Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Nina Valberg Nyegaarden

Multi-Task Attention-Based Convolutional Neural Network for SAS Change Detection with Self-Supervised Pre-Training

Master's thesis in Cybernetics and Robotics
Supervisor: Kristin Ytterstad Pettersen
Co-supervisor: Øivind Midtgaard and Narada Warakagoda
June 2023

NTNU
Norwegian University of Science and Technology

FFI Norwegian Defence Research Establishment

Nina Valberg Nyegaarden

Multi-Task Attention-Based Convolutional Neural Network for SAS Change Detection with Self-Supervised Pre-Training

Master's thesis in Cybernetics and Robotics
Supervisor: Kristin Ytterstad Pettersen
Co-supervisor: Øivind Midtgaard and Narada Warakagoda
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Problem description

This thesis is affiliated with the Norwegian Defence Research Establishment (FFI).

Synthetic Aperture Sonar (SAS) deep learning-based change detection aims to automatically identify temporal changes over a seafloor region by comparing pre-change and post-change SAS images. However, the detection performance is usually subject to several restrictions, including the scarcity of labeled SAS images to train deep-learning models. The precedent project thesis explored the use of transfer learning and data augmentation to address the issue of limited SAS data. However, the results indicate that the distribution discrepancy between the target SAS and source VHR domains cannot be easily bridged, restricting the performance of transfer learning-based methods.

In this thesis, the student will develop and analyze a self-supervised pre-training change detection method for SAS imagery in order to teach the model to recognize relevant changes with a minimal amount of labeled data.

Specifically, the student will perform the following tasks:

1. Conduct a literature review on deep-learning methods for change detection with a small, annotated dataset, focusing on self-supervised strategies.
2. Evaluate the suitability of the methods found in task 1 and implement one or more of these methods.
3. Evaluate the results through testing on real SAS images.

If time permits, alternative methods within unsupervised learning, domain adaptation, or multi-task learning will be investigated and considered implemented for performance comparison.

Abstract

Underwater mines pose a serious threat to marine vessels, serving as means to control navigation or prevent passage through restricted waters. To address this, mine countermeasures (MCM) focus on locating and disabling mines to ensure freedom of movement. Autonomous underwater vehicles (AUVs) equipped with synthetic aperture sonar (SAS) are increasingly used for mine searches, providing detailed acoustic imagery of the sea floor. Since AUVs collect large amounts of SAS imagery, automated change detection techniques (ACD) have proven valuable in discriminating potential mines from other objects. While ACD of mines has been studied extensively, the outstanding performance of deep convolutional neural networks (DCNNs) for image classification has created an interest in how deep learning (DL) may be useful for change detection of SAS imagery. Recent studies reveal that DCNNs offer superior performance in mine detection, with a greater probability of detecting mine shapes and lower false alarm rates compared to traditional target classifiers. However, developing deep learning-based applications in a military context is challenging due to the lack of high-quality, sufficiently large labeled datasets that can be used to learn from and gain insight to. This thesis explores the potential of self-supervised learning (SSL) as a pre-training technique for DL-based mine change detection of SAS imagery. SSL eliminates the need for labeled training data and addresses the data scarcity problem encountered in supervised learning. The pre-trained SSL weights are fine-tuned using a multi-task attention-based DCNN named Siam R-CNN, which aims to simultaneously learn object-based bounding boxes and pixel-based change maps of deployed mines. Experimental evaluation conducted on a HISAS-1030 SAS dataset demonstrates the potential of Siam R-CNN for DL-based mine change detection.

Sammendrag

Undervannsminer utgjør en betydelig trussel mot marinefartøyer, og brukes til å styre bevegelse eller hindre passasje gjennom minelagte farvann. Minemottiltak (MCM) har som mål å lokalisere og uskadeliggjøre miner for å muliggjøre fri bevegelse. Autonome undervannsfarkoster (AUVer) utstyrt med syntetisk apertur-sonar (SAS) brukes i økende grad til å søke etter miner og gir et høyoppløselig og detaljert bilde av havbunnen. Det kan imidlertid være vanskelig å detektere miner i SAS-bilder på grunn av redusert bildekvalitet og/eller komplekse sjøbunnsforhold. Derfor har automatiserte endringsdeteksjonsteknikker (ACD) vist seg å være nyttig for minedeteksjon ved å sammenlikne bilder tatt på to forskjellige tidspunkter. De siste årene har dype konvolusjonelle nevrale nettverk (DCNN) revolusjonert feltet innen kunstig intelligens og bildegjenkjenning på grunn av deres evne til å ekstrahere relevante endringer uten behov for menneskelig domenekompetanse. Nylige studier har dessuten vist at dyp læringsbasert (DL) minedeteksjon reduserer antall "falske alarmer" sammenliknet med tradisjonelle deteksjonsmetoder. Imidlertid er utviklingen av DL-baserte applikasjoner i militær sammenheng utfordrende på grunn av begrensede mengder med annotert treningsdata. Denne masteroppgaven undersøker potensialet til self-supervised læring (SSL) som et pretreningssteg for DL-basert endringsdeteksjon av sonarbilder. SSL takler utfordringen med mangelen på tilstrekkelig med annotert data ved å bruke et umerket datasett for å lære nyttige temporære egenskaper mellom SAS-bildene. Avhandlingen presenterer også Siam R-CNN, en nettverksarkitektur som lærer objektbaserte "bounding-boxes" og pikselbaserte endringskart for utplasserte miner. Den foreslåtte metoden testes på HISAS-1030 SAS-bilder og viser lovende resultater for anvendelse og utvikling av DL-baserte endringsdeteksjonsmetoder for sonarbilder.

Contents

List of Tables

List of Figures

Preface

This master's thesis is submitted as a part of the requirements for the master's degree at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology. The work presented in this thesis has been carried out under the supervision of Prof. Kristin Y. Pettersen and Principal Scientists at FFI, Øivind Midtgaard and Narada Warakagoda.

This master's thesis is a continuation of a specialization project I conducted during the autumn of 2022. As is customary, the specialization project is not published. This means that important background theories and methods from the project report will be restated in full throughout this report to provide the best reading experience. Below, a complete list of the material included from the specialization project is listed.

- Chapter 1 (Specifically sections 1.2.1, 1.2.3, and with some changes to section 1.2.4)
- Chapter 2 (Parts of Section 2.2.3)

Throughout this project, I have been fortunate to have access to a range of tools and resources that have contributed to the results presented in this thesis. FFI, through Øivind Midtgaard and Narada Warakagoda, provided me with access to valuable pre-processed SAS images to use in my work and answered any questions regarding the images and current change detection technologies for SAS images. Additionally, NTNU granted access to the IDUN high-performance computational resource, allowing me to train the network with a large number of parameters. The IDUN cluster is described in a technical report by Själander et al. (2019). Finally, I have gained valuable experience from reading various reports on change and object detection. Notably, the works of

Li et al. (2020) and Ren et al. (2016) have significantly influenced the selection of the network architecture presented in Chapter 2. Additionally, the paper by Leenstra et al. (2021) has motivated the self-supervised pre-training methods presented in the same chapter. Overall, these reports have demonstrated remarkable results in their respective domains.

Unless otherwise stated, all figures and illustrations have been created by the author.

I would like to express my deep gratitude to my supervisors for their technical and emotional support throughout this process. I also want to thank my family for their continuous encouragement and interest in my technical education.

Nina Valberg Nyegaarden
Trondheim, June 2023

Chapter 1

Introduction

This chapter gives an introduction to the essential topics relevant to this work. It is assumed that the intended audience of this thesis has knowledge of underlying machine learning concepts. The literature review in Section 1.2 provides any supplementary information required to comprehend the content of this thesis.

1.1 Motivation

High-quality, accurate maps of the seafloor are essential in a range of industries, including offshore survey mapping and monitoring, search for objects, and military applications such as naval mine countermeasures and intelligence, surveillance, and seabed warfare (Hansen; 2022). Regarding seafloor mapping, two considerations are important: the ability to discriminate fine features (spatial resolution) and the ability to cover a wide area (Callow et al.; 2012). In recent years, synthetic aperture sonar (SAS) has emerged as a state-of-the-art technology offering high resolution and larger coverage rates. For autonomous underwater vehicles (AUVs), SAS technology has become the leading approach for imaging and mapping the seabed (Hansen; 2022). As a result, SAS has largely superseded the use of traditional side scan sonar (SSS) systems in marine operations.

The Norwegian Defence Research Establishment (FFI) has a well-established collaboration with the Norwegian company Kongsberg Maritime in developing AUV and SAS

technology. Kongsberg Maritime offers multiple products within the HUGIN family of AUVs and HISAS family of SAS systems, partly developed at FFI (Hansen; 2022). HISAS-1030 is the third generation of interferometric SAS systems used on HUGIN vehicles. Interferometric SAS technology combines interferometry with aperture synthesis, significantly improving bathymetric mapping efficiency compared to conventional side scan sonars (Callow et al.; 2012). In addition, HISAS-1030 provides a range-independent resolution of approximately 3×3 cm up to a distance of more than 200 meters from both sides, allowing for detecting and correctly classifying mines and other small objects (Kongsberg Maritime; 2010).

