [31] is better at semantic segmentation than its successor, DINOv2, by Oquab et. al. [39].

This work implements semantic segmentation routines using a number of different approaches. The first is a self-attention segmentation routine introduced in the original DINO paper [31]. The second is a principal component analysis (PCA) of embeddings. This is implemented both for patch embeddings and key, query and value embeddings in the last attention block. Finally, a modified version of methods in *Localizing Objects with Self-Supervised Transformers and no Labels* (LOST) [40] is implemented. Two modified versions are implemented and each works on both keys and values.

The "resolution" of the attention mechanism corresponds to the spatial arrangement of patches. As a result, the segmentation masks can only resolve patches, not individual pixels. The size of the segmentation mask, M_{feat}, is given by

$$M_{feat} = \frac{M_{cs}}{M_{ps}} \tag{3.2}$$

For example, with chunk size $M_{cs} = 224$ and patch size $M_{ps} = 16$ the segmentation mask has size $M_{feat} = 14$ equivalent to the 14×14 patches that cover the entire chunk. The mask can then be upscaled by the patch size to match the original chunk resolution. This is illustrated in Figure 3.8.

Self-attention thresholding. The attention scores (2.24) from the last attention layer is extracted, yielding a matrix $\mathbf{A}$ for each of the h attention heads. For each matrix, the attention scores between the `[class]` token and all patch tokens are selected and reshaped into an attention map. The resulting matrix has shape $\mathbb{R}^{h \times M_{feat} \times M_{feat}}$. Each $M_{feat} \times M_{feat}$ map indicates how strongly the `[class]` token attends to each image patch for that particular head. Each map is normalized (so attention scores of each map sums to 1), and then thresholded to keep 60% of the attention mass. The final segmentation mask consists of patches that the `[class]` token attended to in at least half of the attention heads.

PCA of embeddings. The PCA segmentation is conceptually straightforward. For each patch, a corresponding embedding vector is extracted. The

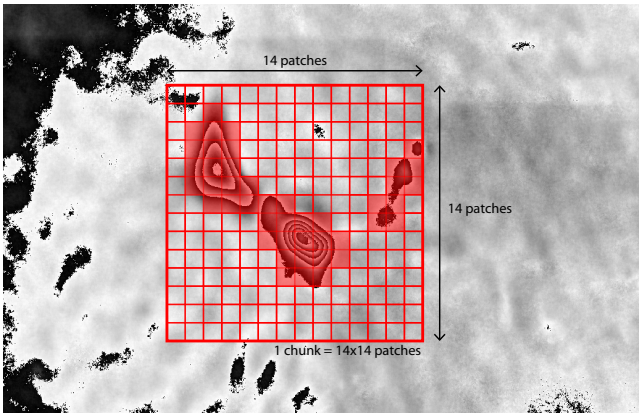


Figure 3.8: **Segmentation mask in relation to image chunk.** Masks are created from attention embeddings, so for a chunk of size $M_{cs} = 224$ with patch size $M_{ps} = 16$, the segmentation mask is $M_{feat} = 14$. It can be upscaled by the patch size to return to the size of the chunk, but the resolution is still limited by the size of patches.

first principal component (PC) across all patches is computed, and a segmentation mask is obtained by thresholding the resulting PC values.
Four types of embeddings can be used. Most directly, the output patch tokens from the ViT, each with embedding dimension D, can be rearranged into a feature map of shape $\mathbb{R}^{D \times M_{feat} \times M_{feat}}$. Alternatively, embedding can be derived from the **Q**, **K**, or **V** matrices (2.22) from the last attention block.
For instance, if **Q** embeddings are used, each of the h attention heads produces an embedding of dimension $D_h = D/h$ per patch. This yields h feature maps of shape $\mathbb{R}^{D_h \times M_{feat} \times M_{feat}}$. Concatenating these along the embedding dimension results in a single feature map of shape $\mathbb{R}^{D \times M_{feat} \times M_{feat}}$, identical to the patch token case. Applying PCA along the embedding dimension and thresholding the first PC yields a segmentation mask of size $M_{feat} \times M_{feat}$.

Patch similarity. Another segmentation method is implemented by modifying the LOST object discovery algorithm [40]. This method leverages cosine similarity between patch embeddings to distinguish foreground from background.[9] Similar to the PCA method, embeddings are extracted from either the **K** or **V** matrices (2.22) of the last attention block. Concatenating all h feature maps along the embedding dimension yields a single feature map of shape $\mathbb{R}^{D \times M_{feat} \times M_{feat}}$. The similarity is calculated as the dot product between every pair of patch embeddings. A seed patch is selected as the patch least similar to all others (i.e. the patch that stands out most from the background). Assuming this seed belongs to the foreground, a segmentation mask is created

[9] In the implementation in Python, this method is referred to as 'Correlation' though 'Similarity' would have been more fitting.

by thresholding the similarity between the seed patch and all other patches. Alternatively, instead of relying on similarity to a seed patch, the *degree* can be used. For this, each patch is assigned a value equal to the sum of its similarities with all other patches. The inverse of this sum is referred to as the degree. Patches that are very dissimilar to others have smaller summed similarity and thus higher degree values. this degree map has shape $\mathbb{R}^{M_{feat} \times M_{feat}}$ and can be thresholded to yield a segmentation mask.
A major limitation of this method is its assumption that foreground elements occupy fewer patches than the background. If the foreground spans more patches than the background, then the background patches become the least similar and are incorrectly selected as seed patches or assigned large degrees. This assumption does not hold for SHA events.

3.3 Training and testing

Unlike the pretrained DINO ViT, the binary classifier head requires training. Once trained, the performance of the combined DINO backbone and classifier head must be tested. A dataset of manually labelled interferograms is created for this purpose. About 80% of the dataset is used for training, 10% is used for validation during training (so combined training/validation dataset makes up approx. 90% of all data), and the last 10% is data from satellite tracks not included in the training data, which is used to test performance. The dataset itself is described in Section 4.2. Ground truth (GT) labels are made in the form of masks - one mask for each interferogram. The mask contains segmentations of SHA events. During training, the DINO feature vector from each chunk is passed through the classifier, and the manual segmentation mask is used to determine the GT label, $\{0, 1\}$, based on whether the chunk overlaps any masked SHA events.

Handling uncertainty. Manually segmented GT labels introduce a problem. Consider the three crop-outs of SHA events in Figure 3.9. These are from real interferograms and show that real signals are not always easy to interpret. While the left-most fringe-pattern is easy to identify, the signal in the middle is weak, and the signal on the right is noisy. A question arise: how should these ambiguous signals be segmented? A binary mask leaves no room for uncertainty - regions are labelled (0) or (1) with full certainty - but the noisiness and strength of signals has no clear cutoff. As what point does a SHA pattern become too noisy to reasonably be labelled as a detectable event? And likewise for the strength of SHA patterns. The choice will inevitably be arbitrary and include a bias from the person doing the segmentations. To make matters worse, having uncertain events be labelled with full certainty are likely to confuse the classifier network and limit it's ability to generalize. It has been

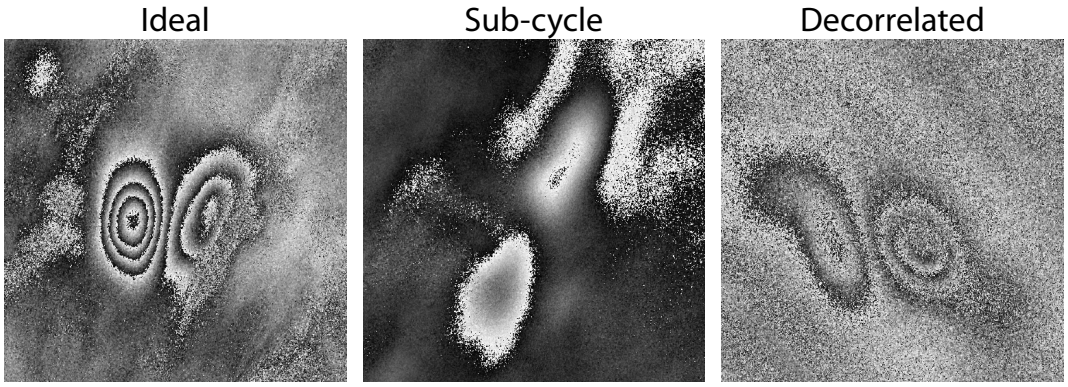


Figure 3.9: **Variety of SHA signals in real interferograms.** (left) An ideal SHA pattern, where multiple fringes (module-2π discontinuities) are present. (middle) Weak SHA pattern, where small deformation causes a phase shift below a full 2π-cycle. (right) Noisy SHA pattern, caused by interferometric decorrelation.

found through trial and error that the best approach is to exclude all chunks where the proper label is ambiguous. In this way, the network only learns from reliable, labelled data. This is implemented by including an additional manual segmentation mask over ambiguous regions. Similar to the grounding line mask, the network excludes chunks overlapping ambiguous regions during training.

In additional to these ambiguous regions, another form of uncertainty exists: As interferograms are split into small 224×224 or 448×448 chunks, how much does a SHA event need to overlap a chunk, before that chunk is labelled as SHA positive? Deciding that a single pixel grants a "positive" label would inevitably lead to a lot of falsified false negatives. The network will simply not be able to predict a positive label from a single pixel. The opposite is true, if chunks are instead labelled as SHA negative, if a SHA event is not within e.g. the central 50% of the chunk. This would instead induce falsified false positives, as the network correctly identifies SHA patterns near the chunk margin, but the ground truth label says otherwise. As before, binary labelling requires us to make a decision with no obvious solution. Any choice of minimum overlap will introduce falsified false positives and negatives, and this will negatively affect training and the networks ability to generalize. We alleviate this during training by excluding all chunks where SHA events only overlap the outer 50% of the chunk. Then, only two scenarios remain: either a chunk contains a SHA event within the central 50% of its area and it is annotated as "positive", or a chunk contains no SHA event at all and it is annotated as "negative".

In summary: during training, the algorithm makes a decision about chunk labels and whether chunks should be excluded based on three masks: one masking reliable SHA patterns, one masking ambiguous SHA patterns, and

one masking grounding lines. The decision is made from a hierarchical set of rules, which are illustrated in Figure 3.10. Excluding some chunks altogether improves the reliability of the ground truth labels, from which the network learns to make reliable predictions.

Training SGLNet. SGLNet uses pretrained parameters for the DINO ViT. It is theoretically possible to do finetuning on these parameters, to tailor the model(s) towards the given task, but it has not been pursued for this work. The technical term is that we use DINO with a frozen backbone. This means that only the classifier network requires training. One trained model is needed for each ViT configuration - not because the actual classifier architecture changes when the ViT changes, but because feature vectors (i.e. the classifier inputs) change when the ViT changes. The DINO backbone has four configurations[10], which increases to eight when input resolutions of both 224^2 and 448^2 are implemented. Furthermore, four different image representations are used[11], pushing the number of different model configurations up to 32 combinations. The total number of classifier networks to be trained is therefore 32.

The training process is divided into two steps. First, for each of the 32 configurations, feature vectors for all chunks across the entire training dataset are computed with DINO. At the same time, ground truths labels are associated with each feature vector. These features and corresponding labels are then loaded into the classifier network where training occurs. The classifier uses binary cross-entropy loss (2.44) and class weights are included to account for class imbalance.[12] Model parameters are updated with SGD using a momentum of 0.9. The learning rate is $lr = 0.001$ with cosine annealing. The network is trained on batches of size 128 over 100 epochs. No early stopping is used, but model parameters and performance metrics are stored for all epochs. The final model parameters are chosen to be the first epoch with the highest F_1-score (2.54) on the validation data.

With 32 network configurations to be tested, this work did not find time to also test how training hyperparameters affect performance. All inference and testing is done on DTU's HPC. In all cases, only a single NVIDIA Tesla V100 GPU with 16GB memory is used, together with four CPU cores.

Testing SGLNet. As mentioned at the start, testing is performed on 10% of the total dataset, and this data is from tracks that are not used in the remaining 90% training and validation data. Testing only requires the model to run inference, so no gradients are computes and no backpropagation performed.

[10] See Section 3.1.2 (DINO ViT).
[11] See Section 3.1.2 (Chunk Loader).
[12] See Section 4.3.

Both classification and segmentation performance are measured with metrics described in Section 2.3.4.

The classifier performance is measured with precision (2.53), recall (2.51) and the combined F_1-score (2.54). Performance on test data both with and without exclusion of ambiguous regions is reported. Segmentation performance is given as the Dice coefficient (2.55) for each of the segmentation methods described in Section 3.2 (9 in total). For both classification and segmentation, the performance metrics are calculated as the global average over all chunks from all satellite images in the test dataset.

Finally, the performance of the network as a whole is tested. These tests focus on the primary goal of SGLNet—to produce boundary boxes (coordinates) around SHA events in Sentinel-1 interferograms. For this, the total number of lakes in the test data, and the number of lakes captured in the output event coordinate files are computed. A lake is considered captured only if the entire GT segmentation is within the predicted coordinate box. The number of empty coordinate boxes are also counted, and the algorithm inference time is measured.

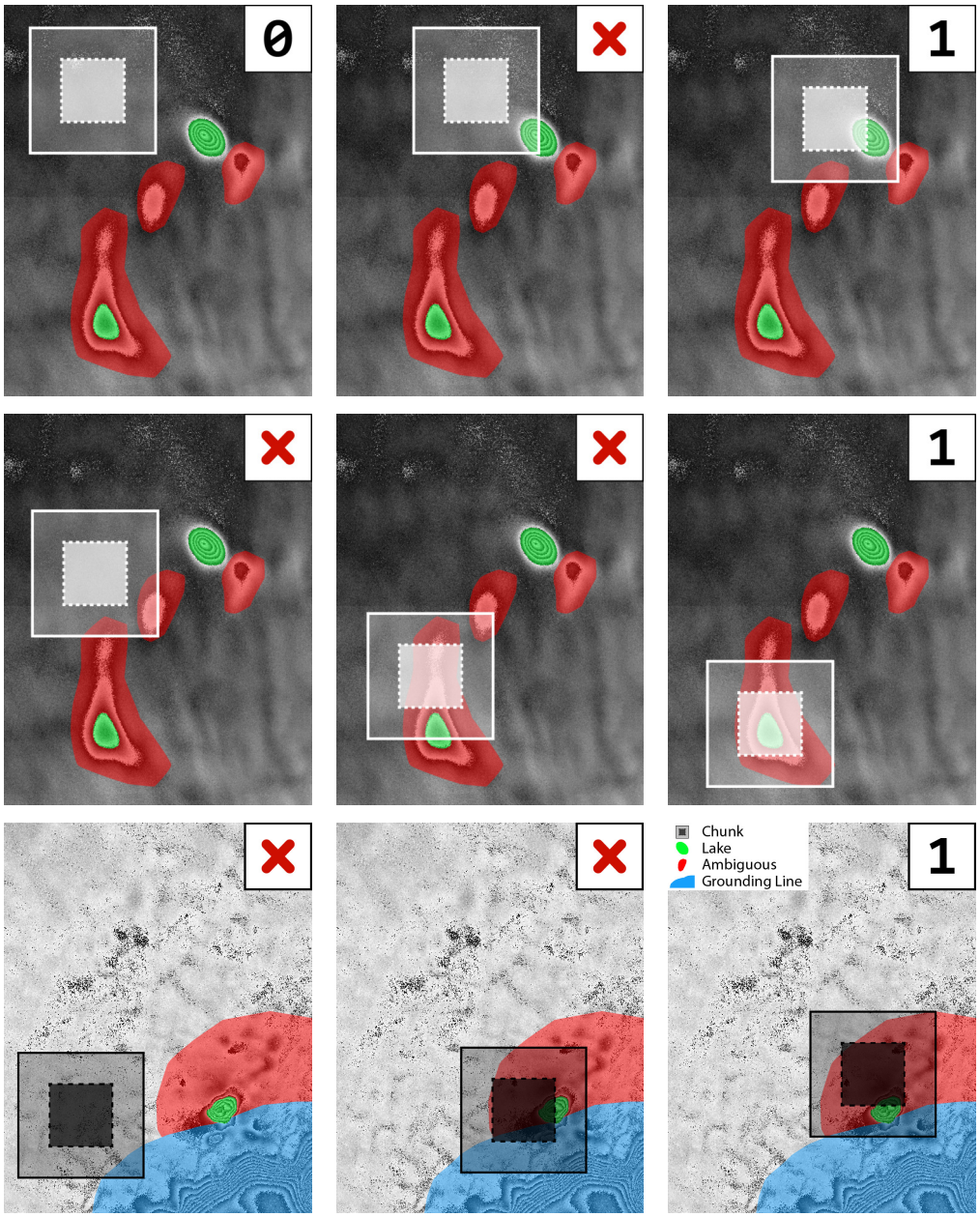


Figure 3.10: **Ground truth labelling with manual segmentation masks.** Masks are coloured as follows: green = well-defined SHA pattern, red = ambiguous SHA pattern, blue = grounding line region. The transparent white/black square shows a single chunk (in this case a 448 px × 448 px window) representing the image sample passed to the ViT. The inner square marks the centre 50% of the chunk, which is used for decision making. In the upper right corner is shown the ground truth class label assigned to the given chunk: `0` is SHA negative, `1` is SHA positive, and red cross means chunk is excluded.

Chapter 4

Data and study area

Training a neural network requires a dataset. This dataset should constitute three subsets: a training set, a validation set, and a testing set. The training dataset contains most data and is used to update model parameters. The validation dataset is also included during training, but used to evaluate performance and tune hyperparameters. The testing dataset is used to get an unbiased estimate of the final model performance.

This chapter presents the study area from where the dataset is formed.

4.1 Study area

Sentinel-1 delivers data from much of Antarctica (Figure 2.5), though with varying frequency. Using all this data is unfeasible for this work. Instead, six different S1 tracks covering four distinct regions have been chosen. These are: three tracks over the Pine Island Glacier, one track over Totten Glacier, one track over Jutulstraumen Glacier, and one track over Cook Ice Shelf. More information about each track is gathered in Table 4.1 and shown in Figure 4.1.

Location	#Data	Track	Direction	Frame	Extent [km]
Pine Island	12	038	Descending	861	190×250
Pine Island	41	065	Descending	901;906	350×250
Pine Island	31	169	Descending	866	210×250
Jutulstraumen	220	002	Ascending	925	200×250
Totten	86	099	Ascending	946	180×250
Cook	8	010	Ascending	939;944	370×250

Table 4.1: **Summary of included tracks.** 'Location' is the name of the glacier that is covered. '#Data' is the number of acquisitions included. 'Track' is the relative orbital path. 'Direction' indicates whether S1 is moving north or south. 'Frame' references a specific segment of the track, with neighbouring frames being five numbers apart. 'Extent' is a rough estimate of the along-track $\times$ cross-track dimensions of the product.

These areas are identified in Project DynaMelt as regions of interest, with data already processed by the IPP. These regions are located upstream of large ice shelves and known to have subglacial hydrological activity, which makes them relevant for this work. Regions are displayed in more detail in Figure 4.2, where S1 tracks are plottet on top of ice velocities and modelled subglacial water flux. The figure shows how these tracks cover areas of glaciers that feed into ice shelves, and that these areas are expected to have active subglacial hydrology.

Sentinel-1 SLC data from IW mode has a resolution in range and azimuth of 2.7 m $\times$ 22 m to 3.5 m $\times$ 22 m. The scene is oversamples with a pixel spacing of 2.3 m $\times$ 14.1 m. The IPP uses 10×2 looks in range and azimuth, respectively, resulting in interferograms with a pixel spacing of 23 m $\times$ 28.2 m. A range spacing of 23 m becomes a ground range spacing of 32 m to 47 m depending on the incidence angle [41]. More details about the S1 SLC IW product in Appendix A.2.

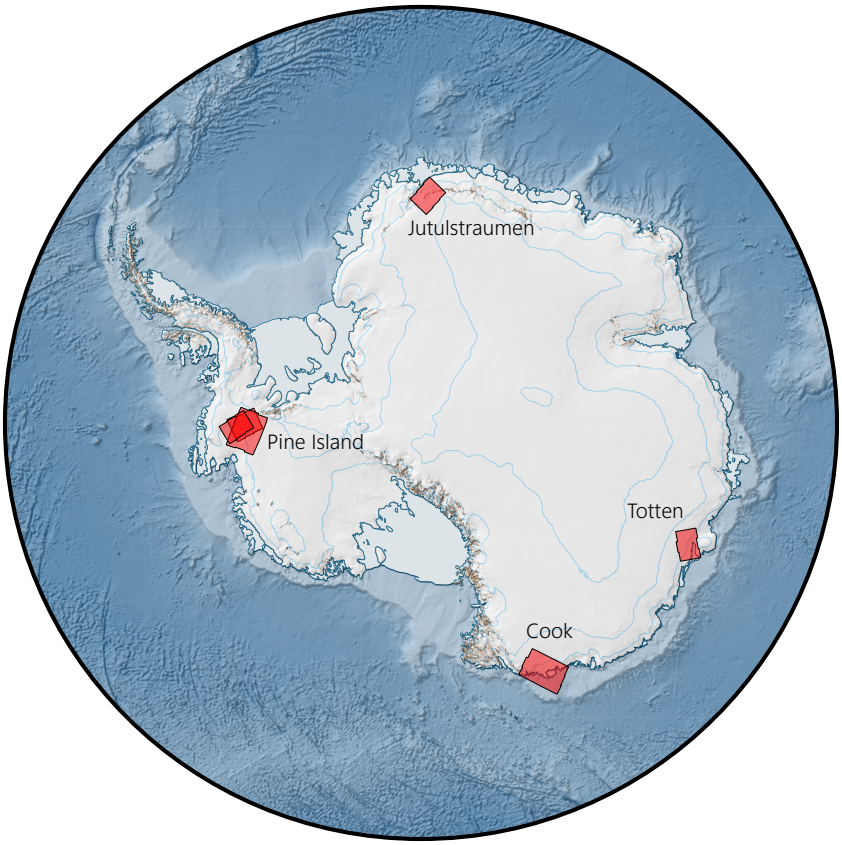


Figure 4.1: **Overview of study areas.** Six tracks are chosen in total. Three at the Pine Island Glacier, one at Totten Glacier, one at Cook Ice Shelf Bassin, and one at Jutulstraumen Glacier. Illustration is made with Quantarctica [21].