Automated target recognition (ATR) techniques have traditionally been used to automatically detect and classify seafloor mines. However, most existing systems for automated mine recognition struggle with an abundance of false alarms (detection of non-mines), especially in complex seafloor areas with high clutter density or when the characteristics of the mines are unknown (Warakagoda and Midtgaard; 2018). In recent decades, FFI and other defense-related research establishments have worked on developing automated change detection techniques (ACD) to identify changes on the seafloor, such as recently deployed naval mines. ACD techniques are designed to detect significant changes rather than specific targets and can help filter out non-relevant variations that may trigger false alarms in ATR. During a reference survey, SAS images of ports, inlets, or sea lines of communications are captured to serve as a baseline, assuming the seafloor is free of mines. Then, during a repeat survey, the change detection algorithm identifies relevant new objects exclusively present in the repeat pass image. In the MANEX'14 sea trials in 2014, FFI performed naval mine hunting with their HUGIN-HUS AUV equipped with a HISAS-1030 sonar. The change detection algorithm successfully detected all eight deployed mine-sized objects during two repeat missions (Midtgaard; 2018). However, several false alarm detections on individual survey lines occurred due to target-like fish responses and specular reflections. To mitigate the false alarms observed in the MANEX'14 trials, a matched filter sensitive to proud, mine-sized objects and a multi-line fusion method were implemented to eliminate inconsistent detections across survey lines. Designing features that can effectively discriminate between mines and other irrelevant features remains a critical part of the change detection algorithm and has traditionally been a time-consuming and labor-intensive process.

In recent years, artificial intelligence (AI) and machine learning have become popular subjects both within and outside of the scientific community. Particularly, the emergence of deep convolutional neural networks (DCNNs) has led to remarkable advancements in remote sensing classification and change detection. DCNNs automatically identify and extract relevant features from a given dataset. Thus, they eliminate the tedious and manual feature design process that is distinctive for traditional classification algorithms. Furthermore, deeper networks can extract intricate and non-linear patterns and representations from the data, leading to improved performance.

While DCNNs can extract highly abstract feature representations from images, their detection performance relies on having a substantial number of training samples. However, collecting a large dataset of high-resolution images can be challenging, and the issue of poor-quality training data remains a significant problem. As a result, it is common to evaluate neural networks on publicly available benchmark datasets specifically labeled for different tasks, such as object classification and change detection. However, there currently exists an insufficient amount of large, labeled SAS datasets, limiting the efficiency of deep learning-based networks when trained in an end-to-end supervised manner. That being the case, the potential of applying a DCNN for change detection of SAS imagery has not yet been studied in detail.

Researchers employ various strategies to ensure efficient learning when limited training data are available. These strategies include transfer learning, data augmentation, and unsupervised learning methods (Shafique et al.; 2022). The specialization project (Nyegaarden; 2022) explored the use of transfer learning for detecting temporal changes in SAS imagery where limited SAS training data were available. The results indicated that pre-training the DCNNs on a larger amount of very high-resolution (VHR) training data enhanced change detection performance for SAS imagery. However, the best-performing model, Siamese U-Net++, still struggled to detect changes in complex environments. This suggests that the distribution discrepancy might be substantial, making a direct transfer of feature extractor capabilities challenging. To mitigate the discrepancy between domains, the project report suggested using SAR data as the source domain or implementing domain adaptation techniques to bridge the gap between the VHR and SAS domains.

Recently, a new pre-training approach, self-supervised learning (SSL), has demonstrated promising results in various applications. SSL does not require annotated labels.

Instead, it is conducted on input data by solving auxiliary tasks defined on the input data samples. By doing so, SSL overcomes the domain discrepancy issues often encountered in transfer learning approaches. This thesis investigates the potential of self-supervised pre-training methods for DCNNs to enhance change detection performance with a limited amount of labeled SAS data. Compared to the specialization project, a larger amount of unlabeled SAS data is provided to facilitate the learning of the self-supervised auxiliary tasks. A multi-task DCNN architecture called Siam R-CNN will also be developed to simultaneously learn object-based bounding box change detection and pixel-based change map generation. The primary objective of employing a multi-task architecture is to leverage shared knowledge between the tasks, thereby enhancing performance when the amount of data is insufficient. Furthermore, combining bounding box regression and change map prediction allows for comprehensive comparisons with a wider range of traditional and deep learning-based methods. To further leverage the learned change map, an internal attention mechanism will be implemented within the network.

Experiments will be conducted to justify the benefits of self-supervised learning as a preliminary step to the change detection tasks. Additionally, these experiments will study the effects of employing a multi-task framework for SAS change detection. The results will be compared with the transfer learning approach employed in the specialization project to evaluate the effectiveness of self-supervised pre-training for SAS imagery. Moreover, a comparison with a well-established traditional change detection method will be performed to evaluate the potential of deep learning-based SAS change detection and identify possible future research directions.

1.2 Literature Review

1.2.1 SAS - Synthetic Aperture Sonar

The material in this section is primarily from the specialization project (Nyegaarden; 2022) and is included for the completeness of the thesis.

Sonar technology uses sound waves to detect and classify objects underwater. It works by emitting sound pulses and measuring the time it takes for the echoes to return. By analyzing the echoes, sonar systems can determine underwater objects' range, speed, and direction. This information can then be used to create a two-dimensional image

of the underwater environment. Synthetic aperture sonar (SAS) is a type of sonar technology with a much higher along-track resolution than conventional sonars. This is accomplished by effectively combining a number of successive pings from a moving platform to synthesize a long sonar transducer, up to many tens of meters in length (Callow et al.; 2012). As a result, synthetic aperture sonar has the potential to produce images that can provide centimeter resolution over hundreds-of-meter ranges on the seafloors. This makes SAS a suitable technique for imaging the seafloor for military applications such as searches for naval mines. Figure 1.1 displays the imaging concept of a SAS.

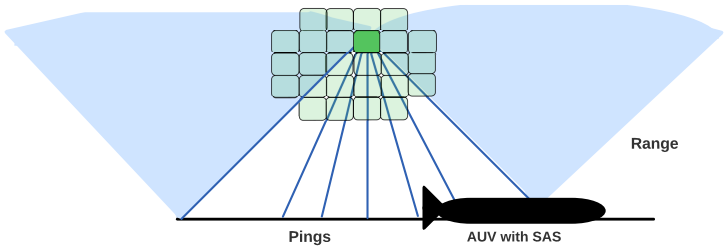


Figure 1.1: Synthetic Aperture Sonar Imaging Concept

Interferometry is a technique that leverages the interference of waves to extract valuable information. More specifically, it exploits the phase of the reflected signal to derive additional seafloor characteristics, such as elevation variations. Interferometric SAS merges interferometry with synthetic aperture technology to enhance bathymetric resolution by five to ten times compared to conventional side scan sonars (Callow et al.; 2012). The HISAS-1030 is an interferometric SAS sonar developed by Kongsberg Maritime in cooperation with the Norwegian Defence Research Establishment (FFI). It can provide detailed images with a high, range-independent resolution of about 3 cm, both along-track and across. Furthermore, the length of the synthetic antenna, and thus the number of integrated pings, increases with a range from the sonar to achieve a constant along-track image resolution (Kongsberg Maritime; 2022).

Successful SAS image processing depends on overcoming several challenges related

to the ocean environment and vehicle stability. One of the fundamental challenges in SAS seafloor imaging is surveying shallow waters, where the sea surface can cause multipath reflections that degrade the imaging quality. In order to overcome this challenge, specialized algorithms and processing techniques are needed to mitigate the effects of this interference. This can be particularly challenging for change detection-related tasks, where differences in sea surface roughness between the reference and repeat pass image can change the multipath contribution to the signal scattered back to the SAS receiver. The coherence difference between the two images can be captured by the aperture interferometer, where the sonar data processing has to be adjusted accordingly to optimize the quality of the images (Hansen et al.; 2011).

SAS bears a solid resemblance to synthetic aperture radar (SAR) that produce high-resolution images of the Earth's surface. However, while SAS technology is rapidly advancing, it is still relatively new and less developed compared to its counterpart in radar (Hansen; 2011).

1.2.2 Change Detection

Change detection (CD) is formally defined as "the process of identifying differences in the state of an object or phenomenon by observing it at different times" (Singh; 1989). It essentially involves quantifying relevant temporal effects using multi-temporal datasets. Change detection finds applications in various domains, including environmental monitoring, urban expansion, and disaster assessment (Singh; 1989).

In recent decades, various CD methods have been developed, including traditional and deep learning-based approaches. These approaches can generally be categorized into two groups based on how they represent changes: pixel-based CD (PBCD) and object-based CD (OBCD) (Shafique et al.; 2022). PBCD methods extract features from individual pixels and their surroundings and predict binary change masks that classify each pixel as changed or unchanged. OBCD, however, takes objects instead of pixels as the analysis unit. Most object-based methods rely on an object detection framework to find changed objects by representing them with bounding boxes. Object detection is one of the most fundamental problems in computer vision and has received significant attention in recent years. Unlike change detection, which quantifies changes between multiple images, object detection aims to detect all instances of an object within one image. In the OBCD approach, the "changed area" is considered as a detected object,

while the "unchanged area" is treated as the background (Jiang et al.; 2022).

Traditionally, the primary focus has been to perform binary change detection (BCD), where regions are classified as either changed or unchanged. However, with the advancements in earth observation satellite technology, remote sensing images now possess higher spatial resolution, leading to a growing interest in multi-class change detection (MCD) (Zhu et al.; 2022). However, MCD introduces additional complexity by considering multiple classes of changes, and most research continues to concentrate on improving the performance of BCD methods (Zhu et al.; 2022).

1.2.3 Traditional Change Detection Methods for SAS Imagery

The material in this section is from the specialization project (Nyegaarden; 2022) and is included for the completeness of the thesis.

Over the past decades, a large number of automatic change detection (ACD) methods have been developed that automatically detect changes in reference and repeat pass images by using image processing filters and algorithms. ACD approaches can be categorized based on the data level where the temporal matching occurs: decision or image. Decision-level methods match the output labels from a classifier operating independently on the reference and repeat pass image, where changes are detected as class transitions between the two data sets. On the other hand, image-level methods directly match new and old image data for regions or pixels (Midtgaard; 2013). Historically, ACD in searches for naval mine hunting has focused on decision-level methods based on the geographical association of mine-like objects due to challenges imposed by shortfalls of the traditional side-scan sonar (SSS) imagery. However, the emergence of SAS technology and Aided Inertial Navigation Systems (AINS) mounted on the AUV has facilitated a growing research activity on image-level SAS change detection methods (Midtgaard; 2013).

G-Michael et al. (2016) propose a complete image-based ACD method for SAS imagery. This method employs the scale-invariant feature transform (SIFT) algorithm and local phase-based co-registration as an image registration tool, followed by a canonical correlation analysis (CCA)-based change detection method. The CCA algorithm is a multivariate statistical method that determines the correlation between the reference and repeat-pass SAS images and extracts a subset of the most coherent change features from the two images (G-Michael et al.; 2016). The study demonstrates that coherent

change detection can be applied in a sandy shallow-water environment over a range of time scales from hours to several days, using co-registration tools and CCA that consider both phase and amplitude levels of the backscattered signals. The coherence maps indicated, however, that coherence between images co-registered with the SIFT algorithm degraded rapidly over time resulting in poor change detection in the signal phase. Midtgaard (2013) addresses the challenge of temporal decorrelation of coherent change detection methods, proposing an incoherent ACD method that successfully detects changes for time scales in the order of years. This complies with the operational demands of most applications. Incoherent methods only consider changes in the magnitude of the reflected sonar signals and do not use phase information. These methods are typically less sensitive to small changes in the scene but are more robust to errors in the coregistration process. Midtgaard (2013) applies a SURF algorithm to extract and describe the feature points in the images. The point pairs are then robustly matched and used to estimate the parameters for the affine transformation of the reference image onto the pixel coordinates of the repeat pass image. After that, pixel-wise subtraction of magnitude values produces a difference image. In order to extract only relevant changes, e.g., mine-like objects, from the decluttered difference image, the processing needs to take the specific characteristics of the target into account. As previously stated, Midtgaard (2018) suggests a matched filter combined with a multi-line fusion method to successfully filter out target-like fish responses and other irrelevant temporal changes.

However, selecting a threshold and criteria to capture all change regions while eliminating undesired ones remains challenging, especially if the signal-to-noise ratio (SNR) is low. Moreover, incorporating domain-specific knowledge and constraints is a tiresome and time-consuming procedure that does not produce high detection performance for new datasets (Shafique et al.; 2022). In recent years, deep learning methods for remote sensing images automatically derive and extract complicated non-linear features, overcoming several limitations of the traditional change detection methods. For SAS imagery, however, the traditional methods have demonstrated the best results to date, as the potential of deep learning is highly reliant on the availability of a large dataset.

1.2.4 Deep Learning-Based Change Detection

Parts of the material in this section are from the specialization project (Nyegaarden; 2022), specifically subsection 1.2.4.1, 1.2.4.3, and 1.2.4.4, and are included for the completeness of the thesis.

Recently, remote sensing (RS) platforms have made significant technological advancements, becoming capable of collecting a broader range of high-quality RS data. Different kinds of satellites have been launched into space, providing valuable data describing land use and land cover and how they vary over time and space (Shafique et al.; 2022). Due to the increased accessibility of satellite data, researchers have made significant efforts to apply deep learning techniques for remote sensing images. However, the literature on deep learning change detection is scarce compared to semantic segmentation and object detection tasks. Involving multiple images multiplies the problems and limitations (e.g., noise) associated with single-image information extraction. Furthermore, change detection aims to extract the minority pixels that have experienced changes, easily confusing the change information with noise. However, the number of publications on DL-based change detection for remote sensing images is increasing rapidly due to the urgent need for high-accuracy change detection applications (Bai et al.; 2022). The most popular deep learning neural networks used in these studies are Deep Convolutional Neural Networks (DCNNs), Auto-Encoders (AEs), Deep Belief Networks (DBNs), Recurrent Neural Networks (RNNs), and Generative Adversarial Networks (GANs) (Bai et al.; 2022). DCNNs have been extensively applied for change detection due to their ability to effectively process and analyze high-dimensional information. The remainder of this section will consider the DCNN architecture, as the structures of the other abovementioned neural networks are beyond the scope of this thesis.

1.2.4.1 Deep Convolutional Neural Networks (DCNNs)

Figure 1.2 illustrates a general DCNN architecture. A DCNN typically consists of multiple layers, each performing a different function in extracting features from the input data. These layers can be broadly grouped into convolutional, pooling, and fully connected (FC) layers. The convolutional layer is the core building block of the network. This layer applies a convolutional operation to the input data, using a set of filters (kernels)

to detect specific data features. The output of the convolutional layer is a feature map representing the extracted features. The earlier layers perform simple operations such as edge detection and color blurring; however, the following layers aggregate these features as shapes or objects (Khelifi and Mignotte; 2020). The pooling layers are used to down-sample and reduce the feature maps' spatial dimensions, decreasing the network's parameter number. The most common type of pooling is max-pooling, in which the maximum value in a small region of the feature map is taken and used as the output of the pooling operation. The fully connected layers extract more high-level information by reshaping the feature maps and combining them in a way that allows the DCNN to predict the label of the input image. The network's last layer, the classification layer, outputs feature maps corresponding to the number of classes to be predicted (Khelifi and Mignotte; 2020). These logits can be passed through a mapping such as the softmax function to produce a probability distribution over the classes. Different successful DCNN architectures have been suggested in the literature, such as AlexNet, VGGNet, ResNet, DenseNet, and U-Net (Jiang et al.; 2022). These architectures explore new and innovative methods for constructing the convolutional layers to achieve more efficient learning and higher accuracies customized to the problem to be solved.

The overall performance of the change detection method primarily depends on the

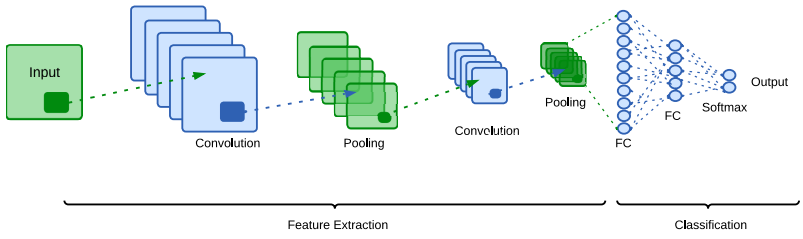


Figure 1.2: General DCNN Architecture

feature extraction ability of the DCNN. The feature extraction strategy can be divided into respectively single-branch and dual-branch structures. The single-branch structure is an early fusion (EF) strategy that fuses the reference and repeat-pass images before feeding them into the network to extract the features. For example, for RGB images with three channels (red, green, blue), the input matrix would have the form $6xWxH$,

where W and H denote the width and height of the image, respectively. The dual-branch structure is a late fusion strategy in which features are extracted from the fused results of two independent branches. This approach is known as a Siamese neural network, where the two branches extract features from both inputs separately with the same structure and shared weights, being merged after the convolutional layers of the network are completed (Jiang et al.; 2022). For RGB images with three channels, the DCNN is fed with two input matrices of shape $3xWxH$.