4.2 Dataset for training and testing

A total of 398 interferograms are included in the dataset, as stated in Table 4.1. These are divided into two subsets, where one set is used for training and validation, and the other set is used for testing. Tracks 002, 038, 099, and 169 make up the training/validation dataset, while the testing dataset is composed of tracks 010 and 065. In this way, the testing dataset makes up 12.3% of the total dataset, and consists solely of data from tracks that the model has not seen during training. This also splits interferograms with grounding lines (GL) nicely between training and testing data, as both track 010 and 099 include prominent GL. This may not matter that much, since GLs are masked out anyway, but it ensures that both sets are representative. The interferograms are split into small chunks before entering the ViT, so the effective size of the dataset in relation to the ML network, is not the number of interferograms but the total number of these chunks. We only need to consider that enough interferograms are included to ensure a diverse set of scenes. The dataset size

is summarized in Table 4.2. It is seen that even though the testing dataset only contains 12% of interferograms, it ends up containing roughly 20% of all chunks in the dataset. This is because interferograms from different tracks have different sized, thus splitting into various numbers of chunks. As was given in Table 4.1, both track 010 and 065 are rather large tracks, made by combining two neighbouring products (frames), hence why the relative size of the training dataset grows when interferograms are split into chunks.

		#Chunks			
		$M_{cs} = 224$		$M_{cs} = 448$	
Subset	$\#I_{dd}$	All	w/o amb.	All	w/o amb.
Training	349 (88)	1008k (72)	989k (72)	253k (72)	243k (71)
Validation		112k (8)	110k (8)	28k (8)	27k (8)
Testing	49 (12)	279k (20)	276k (20)	72k (20)	70k (21)
Total	398	1400k	1375k	352k	341k

Table 4.2: **SGLNet dataset.** Interferograms I_{dd} are divided into two/three subsets: one for training and validation, and one for testing. Interferograms are split into small chunks before the ViT, each yielding thousands of chunks. Filtering is (optionally) applied to exclude ambiguous (amb.) chunks.[1] Number of chunks (Chunks) with and without filtering is given for chunk sizes $M_{cs} = \{224, 448\}$. Grounding lines are always removed, so 'All' gives the number of chunks after removing those. Percentages with respect to the entire dataset are written in parenthesis. Chunks are given in numbers of thousands (indicated by 'k').

4.3 Manual masking

The binary classifier is trained with supervision. In Section 3.3 the training (and testing) method is described: each interferogram requires up to three manually segmented masks. A mask with segmented SHA events is always needed, in order to have ground truth labels. Any interferogram with grounding lines must have an additional mask covering these. Optionally, a mask with ambiguous/uncertain areas can be given to exclude chunks from within these areas. The algorithm then excludes chunks based on a set of rules, as illustrated back in Figure 3.10.

Masks with segmentations of SHA events (ground truths) and segmentations of ambiguous areas have been produces for all 398 interferogram. Grounding lines does not appear in the tracks at Pine Island, so GL masks are only needed for the other three tracks. It was found that a single GL mask can be used for

[1] Chunk sampling is described in Section 3.1.2. Chunk filtering is described in Section 3.3.

all interferograms of the same track, as long as masks are padded to account for GL motion. Network performance is also tested using these masks, for both classification and segmentation.

Most interferograms in the dataset have some SHA events. Nevertheless, these events are still scarce in the data as a whole. This results in a large class imbalance, with a lot more 'negative' than 'positive' instances. Table 4.3 summarize the label distribution for the testing and validation data. When the network is trained, any chunks marked as ambiguous are always excluded. Values given in the table are therefore the resulting numbers after chunks have been filtered. It is seen that the positive:negative class ratio is more than 1:1600 for chunk size $M_{cs} = 224$ and around 1:670 for chunk size $Mcs = 448$. These are huge class imbalances and must be accounted for in the loss function. They also makes metrics such as accuracy meaningless, as the network can theoretically obtain 99.9% accuracy from just predicting all chunks to be negative. That is why we use precision, recall and F_1-score.

	$M_{cs} = 224$			$M_{cs} = 448$		
Subset	#Pos	#Neg	Ratio	#Pos	#Neg	Ratio
Training	600	988276	1:1647	363	242862	1:669
Validation	67	109808	1:1639	40	26985	1:675
Total	667	1098084	1:1646	403	269847	1:670

Table 4.3: **Summary of labels in training and validation dataset.** The two subsets are created by splitting the combined set into two, with 90% of chunks going to the training set, and 10% going to the validation set. The number of positive and negative instances is shown for chunk sizes $M_{cs} = \{224, 448\}$ together with the corresponding ratio (imbalance between classes).

A summary of label distribution for the testing data is given in Table 4.4. Unlike data used during training and validation, testing is performed on data both with and without ambiguous (amb.) chunks. The table therefore shows values for both scenarios.

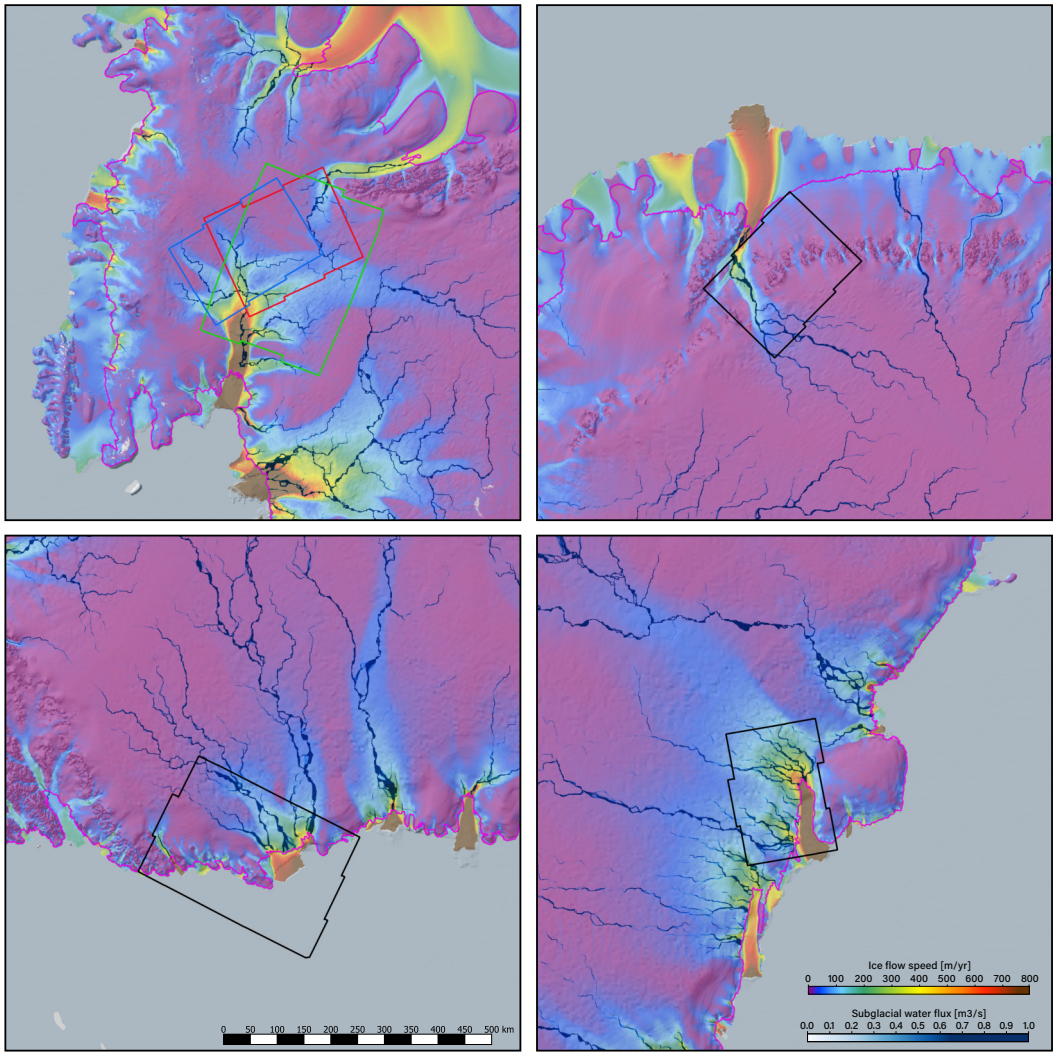


Figure 4.2: **Zoomed view of study areas.** (top left) Pine island. Track 038 is coloured blue, track 065 is green, and track 169 is red. (top right) Jutulstraumen, track 002. (bottom left) Cook, track 010. (bottom right) Totten, track 099. Track outlines are plottet together with ice velocities [42], subglacial water flux [43], grounding lines in purple [44], and DEM virtual hillshade [45]. Illustration is made using Quantarctica [21].

	$M_{cs} = 224$			$M_{cs} = 448$		
Chunk sample	#Pos	#Neg	Ratio	#Pos	#Neg	Ratio
All	860	278521	1:324	410	71237	1:174
w/o amb.	116	276119	1:2380	56	70196	1:254

Table 4.4: **Summary of labels in the testing dataset.** Column 'Testing chunks' specifies whether ambiguous chunks were excluded. The number of positive and negative instances is shown for chunk sizes $M_{cs} = \{224, 448\}$ together with the corresponding ratio (imbalance between classes).

Chapter 5

Results and comparison

Results from this work are given in three main sections. First, we report the performance of SGLNet evaluated using the testing dataset. Second, we present additional findings regarding network behaviour. Third, we include results from Project DynaMelt, where SGLNet have been used to map the subglacial hydrological system in parts of Antarctica.

5.1 SGLNet performance

As outlined in Section 3.3 and Section 4.2, SGLNet is evaluated on 49 interferograms from unseen data tracks. The evaluation focused on Three main aspects of the algorithm's performance.

First, the classification ability of the neural network, composed of DINO ViT and binary classifier, was assessed at the chunk level. This provides insight into how effectively the network identifies SHA events within individual image chunks.

Second, the event detection performance was examined. This tests the entire algorithm in relation to the processing chain of Project DynaMelt. This includes evaluating the accuracy of SHA events captured using output boundary boxes as well as assessing the computational efficiency.

Third, the segmentation capabilities of the algorithm were tested. Although this is an experimental component, it investigates the potential of using attention maps from the ViT to segment SHA events.

Examples illustrating these three types of output are presented in Figure 5.1.

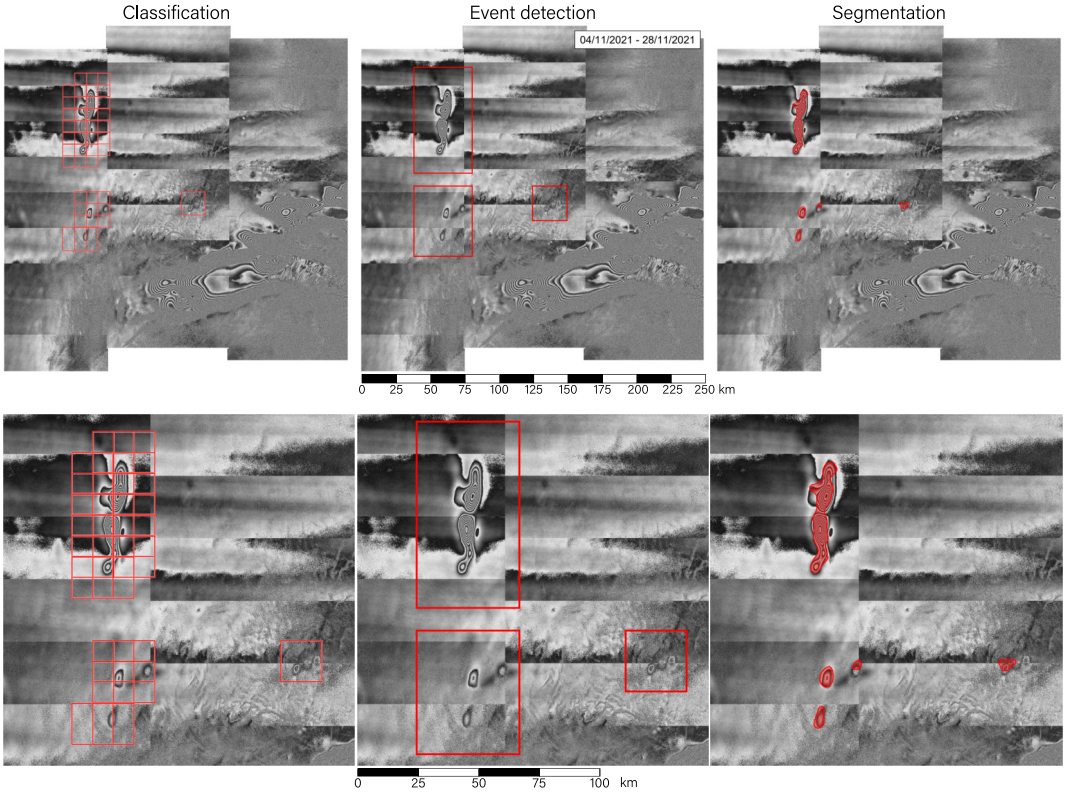


Figure 5.1: **Different SGLNet outputs.** (top row) Full DD interferogram. (bottom row) Zoomed-in view. (left column) Chunks predicted to be "positive". (middle column) Event coordinates capturing entire SHA events—the main output in relation to Project DynaMelt. (right column) Segmentation masks and outlines using self-attention thresholding. A large grounding line is seen in the bottom right. All outputs use ViT-B/16 Phase/224. The interferogram is from Totten (T099A), formed with SLC data spanning 04/011/2021–28/1/2021.

5.1.1 Classification performance

Evaluating chunk classification provides a direct measure of the network's performance in isolation—independent of its downstream applications—for the task it was explicitly trained to do.

A total of 32 different network configurations have been tested in this work.[1] A small empirical example of how these configurations affect chunk classification is shown in Figure 5.2. Even within this limited qualitative example, it becomes clear that changes in network configuration can influence the predictive outcome.

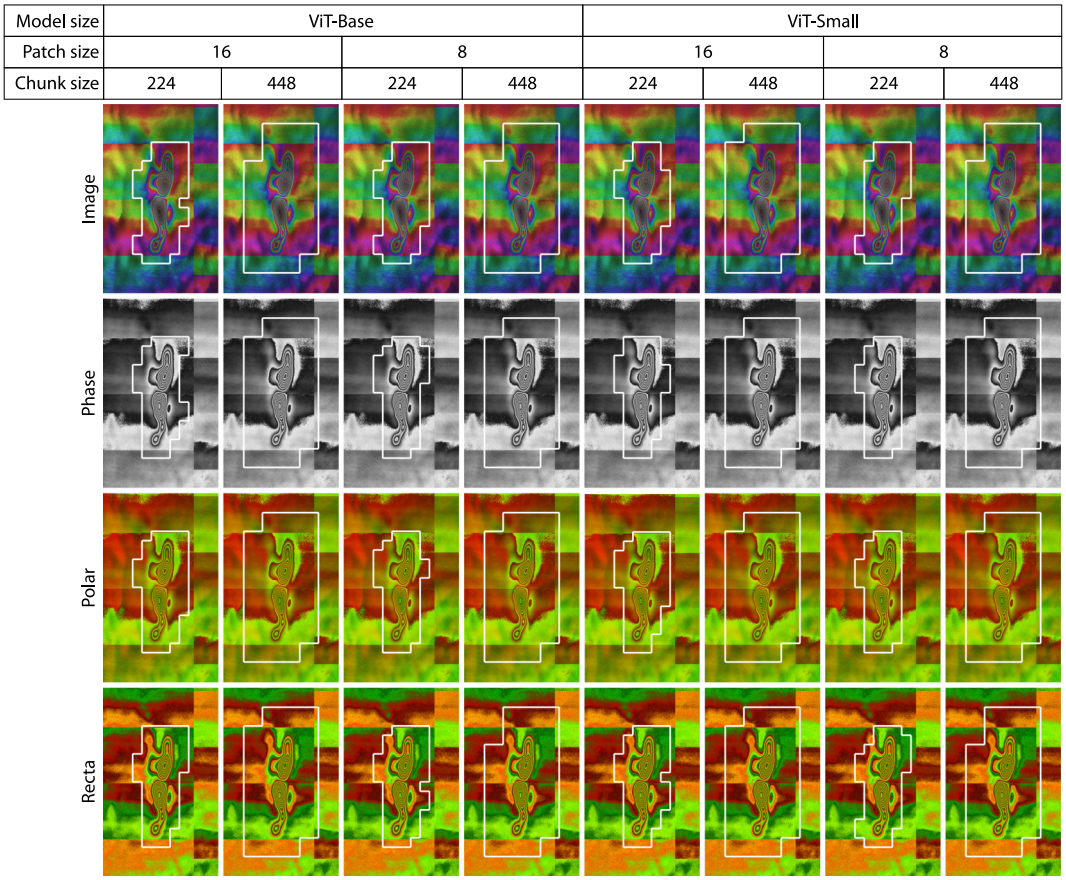


Figure 5.2: **Outlined chunk predictions from all 32 network configurations.** Each square represents one of the 32 tested network configurations. Predictions of individual chunks are not shown, but all chunks marked as "positive" are instead outlined for visual clarity. Each square represents a region of approx 55 km × 45 km cropped out of the interferogram from Totten (T099A), formed with SLC data spanning 04/011/2021–28/1/2021.

To evaluate the impact of these configuration systematically, an ablation study

[1] See Section 3.1.2 for details about the 32 network configurations.

was carried out using recall, precision and F_1-score as the main performance metrics.[2] The results are summarized in Table 5.1.

As previously discussed in Section 3.3 on handling uncertainty, the ground truth class labels used for evaluation are affected by a degree of ambiguity. To address this, performance is reported under two scenarios: one excluding ambiguous chunks and one including them. Excluding uncertain samples mirrors the network's training conditions, but also modifies the underlying data distribution, which may lead to inflated performance estimates. Including all chunks, on the other hand, yields a more representation dataset at the cost of label certainty, which is likely to cause performance underestimation. The next paragraph provides further justification for the interpretation that including all chunks results in a conservative estimate of true performance. Taken together, these two evaluations can be considered as approximate upper and lower bounds, with the actual performance of the model expected to lie somewhere in between.

Several trends are observed in Table 5.1. The highest F_1-score is achieved using the ViT-B/16 architecture when trained on the "Phase" input format. More generally, the "Phase" format consistently outperforms the other three input formats across nearly all ViT configurations. Furthermore, the ViT-B model tends to outperform its smaller counterpart, ViT-S, which is to be expected given its larger embedding space and greater number of attention heads. However, the results reveal little to no performance improvement when using a smaller patch size of 8 compared to 16; in fact, smaller patches may slightly degrade performance. When it comes to chunk size, no consistent trend is evident. Although minor variations are observed, these may stem from shifts in the dataset's statistical distribution introduced by changing the chunk size.[3]

[2] See Section 2.3.4 for details about performance metrics.
[3] See Section 4.3 for details about the dataset distribution.

				w/ chunk exclusion			w/o chunk exclusion		
		Form.	Size	precision	recall	F_1-score	precision	recall	F_1-score
Architecture: **ViT-B**	Patch Size: **8**	Blend	224	0.789	0.966	0.868	0.711	0.409	0.520
			448	0.567	0.982	0.719	0.643	0.515	0.572
		Phase	224	0.927	0.991	0.958	**0.800**	0.487	0.605
			448	0.815	0.946	0.876	0.772	0.505	0.611
		Polar	224	0.858	0.940	0.897	0.749	0.381	0.505
			448	0.800	0.929	0.860	0.767	0.466	0.580
		Recta	224	0.872	**1**	0.932	0.762	0.477	0.587
			448	0.812	**1**	0.896	0.748	0.515	0.610
	Patch Size: **16**	Blend	224	0.713	0.983	0.826	0.685	0.430	0.529
			448	0.700	**1**	0.824	0.676	0.520	0.588
		Phase	224	**0.966**	0.983	**0.974**	0.770	0.449	0.567
			448	0.902	0.982	0.940	0.771	**0.566**	**0.653**
		Polar	224	0.874	0.957	0.914	0.772	0.410	0.536
			448	0.794	0.964	0.871	0.746	0.507	0.604
		Recta	224	0.839	0.991	0.909	0.740	0.473	0.577
			448	0.836	**1**	0.911	0.721	0.517	0.602
Architecture: **ViT-S**	Patch Size: **8**	Blend	224	0.552	0.966	0.702	0.657	0.460	0.541
			448	0.549	**1**	0.709	0.619	0.544	0.579
		Phase	224	0.748	**1**	0.856	0.736	0.513	0.605
			448	0.779	0.946	0.855	0.736	0.463	0.569
		Polar	224	0.701	0.948	0.806	0.688	0.345	0.460
			448	0.646	0.946	0.768	0.665	0.441	0.531
		Recta	224	0.697	0.991	0.819	0.708	0.477	0.570
			448	0.621	0.964	0.755	0.656	0.461	0.542
	Patch Size: **16**	Blend	224	0.820	0.983	0.894	0.744	0.447	0.558
			448	0.571	**1**	0.727	0.635	0.551	0.590
		Phase	224	0.821	0.991	0.898	0.719	0.508	0.595
			448	0.775	0.982	0.866	0.733	0.549	0.628
		Polar	224	0.826	0.983	0.898	0.735	0.384	0.504
			448	0.794	0.964	0.871	0.722	0.405	0.519
		Recta	224	0.797	0.983	0.880	0.735	0.481	0.582
			448	0.812	**1**	0.896	0.723	0.522	0.606

Table 5.1: **Ablation study of classification performance.** 'Architecture' is either ViT-Small (ViT-S) or ViT-Base (ViT-B). 'Patch size' are 8 and 16. 'Form.' is the data format, being one of four representations as given in eq. (3.1). 'Size' refers to the chunk size, either 224 or 448. Precision, recall and F_1-score are given for classification both with and without exclusion of ambiguous chunks in the dataset. Best result in each column is marked with bold.

Performance underestimation from label uncertainty. A significant source of label uncertainty in this study stems from arbitrary definitions used to determine whether a chunk should be labelled as "positive" or "negative".[4] Specifically, ground truth chunks are labelled "positive" only if a SHA event is within the central 50% of the chunk area. Any SHA event that falls outside this central region—yet still within the chunk boundaries—results in the entire chunk being labelled as "negative".