1.2.4.2 Region-Based Convolutional Neural Networks (R-CNNs)

Proposal-based methods have attracted much interest in object detection research in recent years. These methods usually exploit fast measurements to test whether a sampled window is a potential object. They further pass these object proposals to more sophisticated detectors to determine whether they are background or belong to a specific class. Some of the most well-known object detectors that rely on proposals are R-CNNs. R-CNN stands for region-based convolutional neural network, which was initially introduced by Girshick et al. (2014). The concept behind R-CNN involves generating a large number of region proposals, extracting features from each region using a pre-trained CNN, and then classifying each proposed region into several object categories using a linear support vector machine (SVM). The R-CNN improved the mAP (mean average precision) score by more than 30% relative to the previous best result on the benchmark PASCAL VOC dataset (Girshick et al.; 2014). Several advancements have been developed to enhance two-stage R-CNNs, including Fast R-CNN (Girshick; 2015) and Faster R-CNN (Ren et al.; 2016). These improvements address some of the limitations of the original R-CNN while enhancing both speed and accuracy. Fast R-CNN introduces a region of interest (ROI) pooling layer, enabling the network to efficiently compute shared information across different region proposals. It generates a single feature map for the entire image and utilizes the ROI pooling layer to extract a feature vector for each proposed region (Girshick; 2015). Faster R-CNN introduces a region proposal network (RPN) that shares convolutional features with the detection network and achieves further speed-up. To date, Faster R-CNN is the most widely used version of the R-CNN family, being the most efficient and accurate of the three. Figure 1.3 illustrates the general Faster R-CNN architecture. Region-based convolutional neural networks have primarily been evaluated in object detection and classification tasks, with little research investigating

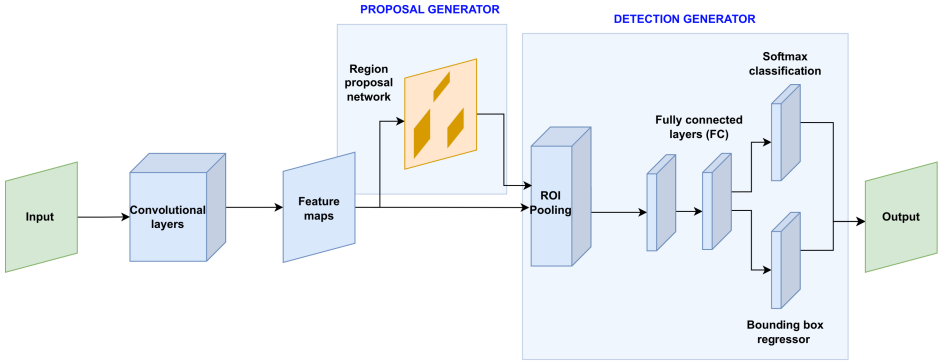


Figure 1.3: Faster R-CNN Architecture

the effectiveness of R-CNNs for change detection. However, Wang et al. (2018) applies the Faster R-CNN algorithm to detect changes in high-resolution remote sensing images, where the changed regions are considered the object to be detected. Their findings demonstrate that their proposed method achieves competitive performance compared to other deep learning-based change detection techniques. Moreover, the authors assess the dual-branch structure, as described in Section 1.2.4.1, and demonstrate that it can lead to better performance than the single-branch structure when used as input to the Faster R-CNN algorithm.

1.2.4.3 Optimization Strategy

An important consideration when implementing a DL-based neural network is selecting an appropriate loss function. The loss function calculates the error between the predicted output from the network and the ground truth output. Binary cross-entropy loss is a commonly used loss function for binary change detection tasks as it measures the similarity between two probability distributions (Jiang et al.; 2022). However, class imbalance is common when using DL-based change detection methods, where the ratio of unchanged and changed pixels may be heavily skewed. Training with very few changed pixels causes the network to predict the majority class of unchanged pixels regardless of their ground truth label. Weighted cross-entropy, focal loss, and dice loss are loss functions proposed to resolve the imbalance problem. The loss functions can also be combined to take advantage of their complementary strengths (Jiang et al.; 2022).

An appropriate optimizer is also essential when implementing a deep neural network. The optimizer's role is to update the weights and biases of the network in such a way that it minimizes the loss function. Among various optimization algorithms available, gradient descent is widely used and the most common method for optimizing neural networks. Gradient descent, also known as steepest descent, minimizes an objective function $J(\theta)$ by updating the parameters in the opposite direction of the gradient of the objective function with respect to its parameters $\nabla_\theta J(\theta)$. The learning rate n determines the size of the step size taken to approach a minimum (Murugan and Durairaj; 2017). Different variants of gradient descent have been developed, including batch gradient descent and stochastic gradient descent (SGD). In batch gradient descent, the parameters θ_i are initialized and updated according to the following procedure:

$$\theta_i = \theta_i - n\frac{\partial}{\partial\theta_i}L(\theta_i : (y_i, \hat{y_i})), \tag{1.1}$$

where $\hat{y_i}$ and y_i are the predicted and actual values, respectively. Stochastic gradient descent addresses the inefficiency of computing gradients for the entire training set by updating the parameter θ sequentially with each iteration (Murugan and Durairaj; 2017). While gradient descent methods are widely used and effective for training neural networks, they pose several challenges. Firstly, these methods can become trapped in numerous suboptimal local minima since they uniformly scale the gradient in all directions. Secondly, selecting an appropriate learning rate is crucial for convergence but can be challenging to tune, especially in complex and large-scale neural networks (Ruder; 2016). To address these challenges, researchers have proposed adaptive methods that adjust the gradients based on the function's curvature estimates. One notable approach is Adam (Adaptive Moment Estimation), which employs a set of learning rates, individually assigned to each parameter, dynamically adapting them as the training advances (Kingma and Ba; 2015). The Adam update equation can be expressed mathematically as:

$$\theta_{i+1} = \theta_i - \frac{n}{\sqrt{\hat{v}_i} + \epsilon}\hat{m}_i, \tag{1.2}$$

where $\hat{v}_i$ and $\hat{m}_i$ are bias-corrected first and second-moment estimates of the gradient that are defined as:

$$\hat{m}_i = \frac{m_t}{1 - \beta_1^t} \tag{1.3}$$

$$\hat{v}_i = \frac{v_t}{1 - \beta_2^t}. \tag{1.4}$$

Here, m_t and v_t are estimated by the following formula:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \quad v_t = \beta_2 v_t + (1 - \beta_2) g_t^2, \tag{1.5}$$

where g_t is the gradient of the loss function with respect to parameter θ. The default values suggested by the authors for Adam are $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ (Kingma and Ba; 2015). Adam has been used in many applications due to its competitive performance over other optimization strategies, faster training speed for larger datasets, and ability to achieve good results with minimal tuning. However, recent research indicates that Adam may result in poorer generalization performance than SGD when training deep neural networks for image classification tasks (Keskar and Socher; 2017). To address these concerns, AdamW has been introduced by Loshchilov and Hutter (2017) as an extension of the Adam optimizer. AdamW incorporates weight decay directly into the optimization algorithm, leading to improved generalization performance compared to Adam. Weight decay refers to a regularization technique that reduces the magnitudes of the model's weights during training to prevent overfitting.

1.2.4.4 Learning Technique

Deep learning-based approaches are typically classified based on their learning technique and the availability of labeled or unlabeled training data (Khelifi and Mignotte; 2020). Supervised methods solve the problem by learning from a labeled dataset and can be highly effective when trained on large amounts of labeled data. The use of supervised learning methods in remote sensing change detection is rapidly increasing as remote sensing data becomes more readily available. Supervised DCNNs have demonstrated superior performance to other state-of-the-art change detection methods (Khelifi and Mignotte; 2020). U-Net is considered one of the standard CNN architectures for supervised learning. The architecture of U-Net is symmetrical, consisting of an encoder that extracts spatial features from an input image and a decoder that generates a segmentation map from the encoded features. Jaturapitpornchai et al. (2019) suggested in detail a CD-based U-Net that detects building constructions using two SAR images captured at different times. Subsequently, studies have modified the U-Net architecture in different

ways to achieve better results and higher accuracy. Amongst these are U-Net++ (Peng et al.; 2019) and Siamese-Nested U-Net (Li et al.; 2020) that will be reviewed in Section 2.2.3.

As previously stated, insufficient datasets for training change detection models often make achieving valuable results with an end-to-end supervised deep learning method difficult. Furthermore, creating ground truth maps that accurately reflect the changes in ground objects can cost lots of time and effort (Khelifi and Mignotte; 2020). Therefore, in many cases, it is more efficient to extract features from the images in an unsupervised manner (Chen and Shi; 2020). Unsupervised deep learning is a technique where the model is trained without using labeled data. In other words, the model must discover the underlying structure of the data on its own without feeding the network with any labels or ground truth images. In recent years, numerous unsupervised deep learning CD approaches for SAR datasets have been proposed due to the limited amount of publicly available labeled training data. Wang et al. (2020) propose an unsupervised SAR-image change detection framework, using hypergraphs to capture local grouping information and generate a difference image using a noise-insensitive partition technique.

In addition, it has become popular to conduct pre-training on established benchmark datasets before fine-tuning deeply learned models for downstream tasks that lack sufficient labeled data. This practice is known as transfer learning. However, the effectiveness of supervised pre-training heavily relies on the similarity between the source and target data domains, where a good pre-trained model renders well on a similar dataset but is not as useful on a different one.