This annotation rule introduces a substantial risk of misrepresenting the true nature of the predictions. During evaluation without chunk exclusion, the model is penalized every time it correctly detects a SHA event that lies outside the central region of a chunk. In such cases, the prediction is counted as a false positive, despite the presence of a SHA event within the chunk itself. An example illustrating this misclassification scenario is provided in Figure 5.3.

Across all 32 network configurations, a total of 3,651 false positives and 10,886 false negatives are recorded. A closer inspection reveals that 2,602 of these false positives correspond to chunks that do contain a SHA event, albeit located outside the central 50% region used for ground truth labelling. This means that approximately 71% of all false positives in the evaluation can be attributed to ambiguous labelling, rather than genuine prediction errors.

While this figure highlights the considerable impact of label ambiguity on performance metrics, it is also worth noting that these mislabelled false positives account for about 18% of all incorrect predictions when both false positives and false negatives are considered together. Still, this proportion is not negligible and suggests that the evaluation procedure is likely underestimating the actual performance of the model.

Coherence dependency. A brief investigation into the relationship between network prediction and interferometric coherence was conducted. The average coherence across each chunk was computed and correlated with the network's prediction outcomes. This investigation focused specifically on the ViT-B/16 Phase/448 configuration, as it achieved the highest performance on the test dataset according to Table 5.1.

The results, illustrated in Figure 5.4, present the probability distribution of chunk-averaged coherence for each type of predictive outcome. The distributions reveal that false predictions are more frequent with chunks exhibiting lower coherence, while higher coherence is generally observed in correctly classified chunks.

[4] This uncertainty is described in greater detail in Section 3.3.

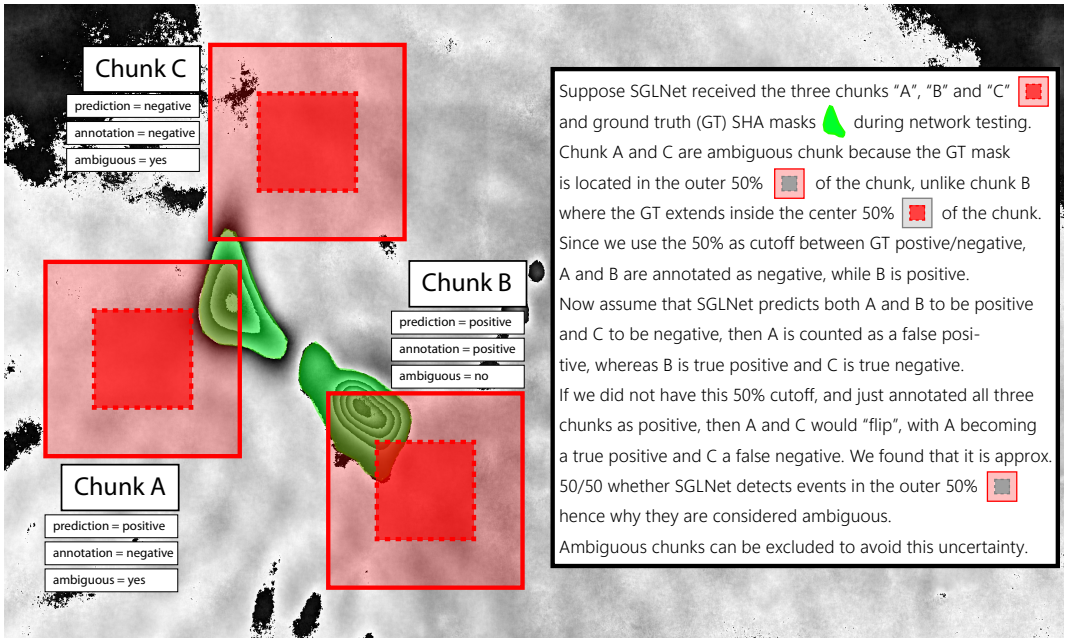


Figure 5.3: **Label ambiguity in chunk margins.** Chunks are annotated according to the location of the GT mask within the chunk. This gives rise to an uncertainty when the GT is located in the outer 50% of the chunk.

5.1.2 Event detection performance

While classification performance offers insight into the neural network's ability to label individual chunks, a broader evaluation is necessary to assess how well the full algorithm functions within the context of Project DynaMelt. This includes testing the algorithm's ability to generate boundary boxes that capture complete SHA events, as well as measuring the time required to process entire interferograms.

Figure 5.5 provides a visual example of output detections on ground truth labels. These ground truths are made by manually segmenting SHA patterns. Given the inherent subjectivity involved in manual annotation, all segmentations have been made publicly available in the SGLNet repository to allow for inspection.[5] In our evaluation, an event is only considered successfully detected if it is entirely enclosed within the output boundary box. Boxes that fail to contain at least one full SHA event are considered to be empty. We also measure the inference time both for individual chunks and for full interferograms. Results are summarized in Table 5.2. ViT-B/16 Phase/448 demonstrates the highest performance. It detects the most SHA events and produces the fewest empty boundary boxes. Smaller models are faster, but slightly worse at detection events.

[5] https://github.com/NielsPeterChristensen/SGLNet

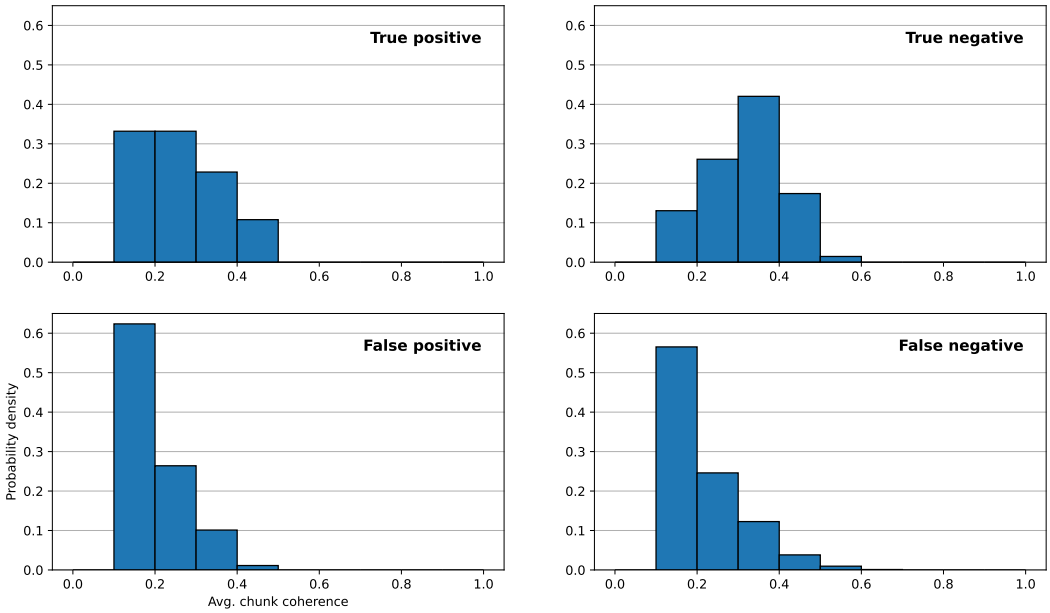


Figure 5.4: **Probability distribution of chunk-averaged coherence.** Network output is categorized according to incorrect (false) and correct (true) predictions of positive and negative labels. Network is configured as ViT-B/16 Phase/448 and uses the entire testing dataset.

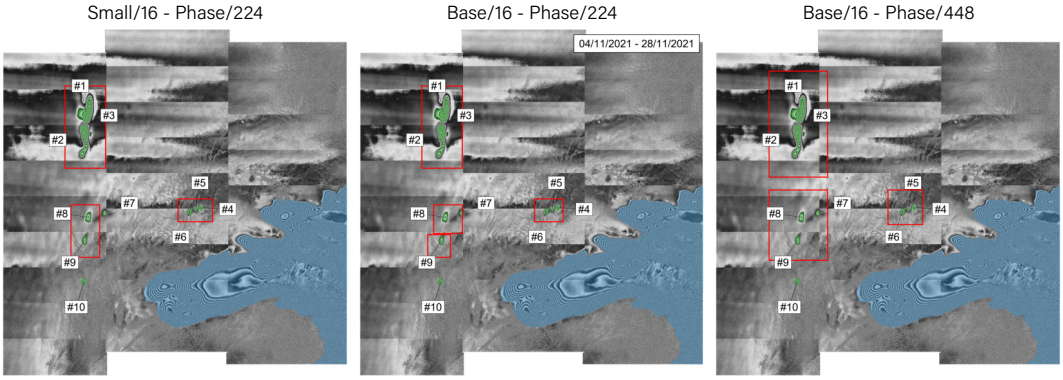


Figure 5.5: **Boundary boxes from SGLNet compared to ground truths.** Boxes are red, ground truths are green, while grounding lines are masked blue. All outputs use ViT-B/16 Phase/224. The interferogram is from Totten (T099A), formed with SLC data spanning 04/011/2021–28/1/2021.

		Form.	Size	#Detected	#Empty	Chunks/sec	sec/I_{dd}
Architecture: **ViT-B**	Patch Size: **8**	Blend	224	102 (53)	16 (24)	57	115
			448	122 (64)	24 (35)	7	244
		Phase	224	122 (64)	6 (10)	55	121
			448	124 (65)	8 (15)	6	289
		Polar	224	106 (55)	10 (16)	57	117
			448	119 (62)	9 (17)	6	275
		Recta	224	121 (63)	11 (17)	55	120
			448	125 (65)	10 (18)	7	245
	Patch Size: **16**	Blend	224	106 (55)	22 (31)	213	31
			448	127 (66)	15 (25)	45	37
		Phase	224	109 (57)	**5 (9)**	205	32
			448	**136 (71)**	**5 (9)**	44	38
		Polar	224	105 (55)	9 (15)	198	33
			448	125 (65)	12 (20)	43	39
		Recta	224	109 (57)	15 (22)	199	33
			448	122 (64)	7 (13)	43	39
Architecture: **ViT-S**	Patch Size: **8**	Blend	224	120 (63)	44 (44)	143	46
			448	128 (67)	23 (33)	15	111
		Phase	224	126 (66)	21 (26)	136	49
			448	116 (60)	9 (18)	15	112
		Polar	224	93 (48)	25 (35)	136	48
			448	110 (57)	14 (25)	15	115
		Recta	224	120 (63)	35 (39)	133	50
			448	116 (60)	21 (35)	15	112
	Patch Size: **16**	Blend	224	118 (61)	12 (17)	423	**16**
			448	126 (66)	17 (26)	92	18
		Phase	224	124 (65)	18 (23)	375	18
			448	123 (64)	13 (22)	80	21
		Polar	224	105 (55)	17 (24)	363	18
			448	94 (49)	7 (18)	81	21
		Recta	224	119 (62)	19 (24)	370	18
			448	124 (65)	9 (17)	83	20

Table 5.2: **Ablation study of event detection performance.** Measures are across a dataset consisting of 49 interferograms. '#Detected' is the number of individual SHA events fully detected, out of 192 total in the testing dataset. '#Empty' is the number of empty boundary boxes. 'Chunks/sec' measures the inference time of the network. 'sec/I_{dd}' is the time taken to run a full interferogram. Numbers in parenthesis gives percentages. Bold numbers mark best measure in each column (left out for chunks/sec).

5.1.3 Segmentation performance

The segmentation methods used in this work were introduced in Section 3.2, but are briefly recapped here for clarity:

- DINOseg: Self-attention thresholding.
- Similarity: Embedding similarity thresholding.
- Degree: Inverse of summed similarities.
- PCA: Thresholds first principle component of embeddings.

An example of each method is shown in Figure 5.6. Internally, these methods first generate a salient feature map, which is then thresholded to create a binary segmentation mask. The mask is fitted with a spline to extract outlines.

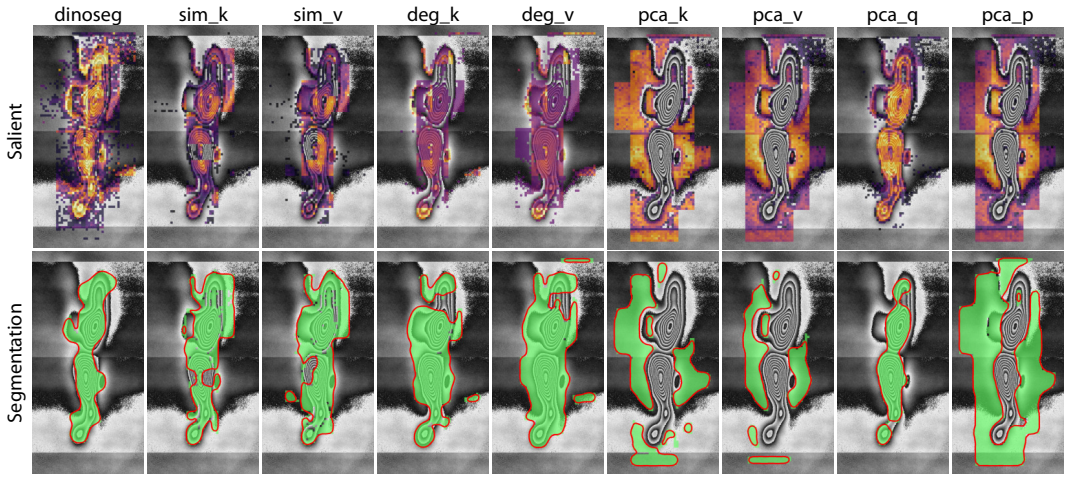


Figure 5.6: **Methods for semantic segmentation.** The first row shows saliency maps created by using a given method (DINOseg, sim, deg, pca) on embeddings: k(ey), q(uery), v(alue), p(atch). Second row shows segmentations (green) and corresponding outlines (red lines) generated by thresholding saliency maps. Each square represents a region on the ground of approx. 55 km × 64 km cropped out of the interferogram from Totten (T099A), formed with SLC data spanning 04/011/2021–28/1/2021.

Embeddings depend on the neural network, so with 32 configurations and nine different segmentation methods there are a total of 288 combinations to be tested. Each of these combinations was evaluated on the full test dataset of 49 interferograms. Segmentation performance was quantified using the global Dice coefficient (2.55) across the dataset, and the results are summarized in Table 5.3.

Overall, DINOseg consistently performs best across most network configurations. In a few instances, the Degree method does marginally better. The highest Dice score is achieved using DINOseg with the ViT-B/16 Polar/224 configuration, followed closely by ViT-B/16 Phase/224 and also ViT-B/16 Phase/448. In contrast, the methods based on similarity and PCA generally perform worse and, in some cases, fail entirely—particularly used with the ViT-S architecture.

					Similarity		Degree		First Principle Component			
		Form.	Size	DINOseg	Key	Value	Key	Value	Key	Value	Query	Patch
Architecture: **ViT-B**	Patch Size: **8**	Blend	224	0.274	0.122	0.217	0.162	**0.289**	0.122	0.217	0.044	0.194
			448	**0.250**	0.161	0.126	0.213	0.217	0.161	0.126	0.150	0.151
		Phase	224	0.354	0.315	0.118	0.347	**0.382**	0.315	0.118	0.168	0.278
			448	**0.338**	0.217	0.078	0.322	0.285	0.217	0.078	0.135	0.165
		Polar	224	0.315	0.092	0.086	0.192	**0.317**	0.092	0.086	0.117	0.228
			448	**0.309**	0.137	0.103	0.244	0.265	0.137	0.103	0.062	0.124
		Recta	224	**0.329**	0.108	0.223	0.226	0.321	0.108	0.223	0.109	0.242
			448	**0.293**	0.244	0.072	0.234	0.268	0.244	0.072	0.126	0.178
	Patch Size: **16**	Blend	224	**0.345**	0.206	0.237	0.182	0.261	0.206	0.237	0.208	0.197
			448	**0.318**	0.241	0.182	0.244	0.265	0.241	0.182	0.197	0.134
		Phase	224	**0.396**	0.174	0.198	0.337	0.289	0.174	0.198	0.213	0.262
			448	**0.388**	0.109	0.148	0.366	0.374	0.109	0.148	0.179	0.148
		Polar	224	**0.398**	0.086	0.164	0.228	0.270	0.086	0.164	0.209	0.246
			448	**0.380**	0.053	0.161	0.277	0.308	0.053	0.161	0.134	0.125
		Recta	224	**0.387**	0.073	0.282	0.239	0.274	0.073	0.282	0.318	0.198
			448	**0.362**	0.088	0.253	0.289	0.300	0.088	0.253	0.301	0.126
Architecture: **ViT-S**	Patch Size: **8**	Blend	224	**0.316**	0.001	0.088	0.124	0.295	0.001	0.088	0.053	0.185
			448	**0.258**	0.008	0.038	0.163	0.250	0.008	0.038	0.118	0.121
		Phase	224	**0.380**	0.018	0.115	0.286	0.358	0.018	0.115	0.018	0.264
			448	**0.300**	0.009	0.108	0.262	0.253	0.009	0.108	0.019	0.163
		Polar	224	**0.322**	0.006	0.148	0.191	0.314	0.006	0.148	0.022	0.205
			448	**0.282**	0.028	0.112	0.224	0.245	0.028	0.112	0.090	0.135
		Recta	224	**0.349**	0.001	0.196	0.198	0.304	0.001	0.196	0.015	0.167
			448	**0.274**	0	0.100	0.198	0.249	0	0.100	0.169	0.083
	Patch Size: **16**	Blend	224	**0.368**	0.162	0.027	0.184	0.349	0.162	0.027	0.006	0.232
			448	0.243	0.252	0.072	0.236	**0.293**	0.252	0.072	0.004	0.117
		Phase	224	**0.370**	0.155	0.042	0.337	0.367	0.155	0.042	0.010	0.229
			448	0.296	0.248	0.013	**0.320**	0.264	0.248	0.013	0	0.137
		Polar	224	**0.380**	0.156	0.040	0.250	0.339	0.156	0.040	0.017	0.231
			448	**0.277**	0.210	0.056	0.265	0.255	0.210	0.056	0.006	0.132
		Recta	224	**0.370**	0.214	0.063	0.264	0.340	0.214	0.063	0	0.193
			448	0.316	0.282	0.081	0.279	**0.317**	0.282	0.081	0	0.116

Table 5.3: **Ablation study of segmentation performance.** Each value is the global dice coefficient of the entire dataset consisting of 49 interferograms. 'Similarity', 'Degree', and 'First Principle Component' are methods that use embeddings 'key', 'value', 'query', or 'patch', while 'DINOseg' always uses the attention score. The best score in each row is marked with bold.

5.2 Additional SGLNet findings

5.2.1 Separation in feature space

The classifier network relies on the `[class]` tokens (feature vectors) to predict chunk labels. This only works if the backbone DINO ViT is able to sufficiently cluster feature vectors related to SHA events in the latent feature space. We investigate this by projecting the high-dimensional `[class]` tokens to a two-dimensional space. The projection follows the same procedure as described in the DINO preprint version [46], where the dimension is first reduced to 30 with PCA, and then to two by running t-SNE with a perplexity of 20 and learning rate of 200 for 5000 iterations. The analysis includes only a subset of 5000 chunks from the entire testing dataset. The resulting two-dimensional class embeddings are presented in Figure 5.7. We see that DINO has excellent separation between generic chunks on one side and SHA events + grounding lines on the other side. Meanwhile, DINO struggles to completely separate grounding lines and SHA events, though some clustering is noticeable.

A qualitative investigation of each cluster is included in Appendix B.2.

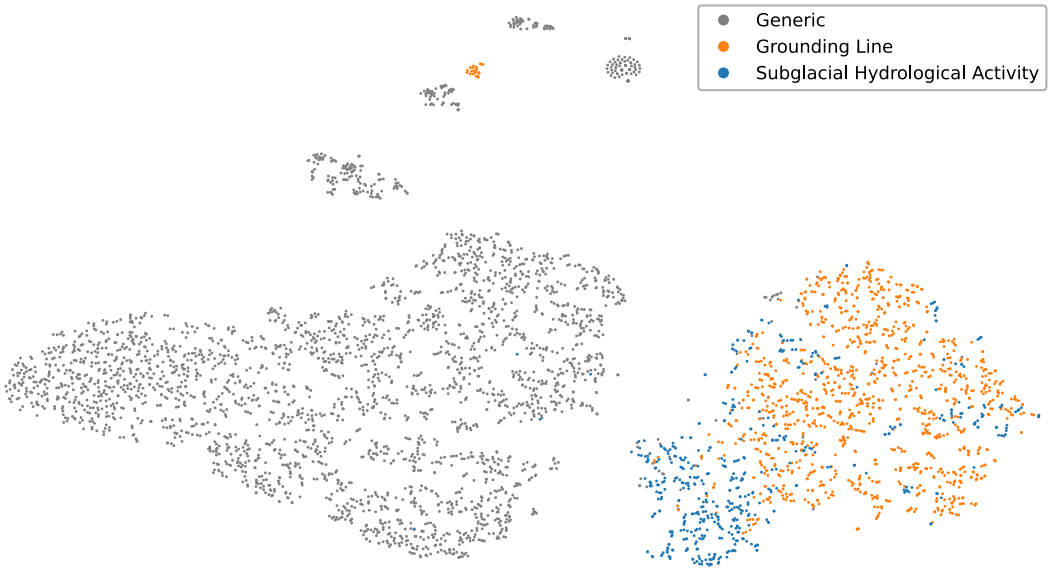


Figure 5.7: **DINO `[class]` embeddings projected to two dimensions.** The dimension is reduced from 1536 to 30 with PCA followed by t-SNE to 2D. A subset of 5000 feature vectors are includes, originating from SHA (blue), grounding lines (orange), and generic areas with no distinct spatial patterns in the interferogram (gray).

5.2.2 Alternative classification methods

Other methods can be used for classification instead of the linear projection head. Two methods have been tested for comparison. One uses simple cosine similarity, where each feature vector from DINO is compared to two subsets of labelled feature vectors. One subset contains only feature vectors from SHA events and the other does not. Each test feature vector is assigned to the set for which the average similarity is highest. Another approach applies labelled K-means clustering: a subset of labelled features are clustered using K-means, with each cluster receiving the label corresponding to the majority class within it. Test features are then assigned the label of the nearest cluster.