1.2.5 Self-Supervised Learning

Recently, self-supervised learning (SSL) has raised considerable attention in the field of computer vision (Wang et al.; 2022). Self-supervised learning is a technique that exploits unlabeled data to acquire valuable information. Instead of relying on manual annotation, it employs a self-produced objective, self-supervision, to train the model. By exploiting a large amount of unlabeled data, the model can learn to capture high-level representations of the input data. These learned representations can then be transferred to downstream tasks for real-world applications. Figure 1.4 illustrates the general pipeline of self-supervised learning. As opposed to supervised pre-training (transfer learning), self-supervised pre-trained models can leverage more general representations

and mitigate the shortcomings of supervised learning. Specifically, self-supervised pre-training eliminates the need for human annotation, achieves good performance on downstream tasks with only a small number of labels, and can reduce the domain gap between the pre-training and downstream datasets by collecting unlabeled data from the target application (Wang et al.; 2022). Self-supervised learning is often classified

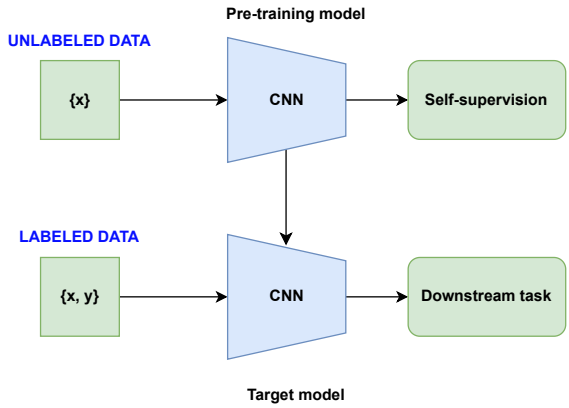


Figure 1.4: Self-Supervised Learning Pipeline

into three categories: generative, contrastive, and predictive. The generative approach involves learning to reconstruct or generate input data, while the predictive method involves learning to predict self-generated labels. Finally, contrastive methods aim to maximize the similarity between semantically identical inputs.

Generative Methods. Generative methods, such as autoencoders (AE) and generative adversarial networks (GAN), learn representations by reconstructing the input data. An *autoencoder* is a feed-forward encoder-decoder network that encodes the input x into a compressed representation $z = E(x)$ and then decodes it back to a reconstructed input $x = D(z)$, aiming to match the original input. This model type is commonly trained using patches as input and ground truth targets. Autoencoders have been widely employed to learn representations from various remote sensing data. For example, Peng et al. (2021) proposed a self-supervised autoencoder that generates patch-wise change-maps for flood mapping using bi-temporal multispectral satellite imagery. Implementing the self-supervised autoencoder demonstrated superior performance compared to traditional change detection methods (Peng et al.; 2021).

Generative adversarial networks consist of a generator and a discriminator. The discriminator is trained to differentiate between real samples from the training dataset and synthetic samples generated by the generator (Wang et al.; 2022). Currently, limited research exists exploring the use of GAN-based methods for self-supervised pre-training.

Predictive Methods. A predictive method involves pre-training a model using self-generated labels. This approach starts by designing a suitable pretext task for the dataset, preparing self-generated labels, and then training the model to predict these labels (Wang et al.; 2022). Pretext tasks allow the model to learn valuable feature representations that can later be applied to downstream tasks. However, designing an appropriate pretext task requires domain-specific knowledge, and even though it enhances performance on a specific downstream task, it may not generalize well to other task domains.

Leenstra et al. (2021) suggested a spatial and temporal pretext task for enhancing change detection in Sentinel-2 imagery. They pre-trained a CNN on a binary classification pretext task to discriminate between overlapping and non-overlapping patches. The underlying hypothesis was that by doing so, the model would learn to disregard irrelevant radiometric variations and instead focus on the relevant spatial differences between the patches. The paper by Jian et al. (2022) addressed the challenge of detecting small objects within the Faster R-CNN framework by introducing a self-supervised pretext task called "CutPaste." The main idea behind this method was to generate negative samples by cutting out and enhancing certain regions from object-free images and subsequently pasting these rectangles back onto the image. The auxiliary learning task focused on detecting the number of cut-and-paste rectangles in the image. Experimental results demonstrated that utilizing the CutPaste pretext task significantly improved performance, with a 17.8% increase in mean Average Precision (mAP) and a 22.8% improvement in detection accuracy.

Contrastive Methods. The performance of predictive self-supervised methods relies on the design of a well-suited pretext task, which often poses significant challenges. To overcome this issue, contrastive methods adopt a training approach where semantically identical inputs are contrasted and encouraged to be closely positioned in the representation space (Wang et al.; 2022). However, this emphasis on similarity can sometimes result in a trivial solution known as model collapse. In order to address this problem, several solutions have been proposed in the literature, including techniques

such as negative sampling, clustering, and redundancy reduction. These approaches aim to alleviate the issue of model collapse and enhance the performance of contrastive self-supervised learning methods.

Negative sampling involves including dissimilar samples to create positive and negative input pairs. The idea behind this approach is for the model to learn valuable representations by bringing augmented versions of the same sample closer while pushing away embeddings from different samples. For example, Jean et al. (2018) employed contrastive learning by training a CNN on a triplet of tiles. Each triplet consisted of an anchor tile t_a, a neighbor tile t_n, and a distant tile t_d. The training objective minimized a triplet loss, which aimed to minimize the distance between the anchor and neighbor tiles while maximizing the distance between the anchor and distant embeddings. Leenstra et al. (2021) proposed a pretext task for Sentinel-2 change detection by embedding a similar strategy, where each triplet consisted of two patches with spatial overlap and a third patch without overlap.

Redundancy reduction is a technique inspired by neuroscientist H. Barlow's redundancy-reduction principle. Its primary aim is to preserve essential information by minimizing redundancy. Zbontar et al. (2021) introduced Barlow Twins, which employs an objective function that measures the cross-correlation matrix between the outputs of two identical networks fed with distorted versions of a sample. The objective is to make the cross-correlation matrix as close to the identity matrix as possible. As a result, the embedding vectors of the distorted sample versions become more similar, effectively reducing redundancy among their components. The loss function for Barlow Twins can be defined as follows:

$$\mathcal{L}_{BT} = \underbrace{\sum_i (1 - C_{ii})^2}_{\text{Invariance term}} + \underbrace{\lambda \sum_i \sum_{j \neq i} C_{ij}^2}_{\text{Redundancy reduction term}}, \tag{1.6}$$

where λ is a trade-off constant and C is the cross-correlation matrix computed between the representations of the input image pairs along the batch dimension:

$$C_{ij} = \frac{\sum_b z_{b,i}{}^A z_{b,j}^B}{\sqrt{\sum_b (z_{b,i}^A)^2} \sqrt{\sum_b (z_{b,j}{}^B)^2}}. \tag{1.7}$$

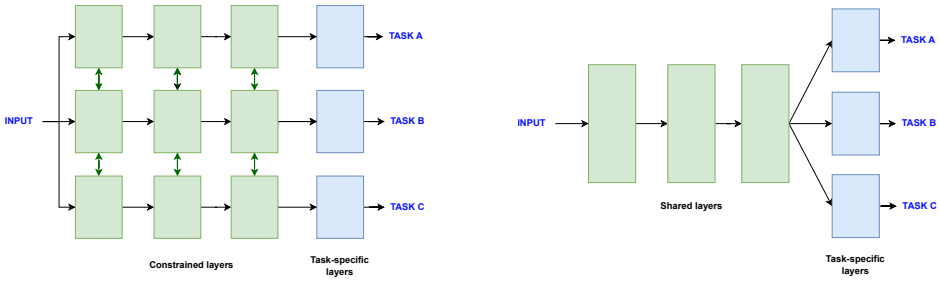
Here, b represents the batch sample, A and B denote two different views, while i and j index the vector dimensions of the network output. For scene change detection (SCD), Ramkumar et al. (2021) propose a self-supervised pre-training method based on Barlow Twins. The approach involves feeding a patch pair T_0 and T_1 representing a scene at two different times into a Siamese encoder trained using the objective function in Equation 1.6. The invariance term makes the image pair invariant to noisy changes, such as seasonal variations, while the redundancy reduction term aligns the representations of the input pairs to be similar. Ramkumar et al. (2022) propose a differencing-based Barlow Twins implementation that uses feature differencing to learn discriminatory representations corresponding to the changed regions of the scene. Instead of maximizing the cross-correlation of the transformed views of the same image to approximate the identity matrix, this method maximizes the difference representations $(d1, d2)$ between an image pair captured at different times in feature space. This approach enhances the downstream change detection performance by 2% compared to standard Barlow Twins pre-training.

1.2.6 Multi-Task Learning

Multi-task learning (MTL) is a learning paradigm in machine learning that aims to jointly solve a set of prediction problems by sharing information across tasks. The basic assumption of MTL is that all tasks are associated. Thus the knowledge contained in one task can be leveraged by other tasks, leading to improved generalization performance for all tasks. MTL is particularly useful in addressing the data scarcity problem, where the number of labeled data for each task is insufficient to train an accurate learner (Zhang and Yang; 2017). By aggregating labeled data across all tasks, MTL helps reuse existing knowledge and reduces the cost of manually labeling the input. Additionally, by sharing layers between tasks, MTL has the potential to substantially reduce memory usage and increase inference speed. Most importantly, MTL can improve performance if the associated tasks share complementary performance or act as a regularizer for one another (Vandenhende et al.; 2020). By jointly optimizing the model parameters for multiple tasks, the model aims to perform well across all tasks instead of overfitting to individual tasks.

A general architecture for MTL is depicted in Figure 1.5. In soft parameter sharing, each task is allocated its own set of parameters, while feature-sharing mechanisms

facilitate communication between tasks. Hard parameter sharing involves dividing the parameter set into shared and task-specific operations. A typical hard parameter sharing design includes a shared encoder that branches out to task-specific decoder heads (Vandenhende et al.; 2020).