A comparison between the linear projection clsasifier and these alternative methods is presented in Table 5.4. While all methods show consistent and reasonable performance, the classifier head consistenly outperforms the others across all tested metrics.

	w/ chunk exclusion			w/o chunk exclusion		
Method	Precision	Recall	F_1-score	Precision	Recall	F_1-score
Linear classifier	**0.966**	**0.983**	**0.974**	**0.770**	**0.449**	**0.567**
Kmeans	0.826	**0.983**	0.898	0.769	0.419	0.542
Similarity	0.711	0.974	0.822	0.685	0.344	0.458

Table 5.4: **Performance with different classifier methods.** Metrics for classification on features from ViT-B/16 Phase/224 using a linear classifier head, K-means clustering, and cosine similarity. K-means uses 1000 clusters and a randomly selected subset of 10000 non-SHA training features combined with 689 SHA-features. Similarity uses a subset of 689 SHA-features and a randomly selected subset of 689 non-SHA features. Performance is computed on the whole test-dataset Best performance is marked in bold for each metric.

5.2.3 Alternative data composition

This section explored the impact of altering the data presented to SGLNet, as well as the effects of different data formatting choices.

Training without masking. As described in Section 3.1.2, certain chunks are excluded from training when their ground truth label is uncertain—this typically happens when SHA signals are faint, noisy, or not overlapping the central part of the chunk.[6] Instead of excluding these ambiguous chunks, an alternative approach includes them in the training data but labels them as

[6] Figure 3.10 shows which chunks are excluded during training.

"negative". Four different models have been training in this way. The classification performance of these are compared to equivalent model trained without (excluding) ambiguous chunks. Results are shown in Table 5.5. Precision, recall and F_1-score are reported in the same way as previously, with two testing scenarios: one where ambiguous chunks are excluded in the testing dataset, and one where they are not.

	w/ chunk exclusion			w/o chunk exclusion		
Model	Precision	Recall	F_1-score	Precision	Recall	F_1-score
Trained incl. amb. chunks						
ViT-B/8 Phase/224	**0.982**	0.931	0.956	0.850	0.323	0.468
ViT-B/8 Phase/448	0.980	0.857	0.914	0.846	0.388	0.532
ViT-B/16 Phase/224	0.954	0.888	0.920	**0.860**	0.293	0.437
ViT-B/16 Phase/448	0.980	0.875	0.925	0.856	0.420	0.563
Trained excl. amb. chunks						
ViT-B/8 Phase/224	0.927	**0.991**	0.958	0.800	0.487	0.605
ViT-B/8 Phase/448	0.815	0.946	0.876	0.772	0.505	0.611
ViT-B/16 Phase/224	0.966	0.983	**0.974**	0.770	0.449	0.567
ViT-B/16 Phase/448	0.902	0.982	0.940	0.771	**0.566**	**0.653**

Table 5.5: **Performance with and without ambiguous chunks.** The first four models are trained with a dataset where ambiguous chunks are included and labelled as "negative". The last four models are trained with a dataset where ambiguous chunks are excluded and thus never seen by the network. Precision, Recall and F_1-score are reported on the testing dataset both with and without chunk exclusion. Only four models are tested, all using ViT-Base with "Phase" format.

It is seen that the precision increases when ambiguous chunks are included in the training dataset. Meanwhile, recall and F_1-score does better when ambiguous chunks are excluded. The trade-off between precision and recall is unsurprising. When ambiguous chunks are included and simply marked as "negative", the network is effectively being trained to only classify the most certain SHA events, which reduces false positives and increases false negatives.

Blue channel for nan pixels. As interferograms are sampled into chunks, they have to be padded so that they are divisible by the size of the chunk sample window. Since chunks are converted to RGB images, the padding value cannot be NaN, but has to be a value between 0–255. SGLNet outputs show that the network occasionally mistakes chunks located on these padded edges to be SHA events, likely due to the apparent phase discontinuity that zero-padding introduces.

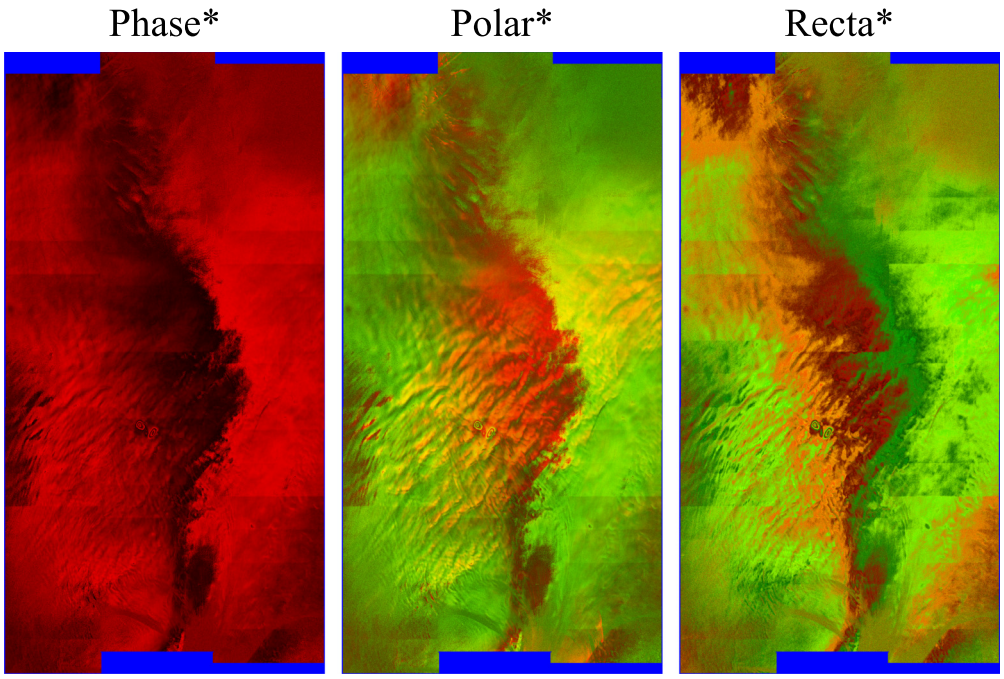


Figure 5.8: **Alternative RGB image representations.** All three types use the blue channel to mark NaN pixels.

We investigate whether network performance increases if the input RGB images contain information about NaN areas within each images. Referring back to the four RGB representations in Equation (3.1), it is seen that both "Polar" and "Recta" leaves the blue channel as zero, while "Phase" has the interferometric phase in all three channels. We change the blue channel to represent NaN values. Thus, it is 255 in the padded regions along the edges of each interferogram, and 0 elsewhere. An example is shown in Figure 5.8.

While the original representations "Phase", "Polar", and "Recta" all allow for the blue channel to be used, the fourth "Blend" format cannot be altered in this way, as it uses a pseudo-colour palette relying on all three colour channels.

Performance comparison is given in Table 5.6. Three networks have been trained and tested on data with this alternative RGB representation, and are compared against similar networks trained on data without NaN marked in this way. The comparison shows that including NaN information in the blue channel does not improve performance, but worsens it in almost all cases.

	w/ chunk exclusion			w/o chunk exclusion		
Model	Precision	Recall	F_1-score	Precision	Recall	F_1-score
NaN in blue channel						
ViT-B/16 Phase/448	0.794	0.964	0.871	0.713	0.539	0.614
ViT-B/16 Polar/448	0.736	0.946	0.828	0.698	0.485	0.573
ViT-B/16 Recta/448	0.812	**1**	0.896	0.725	0.546	0.623
no NaN channel						
ViT-B/16 Phase/448	**0.902**	0.982	**0.940**	**0.771**	**0.566**	**0.653**
ViT-B/16 Polar/448	0.794	0.964	0.871	0.746	0.507	0.604
ViT-B/16 Recta/448	0.836	**1**	0.911	0.721	0.517	0.602

Table 5.6: **Performance with and without NaN in blue channel.** The first three models are trained with NaN-pixels marked in the blue channel. The last three models have no information about NaN pixels. Precision, Recall and F_1-score are reported on the testing dataset both with and without chunk exclusion. All models uses ViT-Base architecture with patch size 16 and chunk size 448.

5.3 Investigating subglacial hydrology in Antarctica

This section describes use cases for SGLNet in subglacial hydrological research in Antarctica. The first half uses the Geophysical Inversion Module (GIM, Section 3.1.1), where we show examples of measured surface deformation from detected events at Totten Glacier and Vanderford. The latter part shows the spatial distribution of detected events from three regions in Antarctica.

5.3.1 Observed surface deformation

The primary goal of SGLNet is to provide coordinates of SHA events. The corresponding regions of the DD interferograms can be unwrapped by the GIM. The unwrapped phase can then be related to surface displacement along the radar's LoS using (2.8). Two examples of detected events and their GIM-processed displacement are included in Figure 5.9.

The first row is the same interferogram as used in previous figures, showing a very distinct displacement pattern in the upper left corner. Notice how this displacement pattern consists of surface uplift (red) closest to the grounding line, followed by subsidence in the upstream direction. This is a very common response, in agreement with Figure 2.3c. SGLNet detects these event perfectly and the GIM is able to translate them into geophysical displacement.

The second row shows a less than ideal noisy interferogram. While SGLNet is able to capture most of the characteristic displacement patterns, the GIM is unable to unwrap these events. This illustrates how phase noise and small displacements can make downstream processing difficult, even when SGLNet provides valid detections.

The results shown in Figure 5.9 are intermediate GIM outputs. The final GIM outputs are geocoded images of the unwrapped phase, which can be converted to vertical displacement using (2.9). This is shown in Figure 5.10 for the prominent event in the—at this point—familiar Totten interferogram. The drainage appear to flow along the modelled subglacial water channels towards the margin.

Time series analysis One of the many advantages of S1 satellite data is the relatively high temporal resolution, allowing time series to be formed. This is not the focus of this study, so the matter is kept brief and used to illustrate how SGLNet can be used in this regard.

As part of Project DynaMelt, a series of uplift and subsidence events from Vanderford Glacier (neighbouring Totten) are observed. They have been geocoded and converted to vertical displacement and are shown side-by-side in Figure 5.11. The SHA events propagating along a 60 km path towards the grounding line. Compared to the modelled hydrological potential along the transect, it would appear that water travels against the gradient at the midpoint.

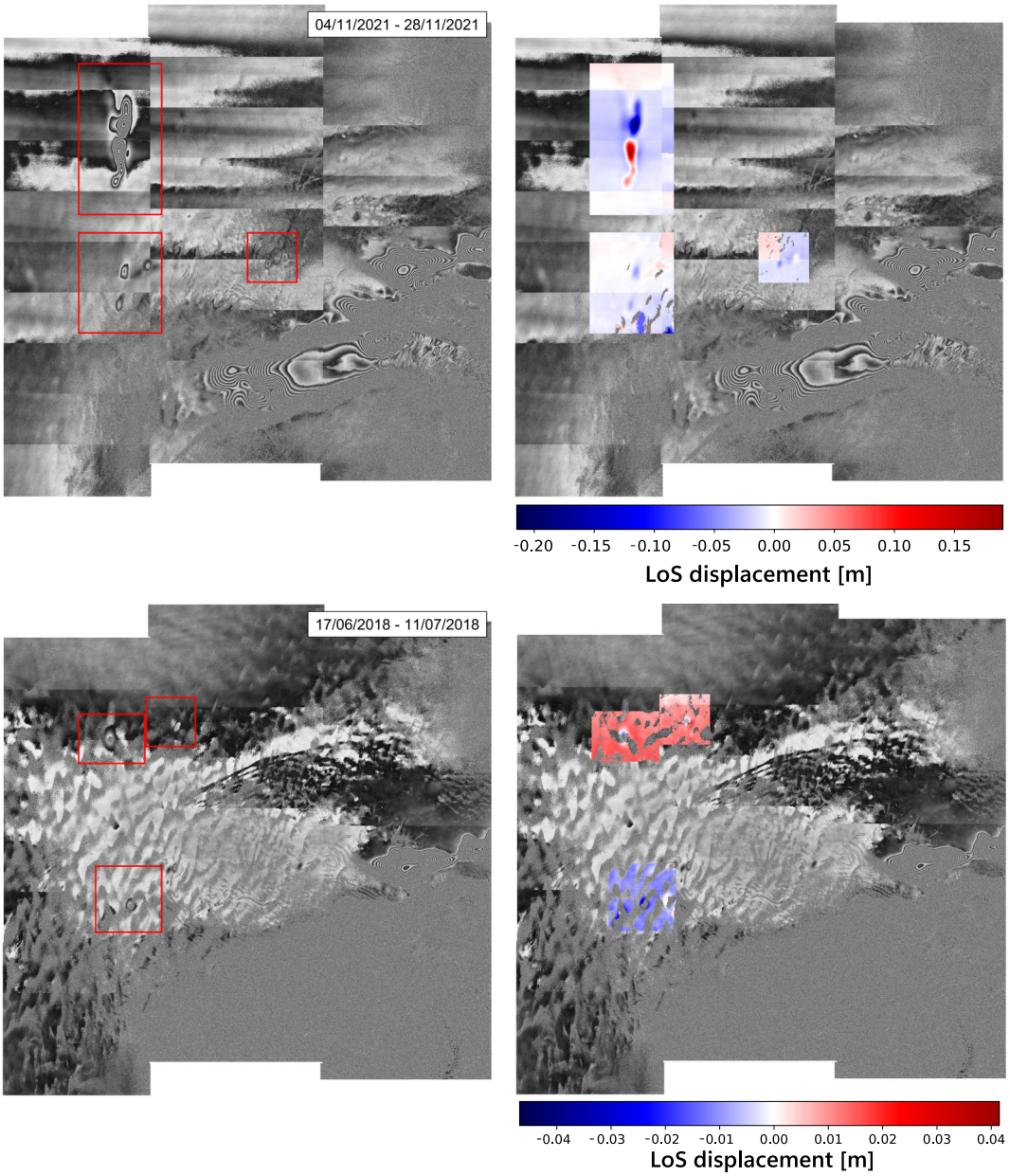


Figure 5.9: **Event detections and LoS displacement.** Left column shows event detections from SGLNet. Right columns are the corresponding LoS displacements derived with the GIM module and imposed onto the original interferograms. Both rows are from S1 track 099 at Totten Glacier. The top row is acquired from 04/11/2021–28/11/2021 and shows a large, coherent phase pattern in the upper left side, together with some smaller detected patterns. The bottom row is acquired from 17/06/2018–11/07/2018 and shows a much more decorrelated interferogram. The phase noise makes GIM unwrapping practically impossible.

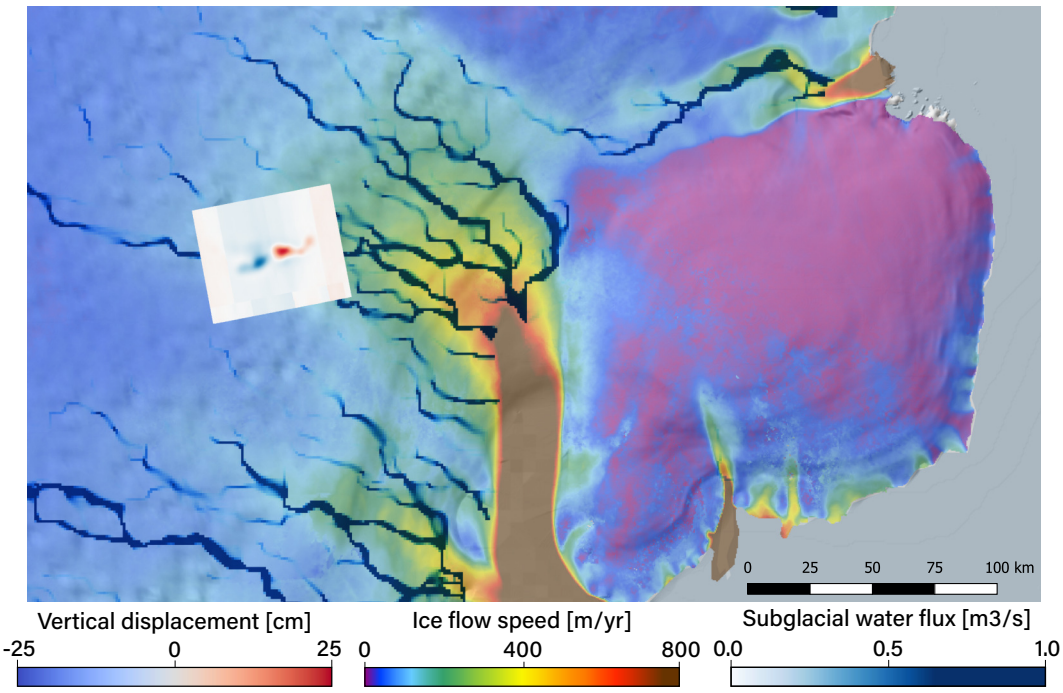


Figure 5.10: **Geocoded vertical displacement.** From track 099 interferogram covering Totten Glacier from 04/11/2021–28/11/2021. The event is plottet together with ice flow velocity [42] and predicted subglacial water flux [43] using Quantarctica [21].

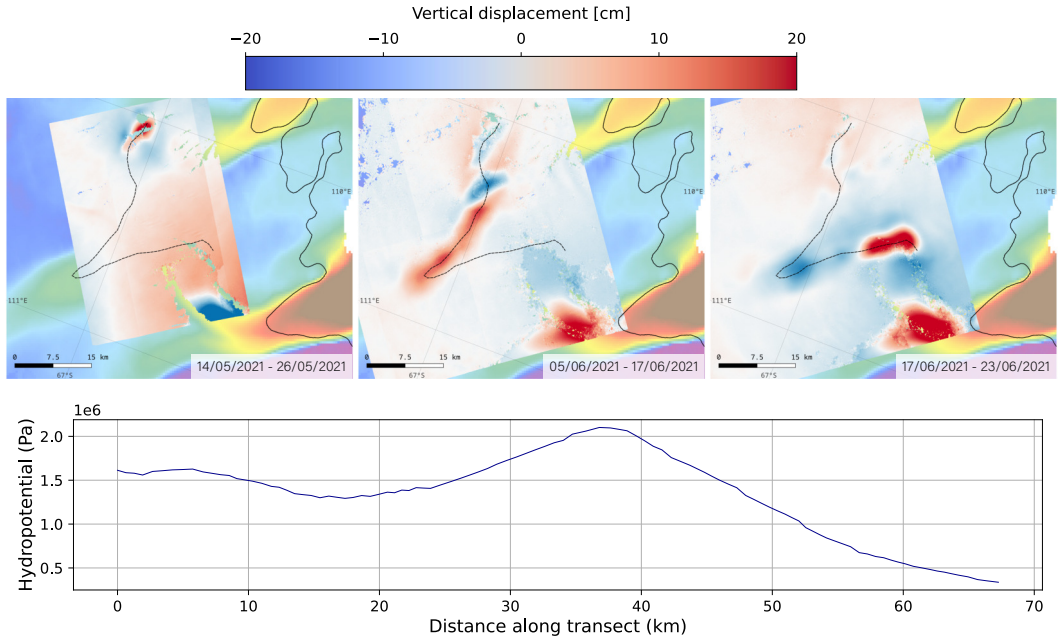


Figure 5.11: **Subglacial drainage pathway compared to hydropotential.** GIM products of vertical displacement are from 14/05/2021–26/05/2021, 05/06/2021–17/06/2021 and 17/06/2021–23/06/2021 at Vanderford Glacier. Transect of the predicted hydropotential [6] along the 60 km path of propagation is shown at the bottom. The figure material is part of Project DynaMelt.

5.3.2 Spatial distribution of drainage events

Project DynaMelt has used SGLNet to automate detection of candidate drainage events over extended regions of Antarctica. Three regions collectively covering more than 10^6 km^2 was investigated. These regions are approximately located in the territories: Ellsworth Land (Region 1), Wilkes Land (Region 2), and Enderby Land (Region 3). Analysis amounts to 3243 double-difference interferograms created from 9705 S1 SLC products. The track numbers are included in Appendix C. To streamline the processing pipeline, the attention segmentation outputs where used to produce outlines of SHA events. Culling was necessary to remove false positives from the SGLNet output. More than 2000 events where automatically detected and segmented, and have been verified manually afterwards. Figure 5.12 shows the result. These same maps are shown in a larger format in Appendix C. Events trace out pathways towards the margin, and mostly overlap areas of increased ice flow speed and predicted subglacial water flux. There are also many events where this is not the case.

Drainage events and bed topography. The spatial distribution of events gives insight into the propagation of water underneath the ice. Compared to the ice flow speed and bed topography at Evans Ice Stream (see Figure 5.13), events are observed along topographic depressions, where ice flow speed increases. Most events are observed upstream with less observations near the outlet. Referring to Figure 5.12 (middle left), this is possibly be due to reduced coherence making detection difficult.

Detection agreement Project DynaMelt has investigated whether the locations of detected vertical surface displacements are in agreement, both internally (in cases where multiple satellite tracks cover the same area) and against external sources. It is found that events are generally detected in the same location across overlapping interferograms from different tracks. Some variability is to be expected, though, as these tracks only have partial temporal overlap and coherence may also differ.

Detections also coincide with active SGL identified in previous studies. The tracks used in Project DynaMelt only covers SGLs observed with space-born instruments (either SAR or altimeters). These are shown as green triangles in Figure 5.14 (left), which shows a small section of Totten Glacier. The same figure also shows a large surface uplift captured by DEMs derived from optical satellite images. A transect through the 10 km wide anomaly shows an uplift of 50 m over a three-year period. Events are detected in this exact location. Bear in mind that DDInSAR observes changes in deformation rate, so events are only detectable if this long term uplift does not occur at a fixed rate.