(a) Soft Parameter Sharing (b) Hard Parameter Sharing

Figure 1.5: Multi-Task Learning Architecture

Deep multi-task networks have demonstrated their effectiveness in various task combinations, such as detection and classification, detection and segmentation, and segmentation and depth estimation (Vandenhende et al.; 2020). The Faster R-CNN architecture detailed in Section 1.2.4.2 employs a hard parameter-sharing design to jointly learn the tasks of object detection and classification for bounding box recognition. Mask R-CNN is an extended version of Faster R-CNN, introduced by He et al. (2017). It incorporates a segmentation branch for predicting an object mask in parallel with the existing branch responsible for bounding box prediction. The results demonstrate a significant improvement of 0.9 points box average precision (AP) compared to the original Faster R-CNN, solely due to the benefits of multi-task training.

1.2.6.1 Multi-Task Optimization

Training multiple tasks while simultaneously learning a shared representation presents several challenges compared to standard single-task learning. Firstly, the joint learning of multiple tasks is prone to negative transfer if the task selection contains unrelated tasks. Secondly, a significant challenge in MTL is the "task balancing" problem, which involves balancing the joint learning of all tasks to prevent a scenario where one or more of the tasks have a dominant influence on the network weights (Vandenhende

et al.; 2020) The general optimization objective in an MTL problem can be formulated as follows:

$$\mathcal{L}_{MTL} = \sum_i w_i \cdot \mathcal{L}_i, \tag{1.8}$$

where w_i and $\mathcal{L}_i$ correspond to the task-specific weights and loss functions. When using the stochastic gradient descent to minimize the objective function, the shared network weights W_{sh} are updated by the following formula:

$$W_{sh} = \mathcal{W}_{sh} - \gamma \sum_i w_i \frac{\partial \mathcal{L}}{\partial W_{sh}}. \tag{1.9}$$

Equation 1.9 indicates that the weight update may not be optimal when the task gradients conflict or when one task dominates due to a significantly higher gradient magnitude compared to the other tasks (Vandenhende et al.; 2020). In most of the existing literature on multi-task learning, manually tuning the task-specific weights w_i in the loss function has been the standard approach, requiring significant effort and time investment. However, recent advancements have introduced systematic methods to optimally balance the task-specific weights, such as uncertainty weighting (Cipolla et al.; 2018), gradient normalization (Chen et al.; 2018), and dynamic weight averaging (Liu et al.; 2019).

Uncertainty Weighting. The technique proposed by Cipolla et al. (2018) weights multiple loss functions by considering the homoscedastic uncertainty associated with each task. The homoscedastic uncertainty captures the relative confidence levels across different tasks, considering the inherent uncertainty in both regression and classification tasks. Specifically, the model weights $\mathcal{W}$ and standard deviations σ_1 and σ_2 are optimized to minimize the following equation:

$$\mathcal{L}(W, \sigma_1, \sigma_2) = \frac{1}{2\sigma_1^2}\mathcal{L}_1(W) + \frac{1}{2\sigma_2^2}\mathcal{L}_2(W) + \log \sigma_1 \sigma_2. \tag{1.10}$$

Equation 1.10 reveals that the impact of a task on the network weight update is smaller when the task's homoscedastic uncertainty is high. This property is advantageous when dealing with noisy annotations since the task-specific weights will automatically be lowered for such tasks (Vandenhende et al.; 2020).

Gradient Normalization. Chen et al. (2018) addresses the task-balancing problem by stimulating the task-specific gradients to be of similar magnitude. The Gradient

Normalization method (GradNorm) aims to balance two properties during the training of multi-task networks: the gradient magnitudes G_i^W and the learning pace $r_i(t)$ for different tasks. The gradient magnitude G_i^W represents the L2 norm of the gradient for the weighted single-task loss $w_i \cdot L_i(t)$ at step t with respect to the weights W. $r_i(t)$ denotes the inverse training rate. GradNorm is implemented as an L1 loss function $\mathcal{L}_{grad}$ between the actual and target gradient norms for each task at every timestep. This loss is then summed over all tasks, resulting in the following equation:

$$\mathcal{L}_{grad}(t; w_i(t)) = \sum_i |G_i^W(t) - \bar{G}^W(t) \cdot [r_i(t)]^\alpha|, \tag{1.11}$$

where $\bar{G}^W(t)$ represent the mean task gradient with respect to weights W at step t, and α is an additional hyperparameter (Chen et al.; 2018). The parameter α enables adjustment of the training rate balance, with a higher value indicating that tasks have very different learning dynamics. In practice, during training, the task-specific weights w_i are updated using backpropagation in each iteration, which may introduce additional computational time and architectural complexity to the network.

GradNorm demonstrates remarkable performance when evaluated on the NYUv2 dataset carrying depth, surface normals, and segmentation labels for diverse indoor scenes (Chen et al.; 2018). Specifically, GradNorm reduces the depth error by $\sim 5\%$ compared to the approach that assigns equal weights. Additionally, the GradNorm implementation surpasses the performance of uncertainty weighting on the NYUv2 dataset (Chen et al.; 2018).

Dynamic Weight Averaging. Similar to GradNorm, dynamic weight averaging (DWA), proposed by Liu et al. (2019), aims to balance the learning pace of multiple tasks. However, DWA takes a different approach by solely relying on task-specific loss values and eliminates the need for separate backward passes to compute task-specific gradients during training. In DWA, the task-specific weights w_i for task i are defined as:

$$w_i(t) = \frac{N \exp(r_i(t-1)/T)}{\sum_n \exp(r_n(t-1)/T)}, \quad r_n(t-1) = \frac{\mathcal{L}_n(t-1)}{\mathcal{L}_n(t-2)}, \tag{1.12}$$

where r_i represents the relative descending rate and N denotes the number of tasks. The temperature T controls the softness of the task weighting in the softmax operator. When the loss of a particular task decreases slower than other tasks, the task-specific

weighting is increased accordingly.

It is important to note that DWA requires balancing the overall loss magnitudes beforehand to prevent specific tasks from dominating others during training. GradNorm avoids this problem by simultaneously balancing the training rates and gradient magnitudes through a single objective (Vandenhende et al.; 2020).

1.2.7 Attention Mechanisms

An attention mechanism enables a model to focus on specific parts of the input data, drawing inspiration from the human cognitive process of selectively attending to relevant information (Guo et al.; 2022). In computer vision tasks, an attention mechanism can be considered a dynamic selection process that adaptively weights characteristics based on their relevance to the input. Attention mechanisms have proved successful in various visual tasks such as image classification, object detection, and semantic segmentation. Existing attention methods can be categorized into four types: channel attention (identifying what to pay attention to), spatial attention (determining where to pay attention), temporal attention (deciding when to pay attention), and branch attention (selecting which branch to pay attention to) (Guo et al.; 2022). The remainder of this section will focus on spatial attention as the other categories lie beyond the scope of this thesis.

Spatial attention can be seen as an adaptive mechanism that selectively attends to specific spatial regions by assigning attention weights to different locations (Guo et al.; 2022). This allows the network to gather information from a broader context, indirectly expanding the receptive field of the neural network. The Faster R-CNN architecture, detailed in Section 1.2.4.2, employs the RPN module as an attention mechanism for proposing relevant feature regions. In addition, the authors of SpotNet (Perreault et al.; 2020) take advantage of the multi-task architecture designed for object segmentation and detection, incorporating an internal attention mechanism. They utilize the predicted segmentation map as an attention map multiplied by the raw feature maps to enhance bounding box regression and classification performance. Similarly, in the study by Pang et al. (2019), a mask-guided attention network is proposed for occluded pedestrian detection. They introduce a mask-guided attention module (MGA) that generates modulated features by multiplying the extracted region of interest (ROI) features with a spatial attention mask. This attention mask is constructed using visible body region

information to improve pedestrian detection in challenging scenarios where occlusions are present.

1.3 Contributions

The main contributions of the work presented in this thesis are as follows:

- A review of multi-task and self-supervised learning techniques that have demonstrated superior performance in their respective computer vision domains.
- Development and implementation of Siam R-CNN, a novel multi-task attention-based deep convolutional neural network specifically designed for change detection in SAS imagery.
- Implementation of self-supervised pre-training techniques to improve the downstream change detection performance of Siam R-CNN.
- Experimental testing and optimization of Siam R-CNN for detecting changes in SAS imagery. In particular, the significance of applying the self-supervised pre-training techniques is evaluated.
- An analysis of the experimental results, along with suggestions for future research to further enhance the performance of the proposed approach.

1.4 Outline

The report is organized as follows. Chapter 2 describes the system used for change detection, including an in-depth description of the pretext and change detection tasks, as well as their corresponding architectural design. Chapter 3 presents the experiments and their results for the self-supervised pretext task on an unlabeled dataset and the change detection performance when applied to a smaller labeled SAS dataset. In Chapter 4, the report concludes by exploring potential avenues for future research and offering final thoughts.

Chapter 2

Methodology

This chapter presents the system and methods used for performing change detection in SAS images. First, it describes the change detection pipeline to provide a high-level view of the proposed system. Following this, a detailed and well-justified presentation of the chosen architecture design for both the change detection and pretext tasks is provided.

2.1 Change Detection Pipeline

A multi-task attention-based learning model is designed for change detection in this thesis. To address the limited size of the SAS dataset, a self-supervised pre-training method is employed to extract informative features that can be effectively applied to the downstream change detection task. The overall pipeline comprises two stages. In the first stage (Section 2.3), the feature extractor network is trained on a predefined pretext task using an unlabeled SAS dataset. In the second stage (Section 2.2), the pre-trained weights obtained from the first stage are used to fine-tune the change detection network through supervised training. An overview of the methodology is visualized in Figure 2.1.

Figure 2.1: Change Detection Pipeline

2.2 Change Detection Task Architecture

2.2.1 Siam R-CNN

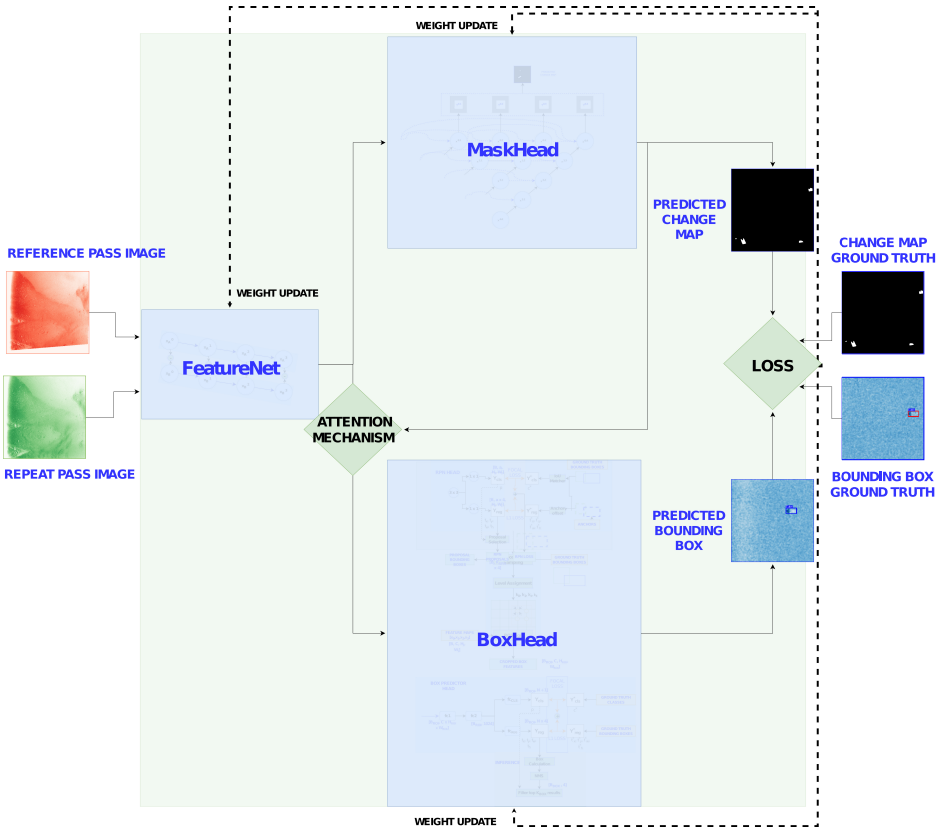


Figure 2.2: Siam R-CNN Architecture

Figure 2.2 displays the Siam R-CNN architecture, designed to train the network to perform multiple tasks simultaneously. The network is specifically trained to predict both bounding boxes corresponding to the changes and a pixel-level change map. The primary objective of implementing a multi-task architecture is to leverage shared knowledge across the tasks, effectively enhancing change detection performance with limited labeled SAS data. Additionally, the combination of bounding box regression

and change map prediction allows for comparison with a broader range of traditional and deep learning-based change detection methods. Furthermore, an internal attention mechanism is incorporated within the network to further leverage the learned change map.

To ensure clarity, it is important to differentiate between the convolutional backbone architecture used for feature extraction, FeatureNet, and the network heads responsible for bounding box recognition and mask prediction, BoxHead and MaskHead. The following paragraphs provide a detailed explanation of these three main modules, along with the attention mechanism and loss functionality, as depicted in Figure 2.2.

2.2.2 FeatureNet

FeatureNet is the convolutional backbone network and feature extractor of the Siam R-CNN architecture. Figure 2.3 illustrates the architecture of FeatureNet. The network adopts a Siamese structure, where the encoding layers are divided into two parallel streams with shared weights. The reference and repeat-pass images are inputted to the two encoder streams and processed through convolutional units and down-sampling modules to generate multi-scale feature maps. Finally, the encoder streams are merged by concatenating the absolute value of the difference between the branches. Siamese networks utilize shared weights to activate the same region in the feature maps of the two images during feature extraction. This allows for a comparison of location information, which is particularly relevant for change detection tasks. FeatureNet is inspired by the encoder from the Siamese Nested U-Net (Li et al.; 2020), which combines the strong feature extraction capabilities of U-Net++ with a Siamese branch to emphasize the specificity of the change detection task. The results obtained from the specialization project (Nyegaarden; 2022) demonstrate that incorporating a Siamese branch into the U-Net++ architecture significantly improved the change detection F1-score from 0.217 to 0.332. This improvement justifies choosing FeatureNet as the backbone network for the SAS change detection task.

The multi-scale feature extraction hierarchy enables the network to capture features at multiple levels of abstraction. This architectural design, referred to as a "feature pyramid network," is commonly employed in the context of object detection. The primary benefit of extracting features from each level of an image pyramid is that it generates a multi-scale feature representation wherein all levels possess robust semantic

information, including the high-resolution levels (Lin et al.; 2017). Hence, it is reasonable to assume that this will benefit the change detection task as well.

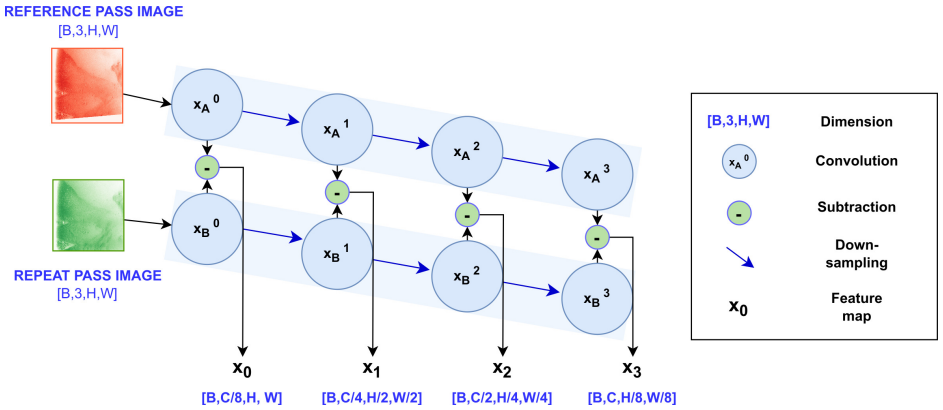


Figure 2.3: FeatureNet Architecture

The stack of feature maps, denoted as X_i, can mathematically be expressed as:

$$X_i = |(x_A^i - x_B^i)|. \tag{2.1}$$

Here, i indexes the down-sampling layer, and x_A^i and x_B^i are defined as:

$$x_A^i = \mathcal{P}(\mathcal{H}(x_A^{i-1})), \quad x_B^i = \mathcal{P}(\mathcal{H}(x_B^{i-1})). \tag{2.2}$$

The function $\mathcal{H}(\cdot)$ denotes a 3×3 convolution operation, and $\mathcal{P}(\cdot)$ denotes a 2×2 max pooling operation used for down-sampling. Equation 2.1 shows that the feature maps receive inputs from the Siamese subtraction operation of the two convolution units at the same level. The convolution units are implemented as residual blocks based on the award-winning deep learning architecture ResNet (He et al.; 2016) to facilitate better convergence abilities for the deep network. Figure 2.4 illustrates the ResNet block in FeatureNet, which is identical to all other convolutional units in the Siam R-CNN architecture. Figure 2.4 has been derived from the specialization project (Nyegaarden; 2022).

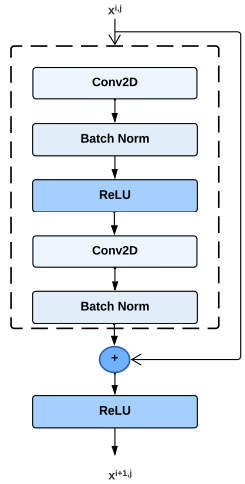


Figure 2.4: ResNet Convolution Unit

2.2.3 MaskHead

The MaskHead serves as a decoder structure and is utilized to restore the spatial dimensions of the input images to generate a predicted change map. The proposed MaskHead architecture, as illustrated in Figure 2.5, takes feature maps from FeatureNet as input to the decoder. When combined with FeatureNet, it forms a Siamese U-Net++ encoder-decoder architecture, which was demonstrated to be effective for SAS change detection in the specialization project (Nyegaarden; 2022). The U-Net++ architecture, first proposed by Zhou et al. (2020), introduces re-designed skip pathways and deep supervision to address some of the limitations of the original U-Net architecture. The decoder in the U-Net++ architecture consists of multiple stages that perform up-sampling of the feature maps from FeatureNet, thereby increasing their resolution. Additionally, the decoder receives skip connections from the corresponding encoding stages, allowing for the fusion of features at multiple scales. The decoder outputs two logits for each pixel: one representing the probability that the pixel has changed and the other representing the probability that the pixel has not been changed. The final change map is obtained by applying a softmax function to the logits and selecting the class with the highest probability for each pixel.