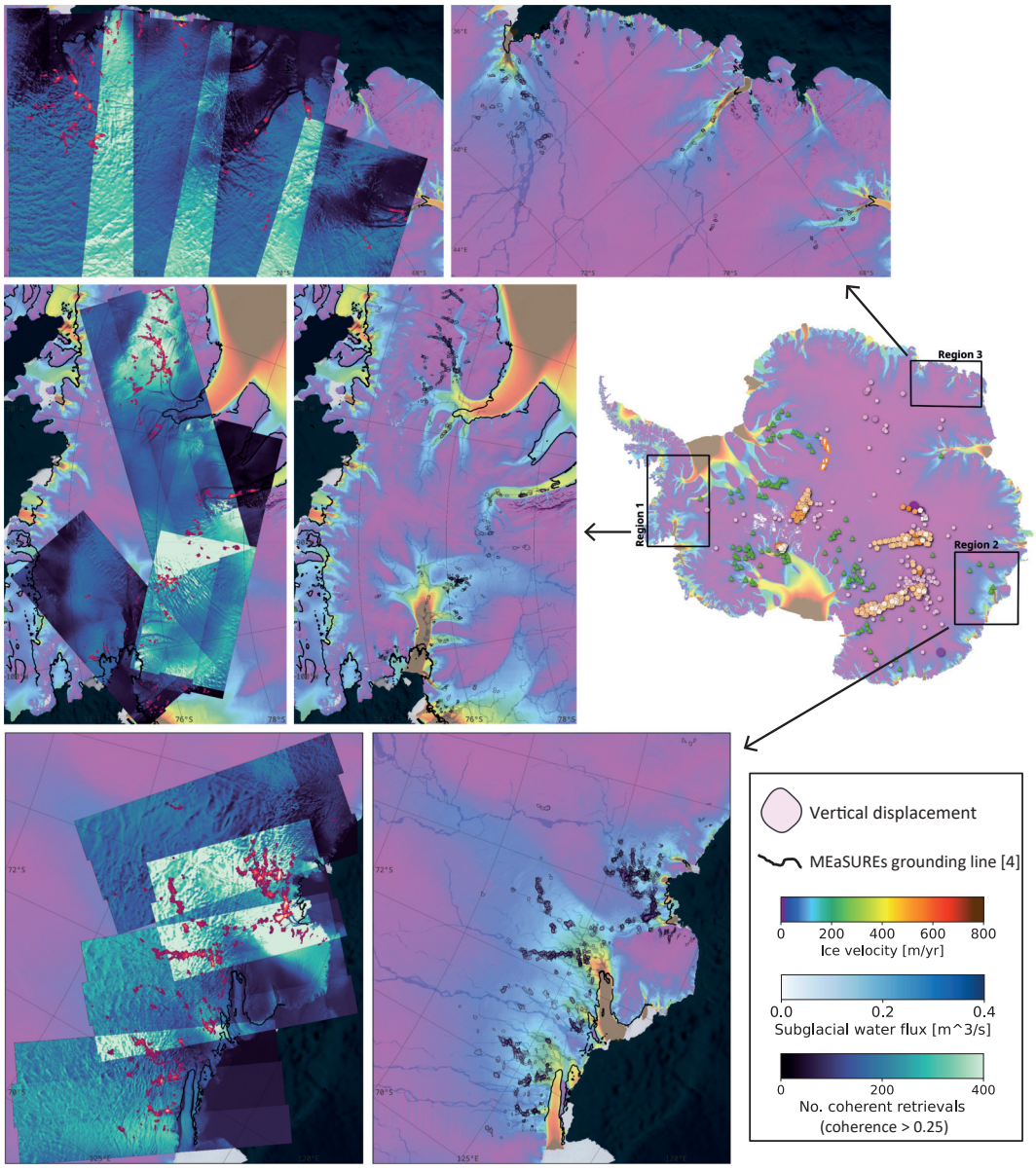


Figure 5.12: **Location of vertical displacements detected with SGLNet.** Three regions are investigated. The left-most maps show event heatmaps plotted atop a coherence count from S1 products. These are accompanied by maps show individual outlines of events. Background maps of ice flow velocity [42] and subglacial water flux [43] are shown together with grounding lines [47] and previously identified subglacial lakes [48]. The figure material is part of Project DynaMelt.

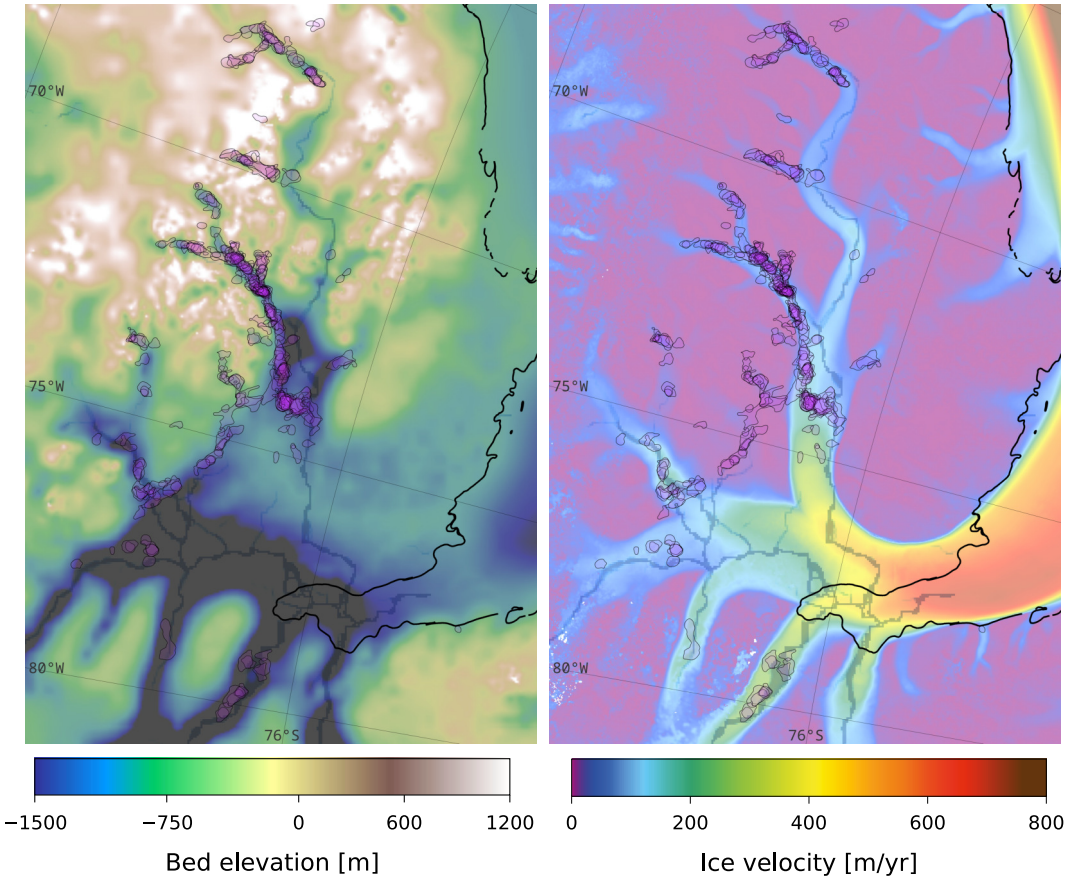


Figure 5.13: **SHA events, bed topography, and ice flow speed at Evans Ice Stream.** Individual events are shown as purple spots. Bed topography is from [49] and ice velocities are from [42]. The figure material is part of Project DynaMelt.

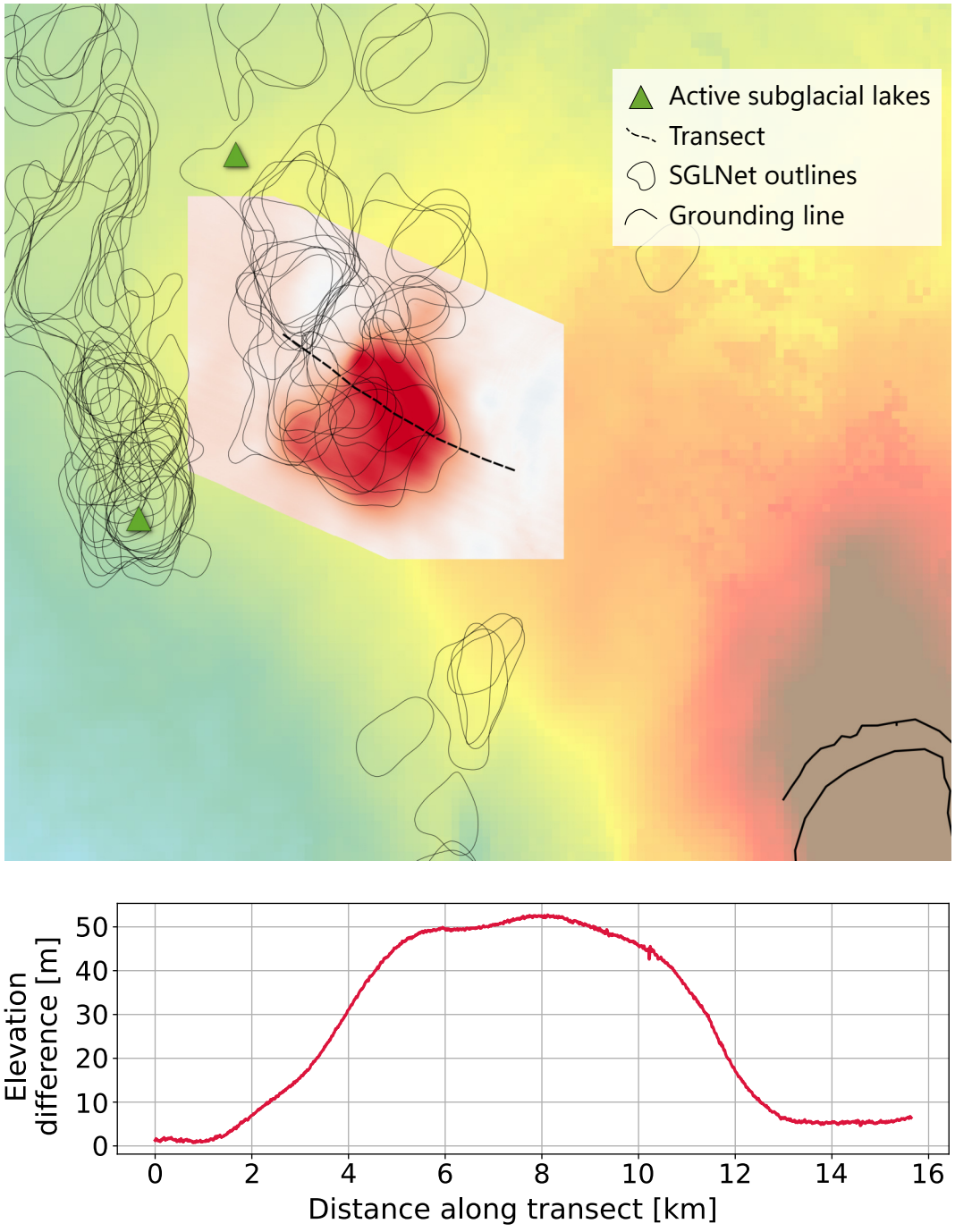


Figure 5.14: **Detected events compared to existing SGL inventory and DEMs.** (left) SGLNet event outlines near grounding line at Totten Glacier. Green triangles are active SGLs from previous studies [48], while the difference between REMA DEM [35] from 17/10/2020 and 03/10/2017 is shown in the middle. Totten grounding zone [47] are in the bottom right, and the background colour indicates ice flow velocity [42]. (right) Elevation difference between the two REMA DEMs from transect shown on the left. The figure material is part of Project DynaMelt.

Chapter 6

Discussion

This chapter includes discussion of key findings, with interpretations and implications of the study, together with comparisons to other works. Limitations and future directions are also presented.

6.1 Summary of key findings

Classification performance. The highest classification performance, measured by F_1-score, was achieved using a DINO backbone with the ViT-B/16 architecture and the "Phase" RGB representation. The optimal chunk size—224 or 448—depends on the testing setup. As expected, ViT-B outperforms ViT-S, consistent with findings from the original DINO study [31]. Interestingly, patch size 16 outperformed patch size 8, despite smaller patches providing higher spatial resolution. This is likely because smaller patches make the network overly sensitive to irrelevant fine-scale noise, while SHA patterns typically span hundreds of pixels.

Event detection performance. We found the best event detection performance was also attained by ViT-B/16 with "Phase" representation and chunk size 448. This aligns with classification results since event detection depends strongly on the accuracy of chunk classification. Avoiding patch size 8 not only improved detection accuracy but also reduced processing time per interferogram. Large chunks (448) outperformed smaller ones (224), probably because a larger receptive field captures more contextual information. Additionally, event detection tolerated excessively large detection areas as long as SHA events are being captured, which favours bigger sample windows.

We noticed that most missed detections occurred in decorrelated regions. We did not place any lower limit on coherence for events included in the ground truth annotation, because we sought to avoid arbitrary biases. It is worth noting that many of these noisy events cannot even be unwrapped by the GIM (as shown in Figure 5.9), making them unusable for Project DynaMelt specifically.

Segmentation performance. Semantic segmentation based on the network embeddings showed the potential for event segmentation using ViTs. This approach is contrary to CNN segmentation used in related works [23], [24]. We found that thresholding the DINO attention scores (DINOseg) achieved best performance as measured by the Dice coefficient. Prior studies also highlight DINO's attention maps as particularly well suited for semantic segmentation [31], [38], [40]. Despite DINO being trained for classification and not segmentation of interferometric phase patterns, it still shows some capability to segment SHA events. Due to time constraints, tuning of segmentation parameters was not explored, leaving room for future improvements.

Validation of SGLNet detections. We found many examples where event outlines agree between satellite tracks covering the same area. The algorithm

also marked the same locations as identified in previous studies at Totten, Rutford, [48], [50], and Jutulstraumen [8] using InSAR and altimetry. Several detections were also found to coincide with a large long-term REMA DEM elevation difference. These observations are good indicators for the robustness of the model. Detecting the same events from different tracks and in the same locations as previous DDInSAR studies shows that outputs are replicable, results from actual displacements on the surface and not just e.g ionospheric interference. Finding agreement between detections and vertical displacements seen in different types of data (either altimetry or DEMs derived from optical images) signifies some level of reproducibility. Additionally, the SGLNet algorithm is completely deterministic; repeating the same exact input always yield the same outcome. However, changing the fringe pattern (e.g. by applying a small constant phase shift across the entire interfeogram) may alter predictions slightly.

Together, these are good implications for the validity of SGLNet detections. That being said, false detections are present in the output. The network is particularly sensitive to errors where ionospheric effects or glacier surges create fringe patterns similar to those associated with SHA-induced vertical surface displacement.

Lastly, we have not been able to relate the network output to direct measurements of subglacial hydrology. An earlier study by Andrew P. Wright et al. [50] made RES measurements at Totten Glacier as part of a larger airborne campaign covering the Auroral Subglacial Basin. They only find "substantial lakes" (defined as water depth ≥ 8 m over a horizontal distance ≥ 500 m) inlands, close to Dome C. Near Totten Glacier, they identify a significant number of lake-like features that are relatively bright compared to their surroundings and are hydraulically flat, but fail to identify as lakes because the surface appear "fuzzy"—either with non-specular reflection or a contorted interface. They conclude that warm wet basal ice is present, and predict a substantial water flux in Totten's subglacial catchment, and suggest that these lake-like features are worth investigating further. They also remark that uncertainty in the absolute bed reflection coefficient is likely larger than the signal due to the presence/absence of subglacial water in areas where ice temperature and impurity are unknown (which require field measurements such as deep ice cores), and that their cross-track (east-west) resolution is insufficient to determine whether hollows exist to allow ponding. We find the conclusions by Wright et al. to be compelling in relation to our study. Even though they does not find definitive subglacial lakes, they do predict a large and active subglacial system at Totten Glacier, with many indications of lake-like activity, similar to the activity we find.

Network alterations. In addition to the 32 configurations included in the primary setup that is tested through classification, detection and segmentation, we also did other alterations of both the network and the training data.

First, we found that a linear classifier produces better predictions than Kmeans- or similarity-based classification methods. The authors behind DINO [31] draw the same conclusion, after tested DINO with a linear classifier versus a k nearest neighbour classification. As the linear classifier is trained on labelled data, it received additional information that these other methods does not. With sufficient training and an architecture that allows it to leverage this knowledge, it can thus improve performance on specific tasks. SGLNet uses a rather simple classifier (a shallow MLP) and yet it is able to distinguish between the feature

Second, we found that the classifier network can be trained either including or excluding those chunks where labelling is difficult/ambiguous, though best performance is achieved when these chunks are excluded from the training data. When manually labelling training interferograms, it takes quite a while to not only mark "positive" regions, but also identify any area that should be excluded. These ambiguous areas are faint or covered in noise so interferograms must be studied in detail to be noticed. Putting together a labelled dataset would therefore be substantially faster if ambiguous labels can be avoided. Thus, there is a trade-off here between fewer, but detailed annotated data versus more data with some label uncertainty. This may be considered in future works.

Third, we found that performance decreased when we included information about NaN-pixels in the input to the ViT, by marking these in the blue colour-channel. Another option we did not pursue is adding interferometric coherence in the ViT input. In the work by Tarekere et al. [24], they implemented CNN delineation of grounding lines in S1 DDInSAR data. They tested different input combinations, using both interferometric (real/imaginary part, coherence, and phase) and non-interferometric (DEM, northing and easting ice velocities, and differential tide levels). They achieved best performance when inputs included real and imaginary (rectangular) interferometric components and all four non-interferometric components. Nevertheless, they recommend using just the rectangular components and nothing else, as they found the model to become over-reliant on non-interferometric components. Our algorithm only uses interferometric components, similar to the recommendation from Tarekere et al. though we obtained better results with only the phase as input, not rectangular components.

6.2 Interpretation and implications of study

Foundational models for remote sensing. When we projected a subset of extracted features to a two-dimensional space, we found a very good separation between generic chunks on the one hand, and chunks containing SHA event and grounding lines on the other hand. We would like to emphasize this finding. In and of itself, this separation in feature space is what enables the classifier to make valid predictions. On a broader scale, it emphasizes just how powerful DINO is (which likely apply to other foundational models as well). DINO was never trained on interferometric phase images, nor any other type of remote sensing data. It is trained on the ImageNet-1k dataset containing small images of houses, animals, food, and vehicles etc., and yet it is able to cluster features from DDInSAR data in a meaningful way. There is even somewhat of a distinction (sub-clustering) between SHA events and grounding lines, though it is less pronounced. This ability owes to how ViTs are capable of learning complex pattern recognition and generalization capabilities, if trained on sufficiently large datasets, which is exactly the case for foundational models.

Pretrained FMs have another advantage: they are not learning from scratch. We did not have a huge database of labelled interferograms available, and "only" had time to construct a dataset from the 398 interferograms used in this project. While it might seem like a descent amount, especially considering that these interferograms are split into 1.4M chunks combined (for chunk size 224), but less than 800 of these chunks contain the type of signals we seek to identify. This is likely not enough data to train a randomly initialized model (not even a CNN). Real data is noisy and unpredictable, so the model would need to see a large variety of scenarios in order to generalize sufficiently. A pretrained DINO backbone allowed us to exploit the knowledge already learned by the model. We did not have to teach it the complexities and contextual patterns within images—that is part of the DINO pretraining—we only had to train a network to use DINO features for our task.

In summary, we ascribe the overall success of SGLNet to these attributes of DINO as a foundational model; that it is able to distinguish relevant spatial patterns in interferograms, and does it so well that our classifier network can be kept simple and trained on a relatively limited amount of labelled data. This observation pairs well with the finding in [31], [39] that DINO and it's successor DINOv2 are reliably able to outperform supervised models and convolution neural networks for a lot of classification tasks. We argue that foundational models have incredible potential within the field of remote sensing data science, especially when it may not be feasible to train a supervised network, or complexities make classical models cumbersome.

Glaciological findings from SGLNet. The aim of this study has been to develop and document a ML algorithm. Nevertheless, we have been able to present some early findings that Project DynaMelt have derived using SGLNet, but keep in mind that the method and derived findings are in their infancy.

We observe a significant number of vertical surface displacements at the glacial margins, typically moving downstream towards grounding lines along subglacial pathways that tend to follow bed topography, ice velocity, hydropotential, and modelled water flux. We also found events where this was not the case or even the opposite, with pathways seemingly moving against the hydropotential gradient. With some further analysis, these findings could be used to validate models of the hydropotential and bed topography. It may also be possible to identify persistent lakes, in locations where multiple events are detected over extended periods of time, such as the comparison with the REMA DEM suggests.

Looking more broadly, we see strong indications of a highly dynamic subglacial system. The multitude of events we observe along predicted subglacial waterways only appear in the DDInSAR products, because the subglacial system is undergoing change, when pathways form/collapse or expand/retract in response to variations in the effective subglacial pressure. This is in contrast to the steady-state assumption that subglacial hydrology models may use. Given the significance of subglacial meltwater on ice dynamics and ice sheet stability, combined with uncertainty in current models, we find our observations derived from SGLNet to be of interest to the field of glaciology. Our approach provides an important source of information about the subglacial system, as it opens the door to the entire back-catalogue of S1 data, of which only a "small" portion have been processed to obtain the results presented in this study.

6.3 Limitations

We have identified some challenges and limitations with the current implementation of SGLNet, particularly relating to training and testing, but also the experimental segmentation.

Class ambiguity. We did not initially expect class ambiguity—i.e. the problem of assigning binary labels to chunks that may equally be classified as "positive" or "negative"—to be as big a challenge as it turned out to be. Our first and second iteration of annotations for training did not take this ambiguity into account. Any SHA event that could be identified in the training data was labelled "positive". This includes events where the fringe pattern was highly decorrelated, very faint or any chunk that included even just a single pixel of an event. This resulted in very ill-behaved models. They were generally able

to identify SHA events, but also had a huge number of false positives, particularly in very noisy parts of interferograms. It was clear that the network struggled to classify these extreme cases, with labels essentially giving "mixed signals" during training. Our third iteration therefore combined binary class labels with ambiguity masks to exclude any chunk where the true class was ambiguous. This drastically reduced the number of false positives, and showed the importance of class-certainty for training.

Class ambiguity reflects our inability to correctly label chunks under certain conditions. The errors they induce are not inherent to the data itself or the algorithm, but due to superficial imposition of binary class labels on data that comprise continuous values. Therefore, we where able to improve the model performance by curating the training dataset. Unfortunately, this makes network testing difficult. If we measure performance on all chunks (incl. ambiguous), the resulting metrics does not reflect model performance in isolation, but also our class uncertainty that is not inherent to the model. In this way, we end up misrepresenting the model performance negatively. Inversely, if we curate the testing data similarly to the training data (by excluding ambiguous chunks), then we alter the statistics of the dataset. We reduce the influence of our class uncertainty, but the dataset becomes artificial (not reflecting true nature). This will likely misrepresent model performance positively. It is for this reason that we chose to report performance in both scenarios, as we would expect the true performance to be somewhere in-between. This allows us to compare the relative performance between different models, but we lack a single fixed metric.

Event segmentation. The primary goal of this study was not to directly segment events. Instead, we aimed a detecting events and marking their coordinates. Events could then be unwrapped by the GIM and segmented afterwards (which we did not have time to implement). This approach has some benefits over ML segmentation. First, it enabled us to use the DINO ViT, which did not require extensive training. Second, it would potentially be possible to use classical segmentation methods on unwrapped interferograms. These can more easily retain the physical interpretability of segmentations, e.g. using a threshold segmentation of the vertical uplift/subsidence.

Unfortunately, phase noise is very common in the 6- and 12-day interferograms from Sentinel-1. The GIM is not always able to unwrap these sufficiently. Either the unwrapped phase is noisy, or extended regions are left completely unwrapped, both of which makes classical segmentation difficult. Thus, we tried to bypass the GIM by segmenting the attention-embeddings directly.

Attention segmentation is not true ML segmentation; no decoder is included in the model architecture and the model is not trained for this purpose. It is

rather an opportunistic use of the attention mechanism. While we were able to produce approximate outlines, neither of our nine segmentation implementations are ideal. They rely on assumptions that cannot strictly be justified. The methods adopted from LOST [40] assume the foreground object (SHA events) to take up less than 50% of the image. The method that thresholds the first PC's assumes that foreground objects are the primary source of variance in the embedding space, so that the first PC is aligned with the foreground-background separation. The attention score thresholding introduced by [31] assumes that the foreground object is located where the top 60% of attention mass is found. These might be considered more or less useful proxies of the foreground object, but they rely on the ViT encoder and are not tied directly to the interferometric input data. They are limited to the native patch resolution, and while it is possible to upsample features in the embedding space [51], we still lack meaningful information in noisy regions.

6.4 Future directions

In this section we discuss future directions for this study. This includes improvements to the existing framework and entirely new frameworks.

Avoiding class label ambiguity. It would be very beneficial to solve the issue of ambiguous class labels. A big part of the problem is the strictly enforced binary labels. Soft labels may be used instead. A set of discrete labels—e.g. $\{0, 0.25, 0.5, 0.75, 1.0\}$—would represent our level of certainty that the ground truth should be labelled as a SHA event or not. During training, the model is then forced more severely by events that we are more certain of. Manually making differentiated labels for each interferogram would unfortunately be even more time-consuming and still include a human bias, though our uncertainty would also be explicit.

We may also find inspiration in other works employing variations of soft labels. In [52] a method for label smoothing is introduced, while [53] used a type of weak supervision referred to as positive-unlabelled learning: a method where some other measure is used as a proxy for confidence in the positive labels. These may provide useful insights or inspirations and should be included in future literature reviews.

Data alterations. We have identified four data-related changes that should be investigated further.

We never implemented data augmentation (e.g. flipping chunks), mainly because we employ a frozen backbone, so augmentations would not affect the

quality of extracted features. It may still be useful to include augmentations, however, as it would increase the quantity and diversity of positive labels in the training data.

We would also like to include interferograms with severe ionospheric effects and ice surges in the training data. We have found that the algorithm is particularly prone to false positives when the ionosphere introduces phase fringes in interferograms. These phenomena are not represented in the training data. If they were, it may reduce the number of false positives.

In addition to the above, we would like to increase the number of SHA events in the training data in general. This would increase the data diversity and potentially allow the model to better generalize. This is made easier by the fact that we have identified more than 2000 events during first algorithm rollout. These new events may be used to form a more extensive training dataset.

Lastly, we did not test whether it would improve model performance to include coherence as part of the ViT input (e.g. in the blue channel). It would go against our previous observations, where similar input alterations reduced performance, but it should nevertheless be tested.

Unfreezing DINO. We have only used DINO with frozen weights. By unfreezing the pretrained weights and finetuning DINO on actual DD phase interferograms, it would likely become better at identifying and extracting features that are specific to interferometric data. This finetuning process should leverage data augmentations to enforce invariances to e.g. phase shifts, phase inversion, image flip and rotation, noise jitter, and NaN-values.

While finetuning requires less training compared to pre-training, DINO is still a fairly large model. According to Caron et al. [46]l pre-training took three days on two 8-GPU machines with 16GB memory in each GPU. Finetuning may also require clustered GPU computing. This can be facilitated on the DTU HPC, but would require some alterations to the algorithm. The increased queue waiting time on the HPC should also be considered.

Other foundational models. A successor to DINO is DINOv2 by Oquab et al. [39]. They curate a much larger training dataset of 142M images, compared to 1.2M for DINO. They train four ViT models: a small (S), base (B), large (L) and giant (G). The ViT-G has an embedding dimension of 1536 with 24 attention heads, giving a total parameter count of 1.1B. This newer version is computationally demanding, but also shows improvements over other models (including DINO) when measured on common classification benchmarks [39]. This model may be even better at separating interferometric features. It would be interesting to test whether it can tell e.g. SHA events and ground-

ing lines apart. However, using a larger model requires more compute and may not improve results significantly. Furthermore, it is found that DINOv2 is worse for semantic segmentation than DINO, if the model is not altered to include extra "register"-tokens [38]. Lastly, these extremely large FMs are not environmentally friendly. According to Oquab et al [39], the DINOv2-G model pre-training took 22k GPU-hours and emitted 3.7 tons of CO_2eq, while finetuning takes around 1k GPU-hours and emits 0.2 tons. Inference requires substantially less compute than training, but still require a large number of computations. If a larger model only provide marginal improvements over a much smaller model, then the smaller model should be used for efficiency.

Many new geospatial FMs for Earth observation (EO) applications have been published in recent years, with more on the way. They are trained on satellite data from different sensors, with different modalities, for different tasks. Some even aim at being sensor-agnostic and multi-model with a variety of output options. We would include the following FMs in a future literary review: CROMA [54], DOFA [55], SSL4EO [56], Panopticon [57], TerraMind [58], SMARTIES [59], SkySense [60], and Galileo [61]. All these FMs are trained on Sentinel-1 together with some combination of other sensor data. Some are aimed at classification, segmentation, change detection, etc. Models trained on satellite data may be better adapted to our use case. However, none of these models are trained on (DD)InSAR data, and most of them are still under development. In their current state, these models would likely not improve our performance significantly. Panopticon struggles to outperform DINOv2 on EO benchmarks [57], while DINO is able to compete with another model called FM4CS [62]. Unfortunately, the other models do not benchmark against regular non-EO FMs like DINO. Geospatial FMs will certainly improve in the coming years and should definitely be considered for EO applications.

Segmentation-specific improvements. As previously mentioned, it was originally the goal to detect events with SGLNet, unwrap interferograms with the GIM, and then segment the unwrapped events, though this was never implemented. We think an active contour method such as Chan-Vese should be tested. The snake would be initialized at the location of the attention-segmentation contours. As the contour is regularized, behaviour can be tuned to reduce the influence of noise etc. It may even be possible to do patch-based probabilistic Chan-Vese on the wrapped interferograms.

Segmentation with neural networks is another option, which may be pursued. One way is to train a CNN network from scratch. We chose to avoid CNNs for this study, as we were unsure of the data requirement to train a functioning CNN. With the 2000+ SHA events discovered with SGLNet, it may now be feasible to accomplish this. Alternatively, we would like to replace the classi-

fication head by a segmentation decoder. It could either be custom designed or use an existing architecture such as UPerNet [63]. We may copy the framework from Panopticon [57], which use a Feature2Pyramid as neck, a UPerNet decoder and an auxiliary fully convolutional network (FCN) head to train a semantic segmentation decoder that works on the ViT backbone. The Featuer2Pyramid neck is needed to adapt ViT feature embeddings to convolutional feature pyramids used by the UPerNet. DINO does have a CNN implementation that would be compatible with UPerNet directly, but this would break our current framework.

We find the segmentation decoder to be particularly promising. It can be added onto the existing framework without loss of functionality, and can be trained to produce segmentations from the latent DINO features.

Encoding time series. Individual SHA events often appear in multiple DD interferograms. Encoding this temporal information from time series of interferograms may thus improve performance. Some of the above mentioned geospatial FMs incorporate this. A stand-alone CNN architecture with this capability and the ability to produce binary masks is presented in [64] and is worth reviewing. Alternatively, the ViT architecture of this study could be adapted to work with multiple interferograms. At the ESA Living Planet Symposium, an interesting study titled *"GMSM: A Generic Framework for Mine Site Monitoring with Multimodal Earth Observation"* was presented by Yang Mu from the Technical University of Munich.[1] A temporal attention network (TAN) is formed by concatenating the ViT output tokens from multiple interferograms together to form a single, long sequence of tokens. Temporal "position" embeddings are added to the tokens before they are passed through two consecutive self-attention blocks. This allows tokens to attend to one another across the time series, thus encoding temporal information. While Yang Mu uses the DOFA FM as backbone, TAN works with any sequence of tokens, including those produced by DINO.

[1] It has not (yet) been published anywhere

Chapter 7

Conclusion

This chapter summarizes the main findings and implications of this work. Limitations are included, together with directions for future works.

This study examines the use of machine learning methods to investigate subglacial hydrological activity through DDInSAR. Specifically, a pretrained foundational model (DINO) was used as a backbone feature extractor together with a trainable classification head. We considered training a convolution neural network from scratch, but went with the pretrained DINO Vision Transformer out of concern that annotated data was scarce. This meant implementing the algorithm (referred to as SGLNet) as a binary classifier on small chunks sampled from double difference phase interferograms with a moving window. Detected chunks are used for event detection and an experimental attention-segmentation routine was added.

Different network configurations were tested and it was found that a ViT-B backbone with patch size 16, chunk size 448, and with the interferometric phase in all three RGB channels had the best overall performance across various benchmarks. With this setup, we where able to achieve a classification precision between 0.77–0.90, recall of 0.57–0.98, and F_1-score between 0.65–0.94. These ranges reflect uncertainties in the benchmarking method, with the true network performance expected to be fixed somewhere within those bounds. In the 49 interferograms used for testing, the algorithm detected 136 subglacial hydrological events (71% of total event count), and falsely marked 5 areas as containing events (9% of all marked areas). This evaluation included events located in so much decorrelation noise that they would be practically unusable for any downstream applications. For segmentation, the method referred to as DINOseg proved best. It achieved a Dice coefficient of 0.39.

This study has concluded that vision transformers are capable of recognizing the spatial patterns of vertical surface displacement associated with transient subglacial hydrological changes. By reducing the model feature space, we showed that DINO can separate surface displacement signals from the background in interferograms even without finetuning. This shows how versatile foundational models are, and points to the great potential for foundational models within Earth observation. They are particularly useful if training data is insufficient compared to the task complexity, making it unfeasible or time consuming to train a model from scratch.

We showed how Project DynaMelt was able to map the spatiotemporal extent of subglacial hydrological changes using SGLNet on approximately 10.000 Sentinel-1 SLC products. The algorithm detected more than 2000 transient events over an area covering more than 10^6 km^2. These maps reveal the location and frequency of dynamical changes in the subglacial hydrology. We see significant activity coinciding with troughs in the bed topography and fast-flowing ice, typically upstream of large ice shelves, in regions relatively close to the grounding line. These results may be used to further the scientific community's understanding of subglacial waterflow and it's effect on ice shelf

stability, mass loss, and global sea level rise. Our findings challenge the steady-state assumption commonly used in subglacial hydrological models, and may also point to other inaccuracies.

Our performance benchmarking was limited by the inherent ambiguity that binary class labels introduce when applied to real data featuring continuous values. A future study should explore ways to alleviate this issue, either using multiple subclasses, soft labels, or by drawing inspiration from weakly supervised learning techniques. Another improvement would be to fully implement event segmentation into the algorithm. While event outlines may be generated using the attention embeddings, this method is inaccurate and does not take full advantage of the embedded information. We suggest implementing a Convolution Neural Network, either trained from scratch with additional training data, or as a segmentation decoder added onto the transformer encoder. Lastly, improvements in performance could be pursued by adding more training data, using a different foundational model, and including temporal awareness in the network. Not only would this potentially increase the number of detected events, it may also allow the algorithm to better distinguish phase patterns relating to the subglacial hydrology from e.g. grounding lines, ice flow changes or ionospheric effects, the latter of which are currently a source of false positives in the algorithm.

This study contributes to the ongoing research on utilization of foundational models for remote sensing and Earth observation applications. It shows how machine learning enables us to process large quantities of data. With SGLNet we demonstrate a path towards automatic mapping of transient changes in the hydrological system underneath the polar ice sheets. In Antarctica, subglacial hydrology represents a significant source of uncertainty for glaciological models. Being able to extensively map this system will improve our ability to forecast the evolution of Antarctica in a warming climate and predict it's impact on society through global sea level rise.

Bibliography

[1] M. Rantanen, A. Y. Karpechko, A. Lipponen, *et al.*, "The arctic has warmed nearly four times faster than the globe since 1979," *Communications Earth & Environment*, vol. 3, 2022.

[2] M. Casado, R. Hébert, D. Faranda, and A. Landals, "The quandary of detecting the signature of climate change in antarctica," *Nature Climate Change*, vol. 13, 2023.

[3] B. W. J. Miles, T. Li, and R. G. Bingham, "Totten ice shelf history over the past century interpreted from satellite imagery," *EGUsphere [preprint]*, 2025.

[4] C. F. Dowa, F. S. McCormack, D. A. Young, J. S. Greenbaum, J. L. Roberts, and D. D. Blankenship, "Totten glacier subglacial hydrology determined from geophysics and modeling," *Earth and Planetary Science Letters*, vol. 531, 2020.

[5] C. Zhao, R. Gladstone, T. Zwinger, *et al.*, "Subglacial water amplifies antarctic contributions to sea-level rise," *Nature Communications*, vol. 16, 2025.

[6] S. Ehrenfeucht, C. Dow, K. McArthur, M. Morlighem, and F. S. McCormack, "Antarctic wide subglacial hydrology modeling," *Geophysical Research Letters*, vol. 52, 1 2024.

[7] L. Gray, I. J. snd Slawek Tulaczyk, V. B. Spikes, R. Bindschadler, and K. Jezek, "Evidence for subglacial water transport in the west antarctic ice sheet through three-dimensional satellite radar interferometry," *Geophysical Research Letters*, vol. 32, 2005.

[8] N. Neckel, S. Franke, V. Helm, R. Drews, and D. Jansen, "Evidence of cascading subglacial water flow at jutulstraumen glacier (antarctica) derived from sentinel-1 and icesat-2 measurements," *Geophysical Research Letters*, vol. 48, 20 2021.

[9] S. J. Livingstone, Y. Li, A. Rutishauser, *et al.*, "Subglacial lakes and their changing role in a warming climate," *Nature*, vol. 3, 2022.

[10] J. K. Andersen, N. Rathmann, C. S. Hvidberg, *et al.*, "Episodic subglacial drainage cascades below the northeast greenland ice stream," *Geophysical Research Letters*, vol. 50, 12 2023.

[11] P. Christoffersen, M. Bougamont, A. Hubbard, S. H. Doyle, S. Grigsby, and R. Petterson, "Cascading lake drainage on the greenland ice sheet triggered by tensile shock and fracture," *Nature Communications*, vol. 9, 2018.

[12] E. Kazmierczak, T. Gregov, V. Coulon, and F. Pattyn, "A fast and simplified subglacial hydrological model for the antarctic ice sheet and outlet glaciers," *The Cryosphere*, vol. 18, 12 2024.

[13] J. Moon, H. Lee, and H. Lee, "Elevation change of cooke2 subglacial lake in east antarctica observed by dinsar and time-segmented psinsar," *Remote Sensing*, vol. 14, 18 2022.

[14] J. F. Arthur, C. Shackleton, G. Moholdt, K. Matsuoka, and J. V. Oostveen, "Evidence of active subglacial lakes under a slowly moving coastal region of the antarctic ice sheet," *The Cryosphere*, vol. 19, 2025.

[15] D. R. Fatland and C. S. Lingle, "Analysis of the 1993-95 bering glacier (alaska) surge using differential sar interferometry," *Journal of Glaciology*, vol. 44, 148 1998.
[16] P. Huybrechts, "West-side story of antarctic ice," *Nature*, vol. 458, 2009.
[17] J. J. Mohr and A. Kusk, *Sar light an introduction to synthetic aperture radar*, Mar. 2006.
[18] SentiWiki, *Copernicus programme*. [Online]. Available: https://sentiwiki.copernicus.eu/web/copernicus-programme.
[19] SentiWiki, *S1 mission*. [Online]. Available: https://sentiwiki.copernicus.eu/web/s1-mission#S1-Mission-Satellite-Description.
[20] C. Oliver and S. Quegan, *Understanding Synthetic Aperture Radar Images*. SciTech Publishing, Inc., 2004.
[21] K. Matsuoka, A. Skoglund, G. Roth, *et al.*, "Quantarctica, an integrated mapping environment for antarctica, the southern ocean, and sub-antarctic islands," *Environmental Modelling & Software*, vol. 140, 2021.
[22] J. P. M. Boncori, "Synthetic aperture radar interferometry," 30350 Remote Sensing, Lecture material from DTU course. [Online]. Available: https://kurser.dtu.dk/course/30350.
[23] Y. Mohajerani, S. Jeong, B. Scheuchl, I. Velicogna, E. Rignot, and P. Milillo, "Automatic delineation of glacier grounding lines in differential interferometric synthetic-aperture radar data using deep learning," *Nature Portfolio*, 2021.
[24] S. R. Tarekere, L. Krieger, D. Floricioiu, and K. Heidler, "Deep learning based automatic grounding line delineation in dinsar interfeograms," *EGUsphere [preprint]*, 2024.
[25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021.
[26] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
[27] V. Srinivasa. "Understanding the role of query, key, and value matrices in transformer models." Accessed 28 May 2025. (2025), [Online]. Available: https://transformers-goto-guide.hashnode.dev/understanding-the-role-of-query-key-and-value-in-transformer-models (visited on 05/28/2025).
[28] J. Mody. "An intuition for attention." Accessed 28 May 2025. (2022), [Online]. Available: https://jaykmody.com/blog/attention-intuition/ (visited on 05/28/2025).
[29] C. M. Bishop, "Neural networks," in *Pattern Recognition and Machine Learning*. Springer, 2006, ch. 5.
[30] A. B. Dahl and V. A. Dahl, *Advanced image analysis*, Lecture note from DTU Compute course 02506, 2024.
[31] M. Caron, H. Touvron, I. Misra, *et al.*, "Emerging properties in self-supervised vision transformers," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
[32] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, vol. 14, 2024.
[33] A. Kusk, "Interferometric post processing chain (ipp) user guide, Version 0.2," DTU Space, Tech. Rep., version 0.2, Aug. 25, 2022.
[34] E. S. Agency, *Sentinel-1, Slc sar data*, Alaska Satellite Facility. [Online]. Available: https://search.asf.alaska.edu/.
[35] I. H. et al., *The reference elevation model of antarctica, Mosaics, version 2*, Harvard Dataverse, 2022.
[36] ENVEO, J. Wuite, M. Hetzenecker, T. Nagler, and S. Scheiblauer, *Esa antarctic ice sheet climate change initiative (antarctic_ice_sheet_cci): Antarctic ice sheet monthly*

velocity from 2017 to 2020, derived from sentinel-1, version v1, NERC EDS Centre for Environmental Data Analysis, Jun. 28, 2021.

[37] R. Yehoshua, *Mastering logistic regression, From theory to implementation in python*, https://medium.com/data-science/mastering-logistic-regression-3e502686f0ae, Accessed: 2025-06-05, 2023.

[38] T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski, "Vision transformers need registers," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[39] M. Oquab, T. Darcet, T. Moutakanni, *et al.*, "Dinov2: Learning robust visual features without supervision," *Transactions on Machine Learning Research (TMLR)*, 2024.

[40] O. Siméoni, G. Puy, H. V. Vo, *et al.*, "Localizing objects with self-supervised transformers and no labels," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2021.

[41] SentiWiki, *S1 products*. [Online]. Available: https://sentiwiki.copernicus.eu/web/s1-products.

[42] E. Rignot, J. Mouginot, and B. Scheuchl, *Measures insar-based antarctica ice velocity map, version 2*, NASA National Snow and Ice Data Center Distributed Active Archive Center, 2017.

[43] A. M. L. Brocq, N. Ross, J. A. Griggs, *et al.*, "Evidence from ice shelves for channelized meltwater flow beneath the antarctic ice sheet," *Nature Geoscience*, vol. 6, no. 11, 2013.

[44] R. Bindschadler, H.-C. Choi, and A. Collaborators, *High-resolution image-derived grounding and hydrostatic lines for the antarctic ice sheet*, Digital media, Boulder, Colorado, USA, 2011.

[45] H. Liu, K. C. Jezek, B. Li, and Z. Zhao, *Radarsat antarctic mapping project digital elevation model, version 2*, Boulder, Colorado, USA, 2015.

[46] M. Caron, H. Touvron, I. Misra, *et al.*, "Emerging properties in self-supervised vision transformers," preprint arXiv:2104.14294v2, 2021.

[47] J. Mouginot, B. Scheuchl, and E. Rignot, *Measures antarctic boundaries for ipy 2007–2009 from satellite radar, version 2*, NASA National Snow and Ice Data Center Distributed Active Archive Center.: Boulder, Colorado USA.

[48] A. Wright and M. Siegert, *A fourth inventory of antarctic subglacial lakes*, 2012.

[49] M. Morlighem, E. Rignot, T. Binder, *et al.*, "Deep glacial troughs and stabilizing ridges unveiled beneath the margins of the antarctic ice sheet," *Nature Geoscience*, vol. 13, 2020.

[50] A. P. Wright, D. A. Young, J. L. Roberts, *et al.*, "Evidence of a hydrological connection between the ice divide and ice sheet margin in the aurora subglacial basin, east antarctica," *Journal of Geophysical Research: Earth Surface*, vol. 117, 2012.

[51] S. Fu, M. Hamilton, L. Brandt, A. Feldmann, Z. Zhang, and W. T. Freeman, "Featup: A model-agnostic framework for features at any resolution," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Published as a conference paper at ICLR 2024, 2024.

[52] R. Müller, S. Kornblith, and G. Hinton, "When does label smoothing help?" In *Advances in Neural Information Processing Systems 32*, Published as a conference paper at NeurIPS 2019, 2019.

[53] J. Xia, N. Yokoya, and B. Adriano, "Building damage mapping with self-positive-unlabeled learning," Preprint arXiv:2111.02586v1, 2021.

[54] A. Fuller, K. Millard, and J. R. Green, "Croma: Remote sensing representations with contrastive radar-optical masked autoencoders," in *Advances in Neural Information Processing Systems 36*, 2023.

[55] Z. Xiong, Y. Wang, F. Zhang, *et al.*, "Neural plasticity-inspired multimodal foundation model for earth observation," preprint arXiv:2403.15356, 2024.

[56] Y. Wang, N. A. A. Braham, Z. Xiong, C. Liu, C. M. Albrecht, and X. X. Zhu, "SSL4EO-S12: A large-scale multimodal, multitemporal dataset for self-supervised learning in earth observation," *IEEE Geoscience and Remote Sensing Magazine*, vol. 11, no. 3, 2023.

[57] L. Waldmann, A. Shah, Y. Wang, *et al.*, "Panopticon: Advancing any-sensor foundation models for earth observation," preprint arXiv:2503.10845, 2025.

[58] J. Jakubik, F. Yang, B. Blumenstiel, *et al.*, "Terramind: Large-scale generative multimodality for earth observation," preprint arXiv:2504.11171, 2025.

[59] G. Sumbul, C. Xu, E. Dalsasso, and D. Tuia, "Smarties: Spectrum-aware multi-sensor auto-encoder for remote sensing images," preprint arXiv:2506.19585, 2025.

[60] X. Guo, J. Lao, B. Dang, *et al.*, "Skysense: A multi-modal remote sensing foundation model towards universal interpretation for earth observation imagery," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[61] G. Tseng, A. Fuller, M. Reil, *et al.*, "Galileo: Learning global & local features of many remote sensing modalities," preprint arXiv:2502.09356, 2025.

[62] arnt børre salberg. "Fm4cs: A versatile foundation model for earth observation applications: Arnt-børre salberg (nr)." YouTube video, accessed on 28-06-2025. (2025), [Online]. Available: https://www.youtube.com/watch?v=IogIMCNGmS4%5C&ab_channel=SFIVisualIntelligence.

[63] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11209, 15th European Conference on Computer Vision, ECCV 2018, Springer Verlag, 2018.

[64] V. S. F. Garnot and L. Landrieu, "Panoptic segmentation of satellite image time series with convolutional temporal attention networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[65] N. Yague-Martinez, P. Prats-Iraola, F. R. Gonzales, *et al.*, "Interferometric processing of sentinel-1 tops data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, 4 2016.

Appendix A

Sentinel-1 specifications

This appendix gives additional information on Sentinel-1 related topics. This includes the TOPS acquisition mode and the IW swath mode.

A.1 Terrain Observation by Progressive Scans (TOPS)

For Sentinel-1's Interferometric Wide (IW) and Extra Wide (EW) swath modes the sentinel uses TOPS. This is a technique where the radar beam is steered along subswaths in the azimuth direction. This is illustrated in Figure A.1 for IW mode.

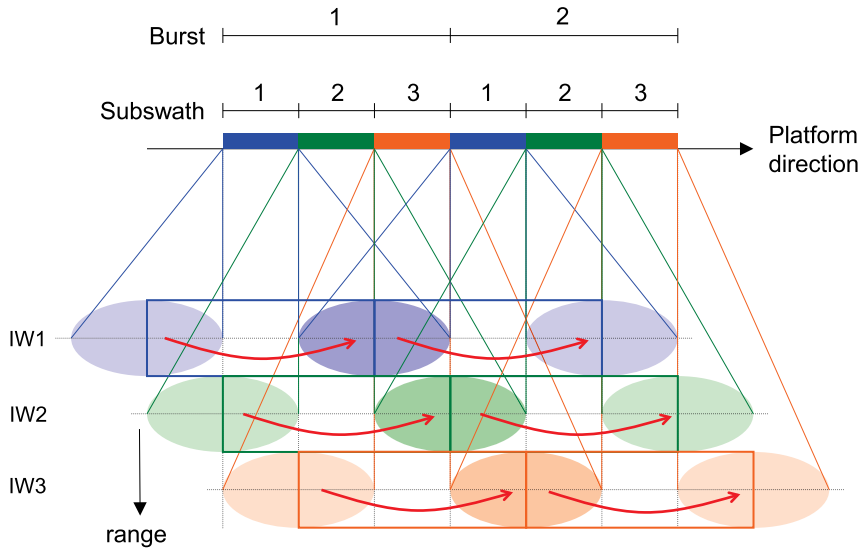


Figure A.1: **TOPS scan pattern for S1 IW mode.** The radar acquires the first burst from each of the three subswaths, starting with IW1 (blue), then IW2 (green) then IW3 (orange). The red arrows indicate how the radar beam is steered in the same direction as the platform moves. The second burst starts back at subswath IW1 and the cycle repeats. Illustration is from [65].

A.2 Interferometric Wide Swath Mode (IW) product parameters

Table A.1 provides SLC product parameters for Sentinel-1 IW mode.

Beam id.	IW1	IW2	IW3
Incidence angles	32.9°	38.3°	43.1°
Slant range resolution	2.7 m	3.1 m	3.5 m
Range Bandwidth	56.5 MHz	48.3 MHz	42.79 MHz
Azimuth resolution	22.5 m	22.7 m	22.6 m
Processing Bandwidth	327 Hz	313 Hz	314 Hz
Doppler Centroid span (Δf_{DC})	5.2 kHz	4.4 kHz	4.6 kHz
Slant range pixel spacing	2.3 m		
Range sampling frequency	64.35 MHz		
Azimuth pixel spacing	14.1 m		
Azimuth sampling frequency	486.49 Hz		
Azimuth steering angle	±0.6°		
Burst length (T_{focused})	2.75 s / ≈ 20 km		
Ground Swath width	250 km		
Slice length	170 km		
Orbital Repeat Cycle	12 days		
Orbit height	698 – 726 km		
Wavelength	5.547 cm		
Polarization	Single (HH or VV) or Dual (HH+HV or VV+VH)		

Table A.1: **SLC product parameters for S1 IW mode.** Copied from [65].

Appendix B

SGLNet in depth

This appendix provides additional insights into SGLNet. This includes attention maps, 2D feature space projections, and classification maps using K-means and similarity predictions.

B.1 DINO attention maps

By extracting the attention score with which the [`class`] token attends to individual patches, it is possible to visualize which parts of the interferograms is given particular attention by the transformer. Figure B.1 shows the attention score for all detected events from Pine Island track 065 descending in the testing dataset. It is these salient feature maps that are segmented by DINOseg to generate event outlines.

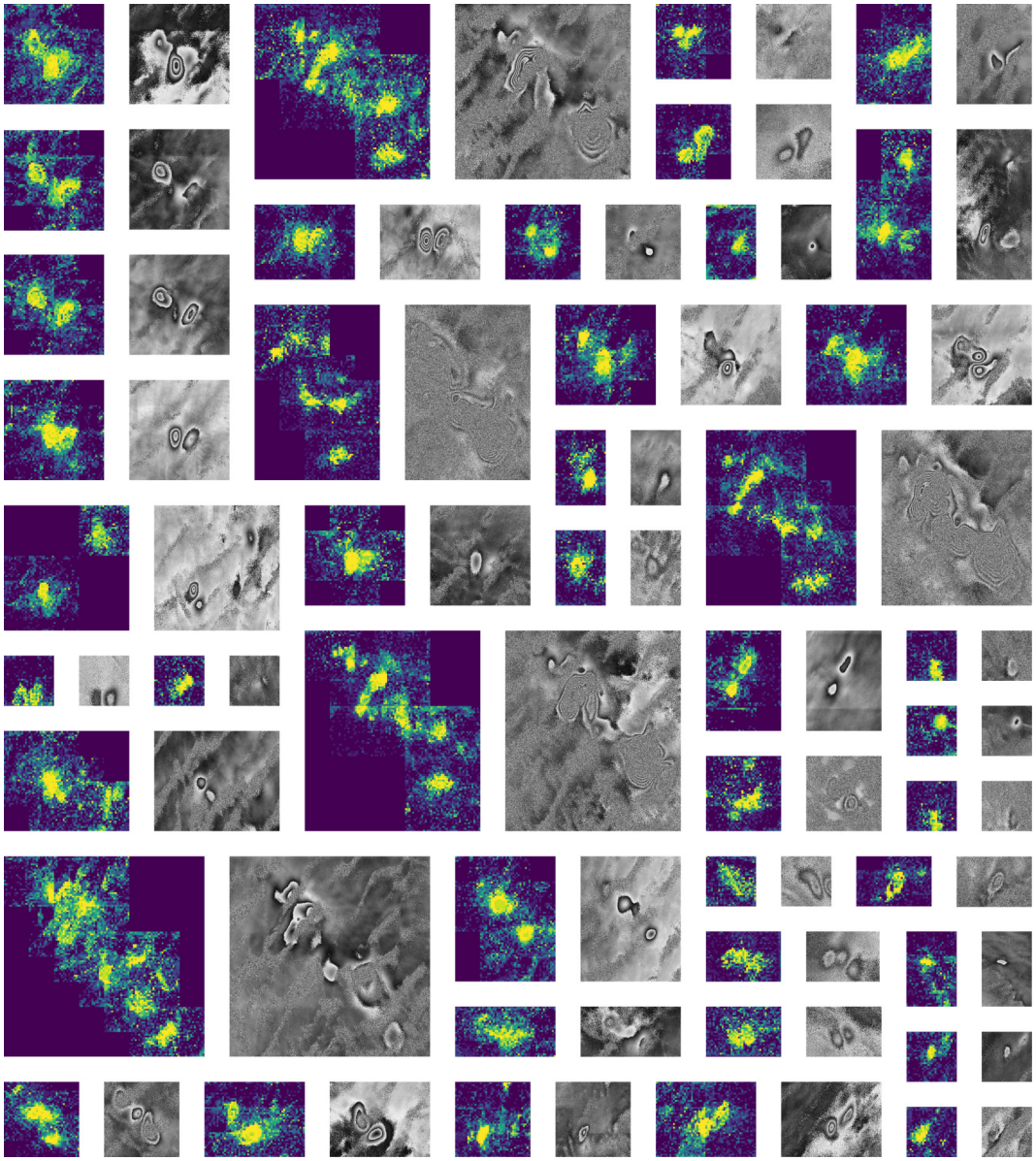


Figure B.1: **Mosaic of DINO attention maps.** Maps are from track 065 interferograms. Each map is the average over all 12 attention heads for each patch that the [`class`] token attends to. The model used is ViT-B/16 Phase/448.

B.2 Two-dimensional projection of feature space

This section investigates clusters in the projected two-dimensional feature space discussed in Section 5.2.1. Nine different regions are chosen, and marked by nine coloured boxes in Figure B.2. The embeddings in each region are evaluated by tracing them back to the specific chunks in specific interferograms from which they originate. Figure B.3 shows six representative examples for each region. Here follows a short description of the qualitative findings.

Region A. While embeddings are very tightly clustered in the projected 2D feature space, there is no obvious similarity between the corresponding chunks. These include decorrelation noise, plane regions, a few noisy grounding lines and similar.

Region B. This region is exclusively from chunks located on the edge of the interferogram and containing grounding lines.

Region C. This cluster or set of clusters come from chunks that can be described similar to those from Region A.

Region D. Same as Region A and C.

Region E. Same as A, C, and D.

Region F. This region was chosen as it only contains SHA events. Features from this region are found to come from chunks with small to medium sized SHA events and high coherence.

Region G. Similarly to Region F, this region only contain grounding lines. Inspection shows that this region appears to be from noisier parts of the grounding lines.

Region H. A mix of grounding lines and SHA events map to this region. Maybe not surprisingly, all captured SHA events are large and have high fringe rates, which makes them similar to grounding lines when viewed as individual chunks.

Region I. This region is comfortably within the large cluster of embeddings labels 'generic' i.e. with no presence of grounding lines or SHA events. It is found that this region encompass chunks from decorrelated parts of the interferograms, or areas with very little spatial variety.

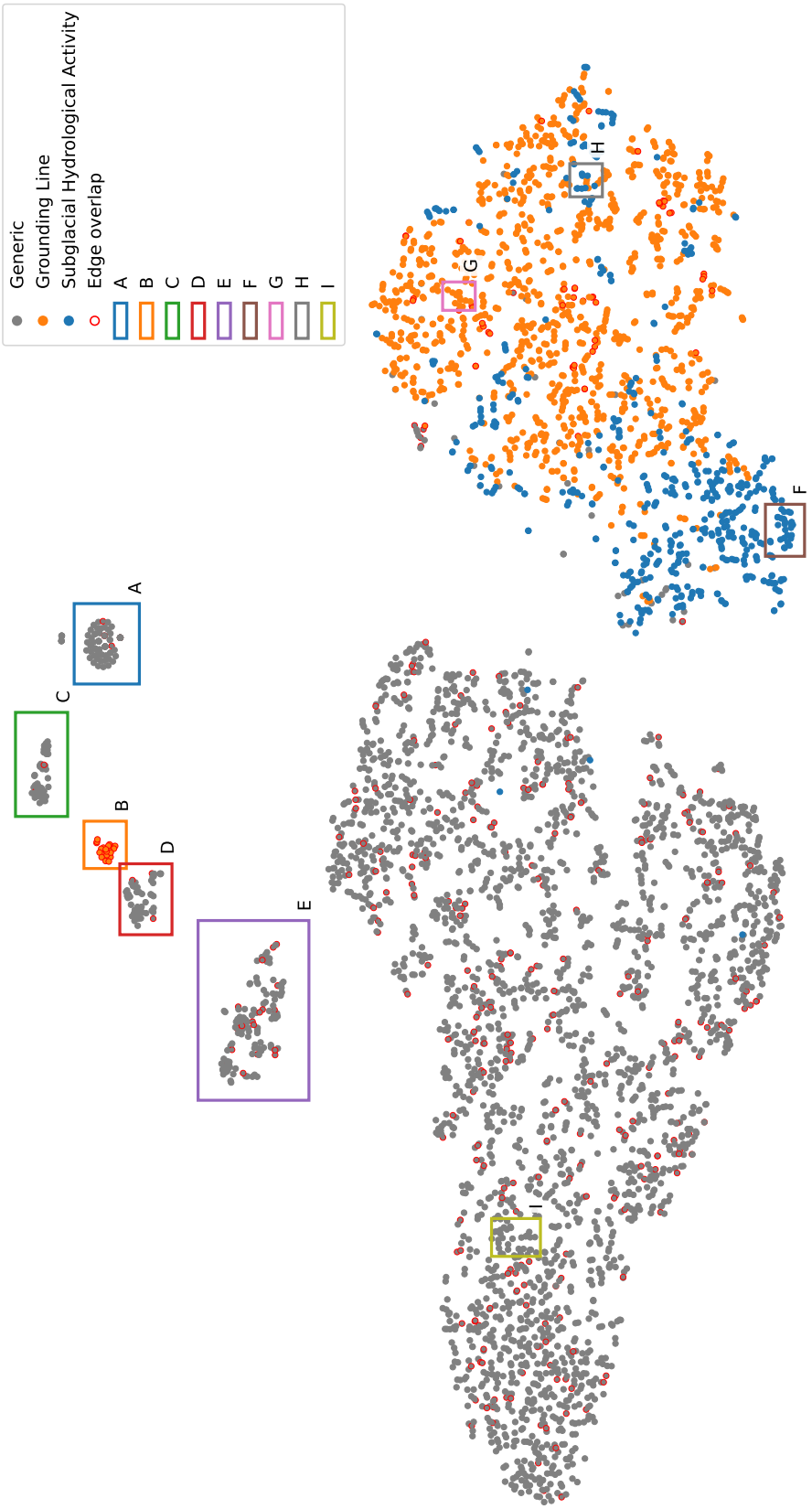


Figure B.2: **Nine areas of interest in the 2D projected feature space.** Chunk examples are shown in Figure B.3. The red circular outlines around points on the left indicate samples situated on the edge of interferograms.

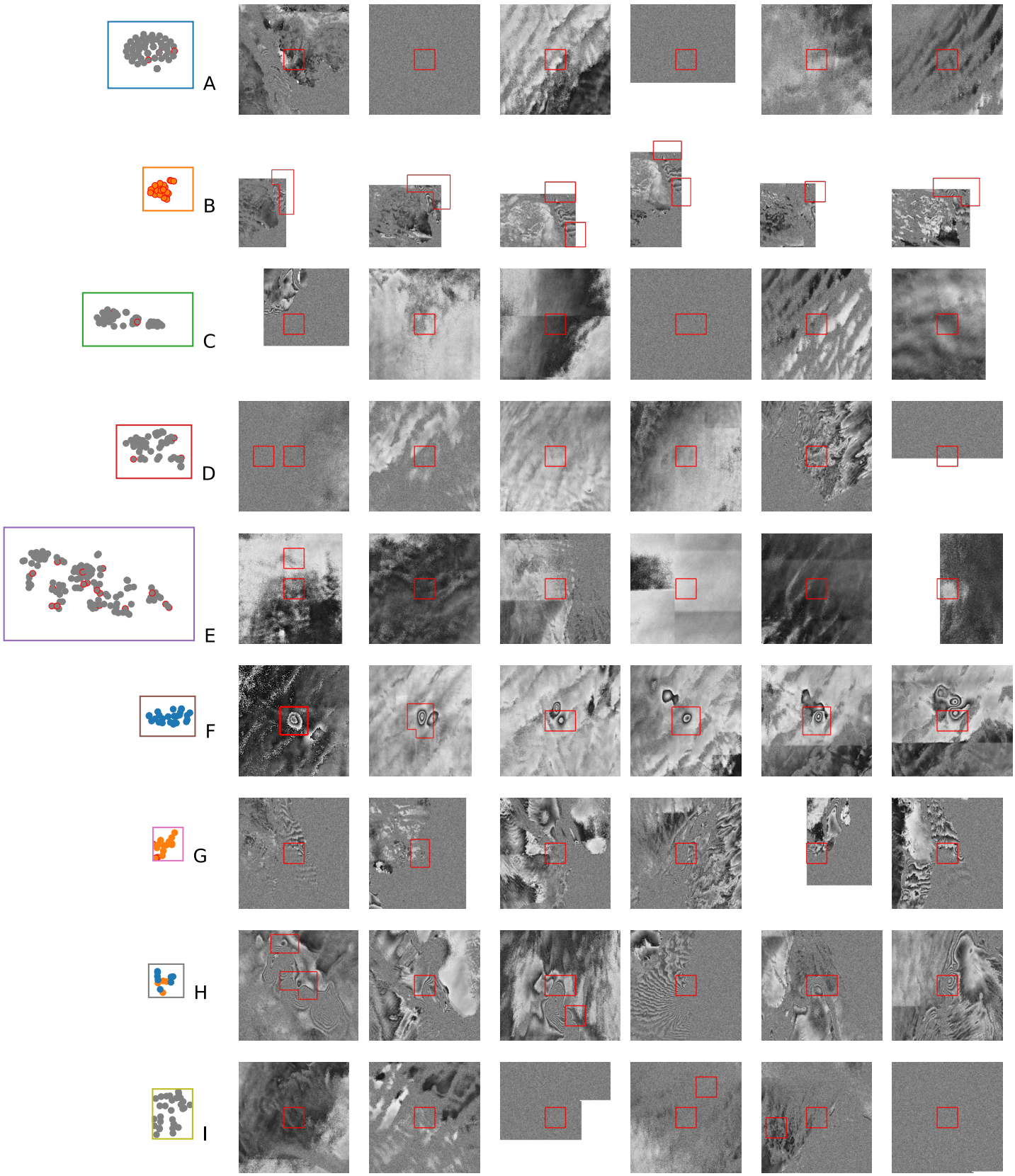


Figure B.3: **Chunks clustering illustrated.** Each row is one of the nine regions from Figure B.2. The red circular outlines around points on the left indicate samples situated on the edge of interferograms.

B.3 Maps from K-means and similarity classification

Here follows a few maps showing classification with K-means and cosine similarity, as discussed in Section 5.2.2. All features are extracted with configuration ViT-B/16 Phase/224. Examples are shown for two acquisitions: one from Track 065 (same as the one used in Chapter 5), and one from Track 010 (which includes a large grounding line).
Figure B.4 shows classification results from the three methods. Figure B.5 shows cosine similarity maps of the same two interferograms.

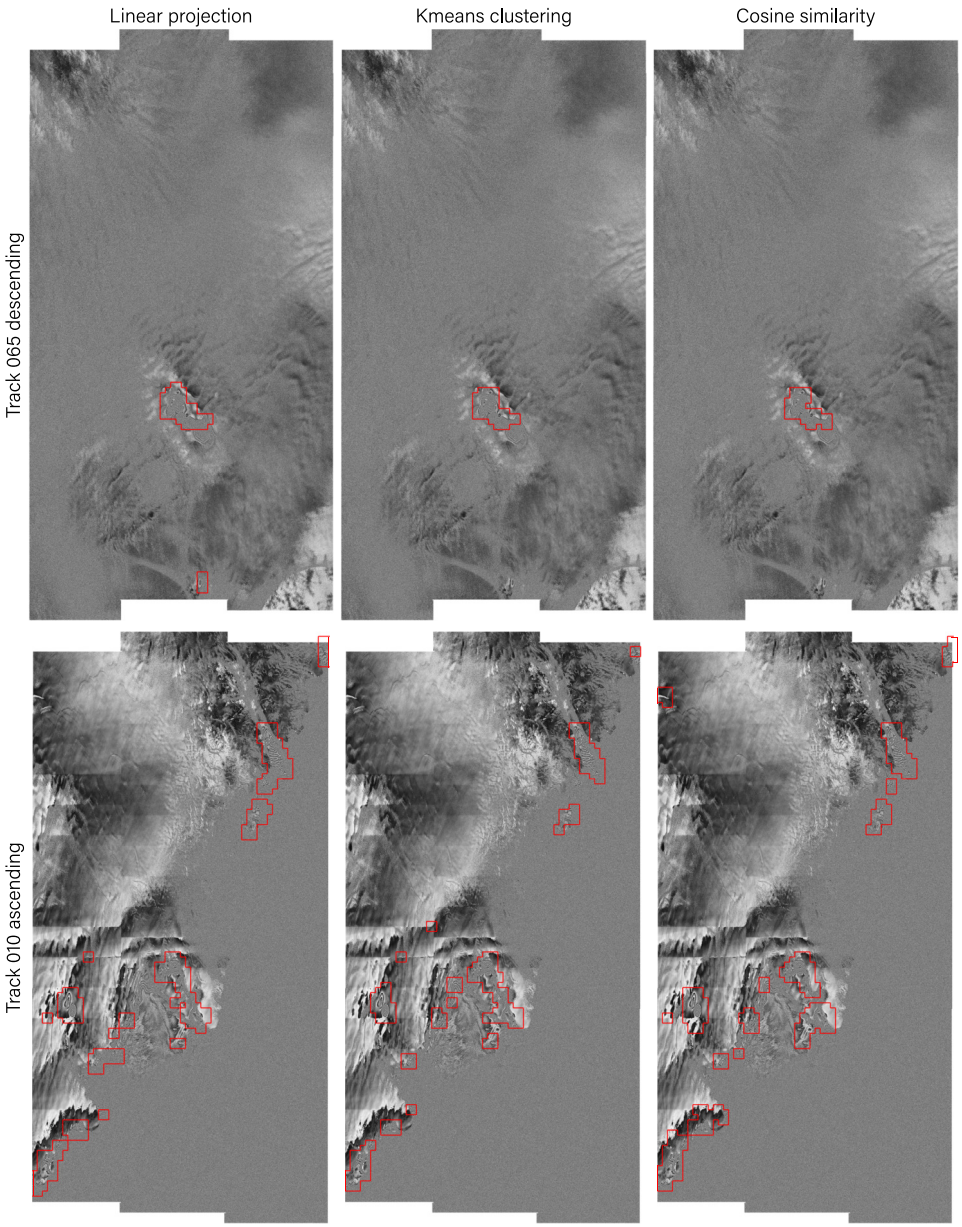


Figure B.4: **Results from three classification methods.** (left) Classifier head. (middle) Kmeans clustering. (right) Cosine similarity. (top) Track 065 from period 18/03/2020 - 30/03/2020. (bottom) Track 010 from period 22/07/2017 - 03/08/2017 with grounding lines. Detected chunks are outlined in red.

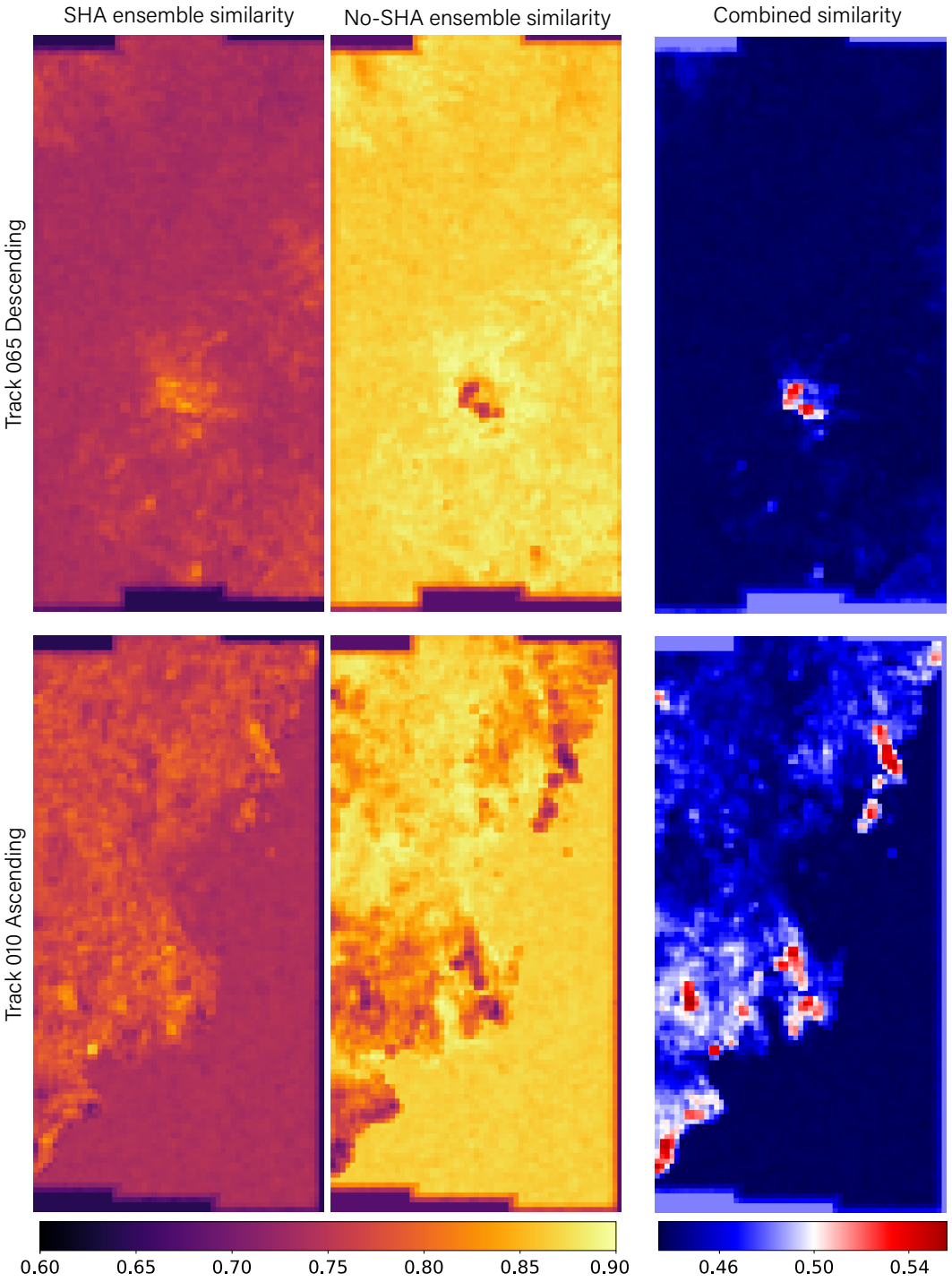


Figure B.5: **Similarity maps.** (left) Cosine similarity between each chunk in the interferogram and a reference ensemble of SHA-positive features. (middle) Similarity with SHA-negative features. (right) Softmax between the positive and negative similarity maps. (top) Track 065 from period 18/03/2020 - 30/03/2020. (bottom) Track 010 from period 22/07/2017 - 03/08/2017 with grounding lines. Detected chunks are outlined in red.

Appendix C

Project DynaMelt findings with SGLNet

This appendix is related to the results from Project DynaMelt using SGLNet to investigate regions of Antarctica. It shows the same maps as in fig. 5.12, but enlarged for better visibility. A summary of tracks used are also included.

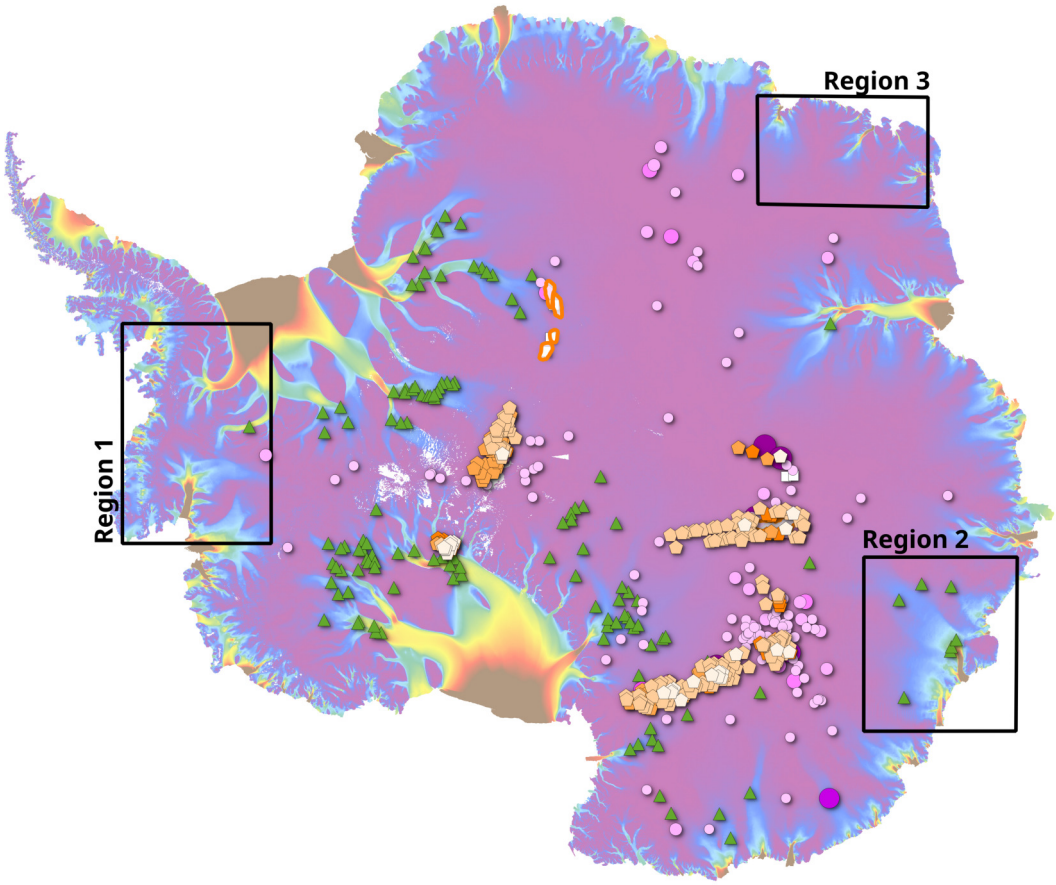


Figure C.1: **Overview of regions investigated by Project DynaMelt.** SGLNet is used to detect events from S1 tracks in these areas.

The results presented in Section 5.3.2 are from a collection of Sentinel-1 tracks covering three separate regions. These are specified in Table C.1.

	Region 1 - Ellsworth Land			
Tracks	052D	053D	007A	
Frames	858–871	850–862	888–904	
	Region 2 - Wilkes Land			
Tracks	099A	070A	055A	041A
Frames	941–951	943–948	940–950	935–948
	Region 3 - Enderby Land			
Tracks	146A	102A	058A	014A
Frames	925–936	929–941	935–946	939–945

Table C.1: **Tracks and frames in Antarctic survey.** These are located within three regions, and used by Project DynaMelt to detect subglacial hydrological activity with SGLNet.

The events detected from these regions are shown in Figure C.2, Figure C.3, and Figure C.4.

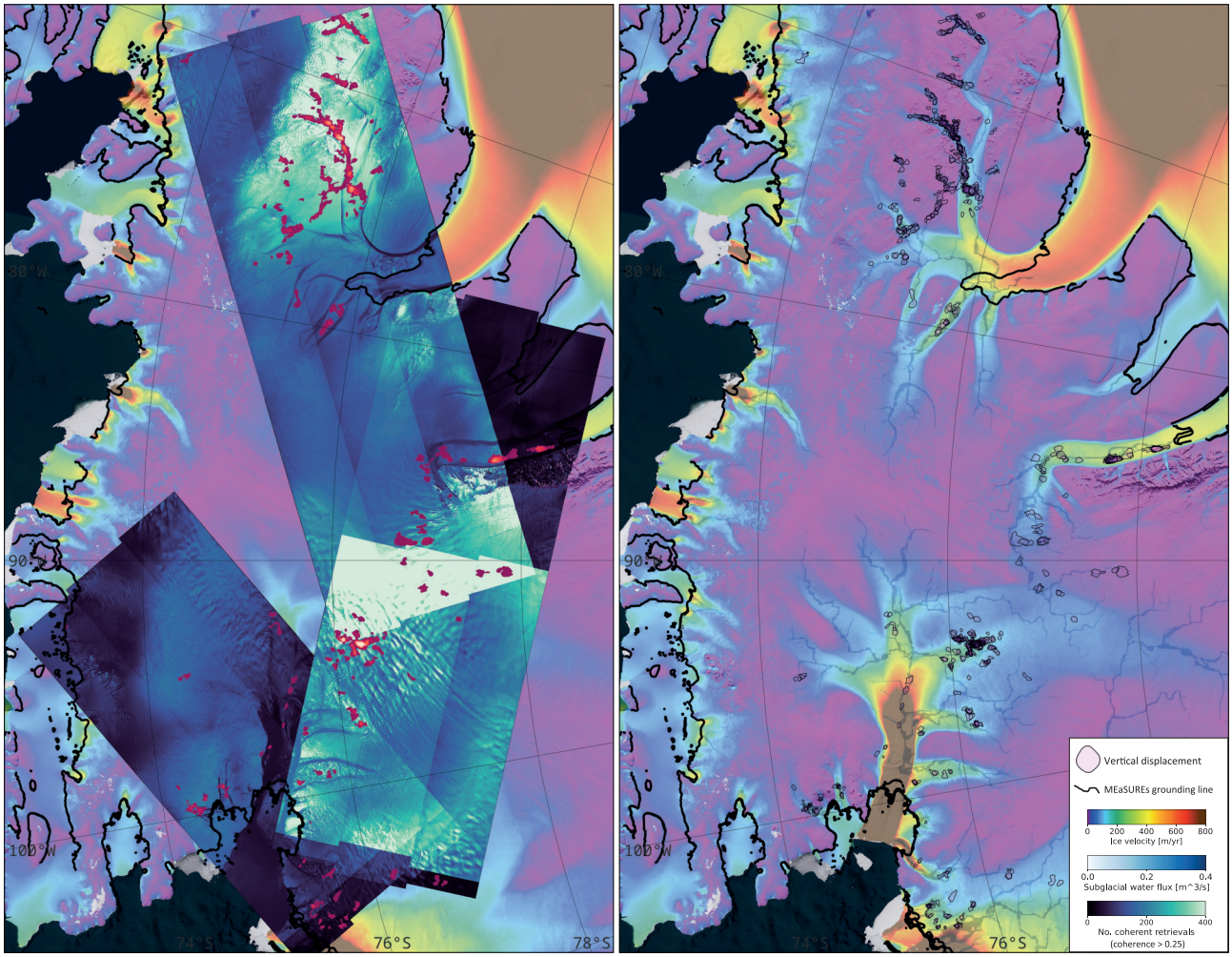


Figure C.2: **DynaMelt Region 1.** The left-most maps show event heatmaps plotted atop a coherence count from S1 products. These are accompanied by maps show individual outlines of events. Background maps of ice flow velocity [42] and subglacial water flux [43] are shown together with grounding lines [47] and previously identified subglacial lakes [48]. The figure material is part of Project DynaMelt.

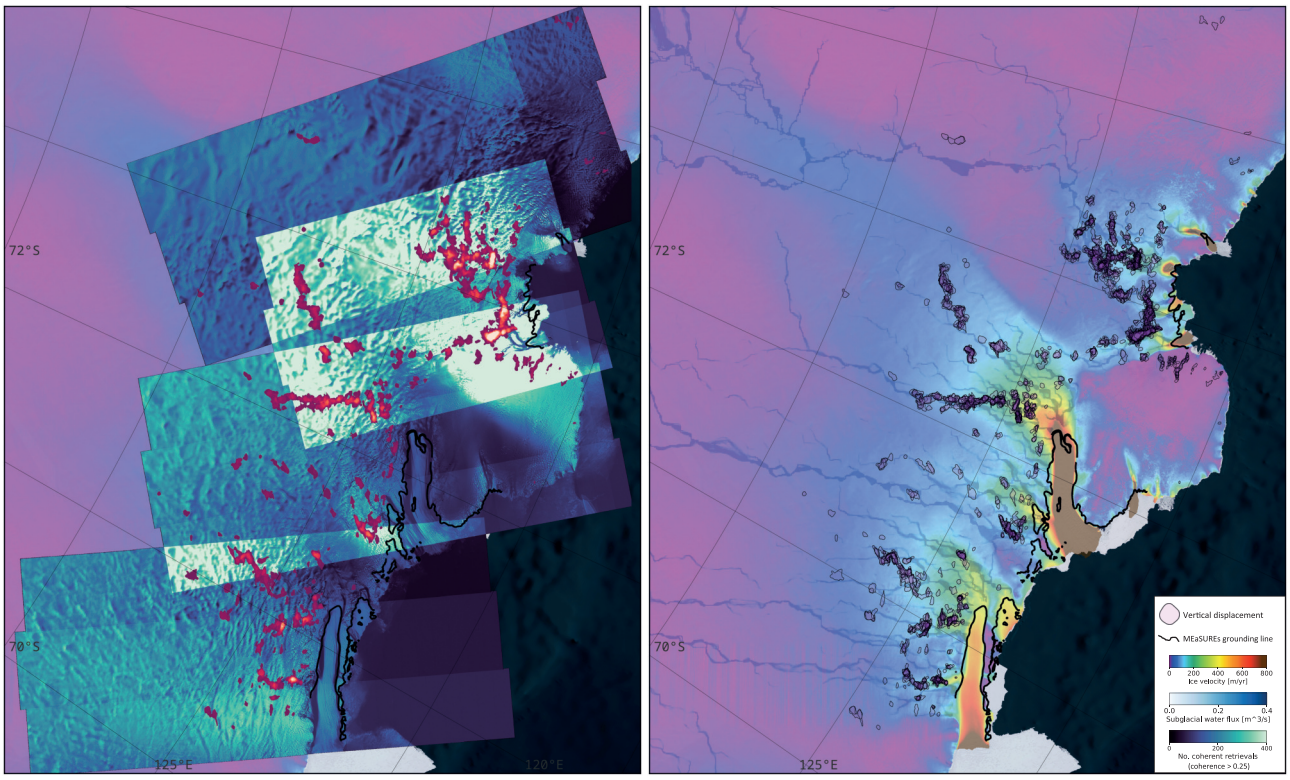


Figure C.3: **DynaMelt Region 2.** The left-most maps show event heatmaps plotted atop a coherence count from S1 products. These are accompanied by maps show individual outlines of events. Background maps of ice flow velocity [42] and subglacial water flux [43] are shown together with grounding lines [47] and previously identified subglacial lakes [48]. The figure material is part of Project DynaMelt.

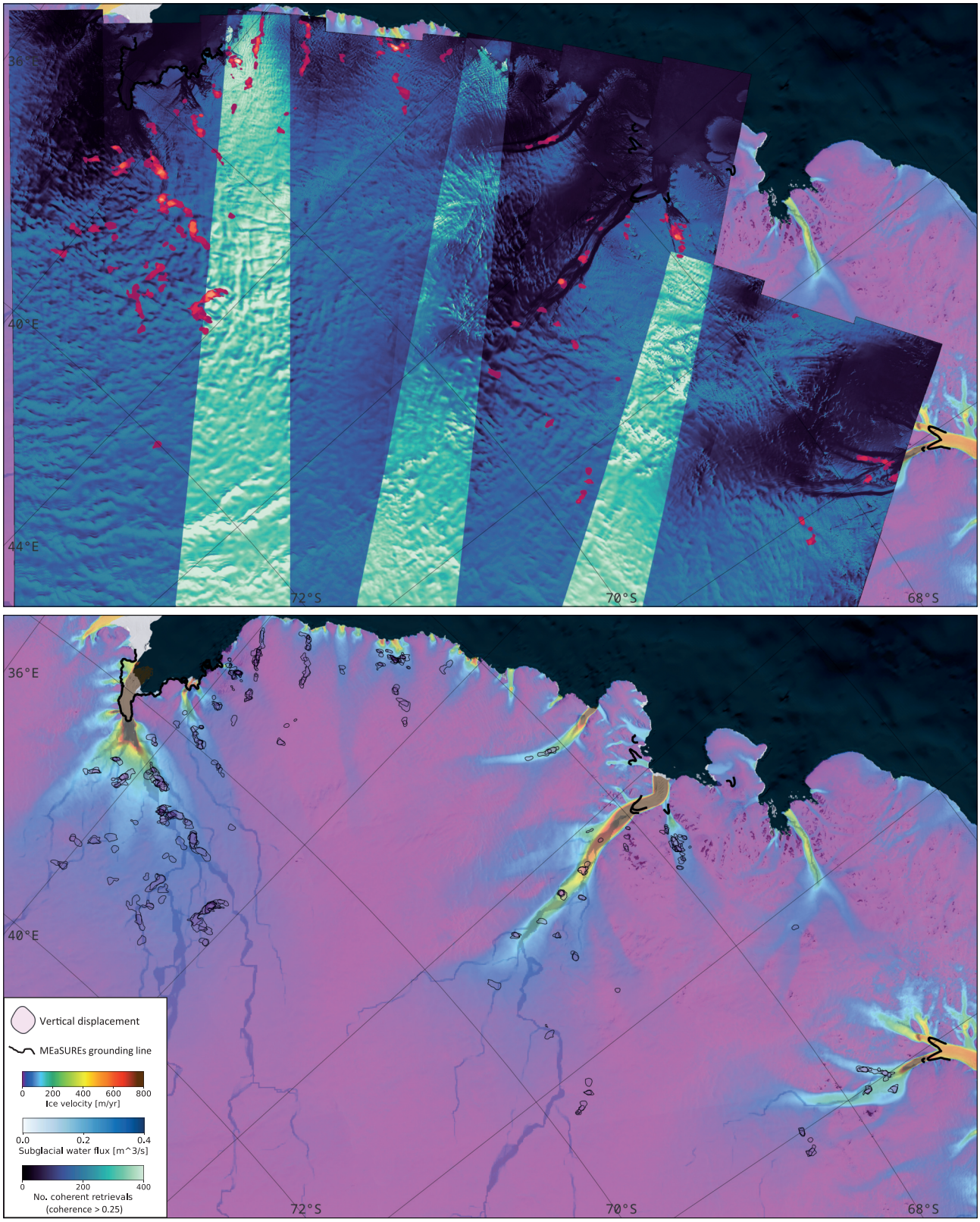


Figure C.4: **DynaMelt Region 3.** The left-most maps show event heatmaps plotted atop a coherence count from S1 products. These are accompanied by maps show individual outlines of events. Background maps of ice flow velocity [42] and subglacial water flux [43] are shown together with grounding lines [47] and previously identified subglacial lakes [48]. The figure material is part of Project DynaMelt.