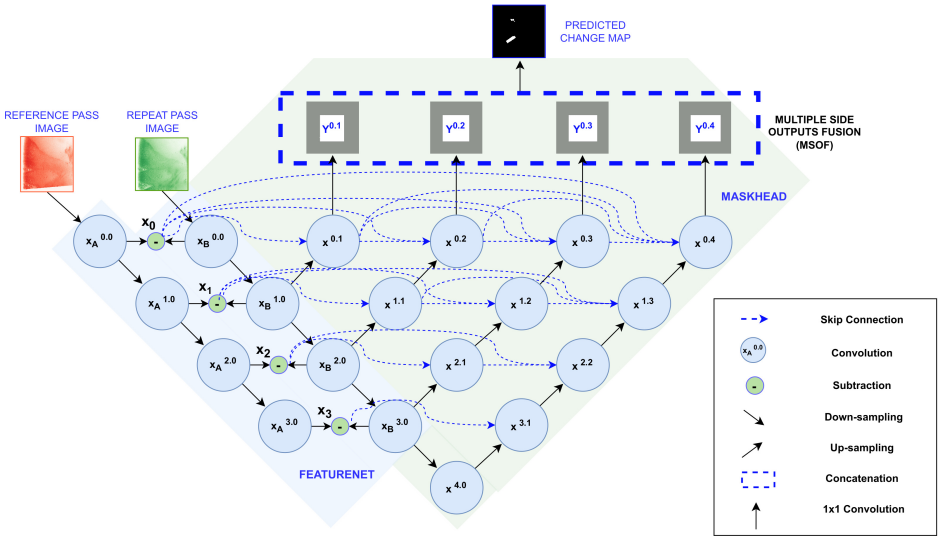


Figure 2.5: MaskHead Architecture

The deep supervision strategy MSOF (multiple sides outputs fusion) proposed by Peng et al. (2019) has been implemented in the presented MaskHead architecture, where multi-level feature information from all side-output layers $\{Y^{0,1}, Y^{0,2}, Y^{0,3}, Y^{0,4}\}$ are embedded in the final change map. These multiple outputs can prevent the network from overfitting to the training data, improving the model's overall accuracy.

The output Y of MaskHead can be formulated as follows:

$$\begin{cases} Y^{0,j} = h(x^{0,j}), & j \in \{1, 2, 3, 4\} \\ Y^{0,5} = h([Y^{0,1}, Y^{0,2}, Y^{0,3}, Y^{0,4}]), \end{cases} \tag{2.3}$$

where $h(\cdot)$ indicate a 1x1 convolution operation and $Y^{0,j}$ denotes the output from the j-th level.

Let $x^{i,j}$ denote the output of node $X^{i,j}$, where i indexes the down-sampling layer along the encoder and j indexes the convolution layer of the dense block along the skip

connection. Then $x^{i,j}$ is formulated as follows:

$$x^{i,j} = \begin{cases} \mathcal{P}(\mathcal{H}(x^{i-1,j})), & j = 0 \\ \mathcal{H}([|(x_A^{i,0} - x_B^{i,0})|, \mathcal{U}(x^{i+1,j-1})]), & j = 1 \\ \mathcal{H}([|(x_A^{i,0} - x_B^{i,0})|, [x^{i,k}]_{k=1}^{j-1}, \mathcal{U}(x^{i+1,j-1})]), & j > 1 \end{cases} \tag{2.4}$$

where $\mathcal{U}(\cdot)$ denote an up-sampling layer. Equation 2.4 displays that for $j > 1$, the node receives inputs from the up-sampling layer, the skip connection from the Siamese subtraction operation, and the skip pathways from the other convolution units at the same level.

2.2.3.1 Loss Functionality

The following paragraph is primarily extracted from the methodology section of the specialization project (Nyegaarden; 2022).

Due to the rarity of mines in the detection field of a sonar, the data collected from the HISAS-1030 are heavily imbalanced. During deep neural network training, dealing with imbalanced data is challenging as pixel-based change detection requires pixel-wise labeling and small mines contribute less to the loss. To address the class imbalance problem, Zhu et al. (2019) propose a hybrid loss function consisting of contributions from both dice loss and focal loss to improve the classification accuracy of detecting the minority class of changed pixels in the dataset. The hybrid loss function is defined as:

$$\mathcal{L}_{hybrid} = \mathcal{L}_{focal} + \lambda \mathcal{L}_{dice}, \tag{2.5}$$

where λ refers to the weight that balances the two losses. Peng et al. (2019) studied the significance λ had for the final change detection results when using a hybrid loss function consisting of weighted cross-entropy loss and dice loss. The evaluation metrics achieved maximum values when λ was set to 0.5, meaning that the influence of the binary cross-entropy loss and dice coefficient loss are well balanced. Consequently, λ has been set to 0.5 for training MaskHead.

The focal loss down-weights the loss for well-classified samples and focuses more on the loss for hard-to-classify samples. This allows the model to pay more attention to the rare classes and helps reduce the bias towards the more common class of unchanged

pixels to improve performance. The focal loss can be formulated as follows:

$$\mathcal{L}_{focal} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t), \tag{2.6}$$

where p_t is the model's predicted probability for the true class "change," α_t is a weighting factor to balance the contribution of each sample to the total loss, and γ is the focus parameter that controls the weighting of easy and hard examples.

The dice loss can be expressed as:

$$\mathcal{L}_{dice} = 1 - \frac{2 \sum_{i=1}^{n} \hat{y}_i y_i}{\sum_{i=1}^{n} \hat{y}_i + \sum_{i=1}^{n} y_i + \epsilon}, \tag{2.7}$$

where $\hat{y}$ and y denote the predicted probabilities and the ground truth labels for the change map, respectively. ϵ is a small constant added to the numerator and denominator to prevent division by zero. Equation 2.7 calculates the ratio between the intersection and union of predicted and ground truth pixels, with a value of 1 indicating perfect overlap and 0 indicating no overlap. By minimizing the dice loss, the model can learn to predict more accurate and detailed change maps, improving the performance of the change detection task. Additionally, dice loss reduces the impact of the class imbalance problem compared to conventional loss functions such as binary cross-entropy loss.

The loss can be calculated from the outputs of five semantic levels of the MaskHead architecture that has integrated a deep supervision strategy. The overall loss function for MaskHead can thus be defined as:

$$\mathcal{L}_{mask} = \sum_{j=1}^{5} w_j \mathcal{L}_{hybrid}^{j}. \tag{2.8}$$

Here, w_j corresponds to the weights of the five semantic level outputs in the network, and $\mathcal{L}_{hybrid}^{j}$ denotes the loss from the j-th side output. In the former use of the U-Net++ structure, the weight-vector w_j has generally been set to 1.0. Li et al. (2020), however, explored the importance level of output at different semantic levels and observed that as outputs of the deeper semantic levels are removed, the influence on the final result has an increasing tendency. In other words, deeper outputs were more important than shallower ones. Therefore, to make the network pay more attention to features of deeper levels, Li et al. (2020) adjusted the weights w_j to $\{0.5, 0.5, 0.75, 0.75, 1.0\}$. This

adjustment improved the performance of the Siamese Nested U-Net, resulting in a 0.3% increase in the F1-score compared to setting $w_j = 1$ (Li et al.; 2020).

2.2.4 BoxHead

BoxHead architecture is based on the Faster R-CNN object detection network (Ren et al.; 2016). It inputs feature maps from FeatureNet and generates refined box locations and classification outcomes using fully connected layers. Faster R-CNN addresses some of the limitations of Fast R-CNN, mentioned in 1.2.4.2, by incorporating a Region Proposal Network (RPN) that generates high-quality region proposals. The RPN and Fast R-CNN detection components are merged into a single network, where the RPN module serves as the "attention" mechanism guiding the network where to look for new objects. The Faster R-CNN model has demonstrated state-of-the-art object detection performance on popular datasets such as PASCAL VOC and Microsoft COCO datasets, making it one of the most widely used object detection algorithms (Ren et al.; 2016). Although object detection and change detection are different computer vision tasks, the hypothesis is that utilizing the pixel-wise difference images (DI) produced by FeatureNet as input to an object detection algorithm will yield satisfactory results for the change detection task. Additionally, Faster R-CNN allows for the easy integration of various backbone networks, facilitating the multi-task design of Siam R-CNN. Figure 2.6 illustrates the general BoxHead architecture. For clarity, in the presented architecture, the Fast R-CNN detection module described in the Faster R-CNN paper (Ren et al.; 2016) will be referred to as BoxPredictor.

2.2.4.1 RPN

The RPN inputs the feature maps from FeatureNet and outputs a set of rectangular object proposals, each with an objectness score. Figure 2.7 shows the detailed schematic of the RPN. The RPN constitutes a neural network, the RPN head, and non-neural network functionalities. The following paragraphs describe some of the main components depicted in Figure 2.7.

RPN Head. The RPN head is the neural network of the RPN. The RPN head functions by sliding a small network across the feature maps to generate proposals. Specifically, the small network processes an $n \times n$ spatial window of the feature map, mapping each

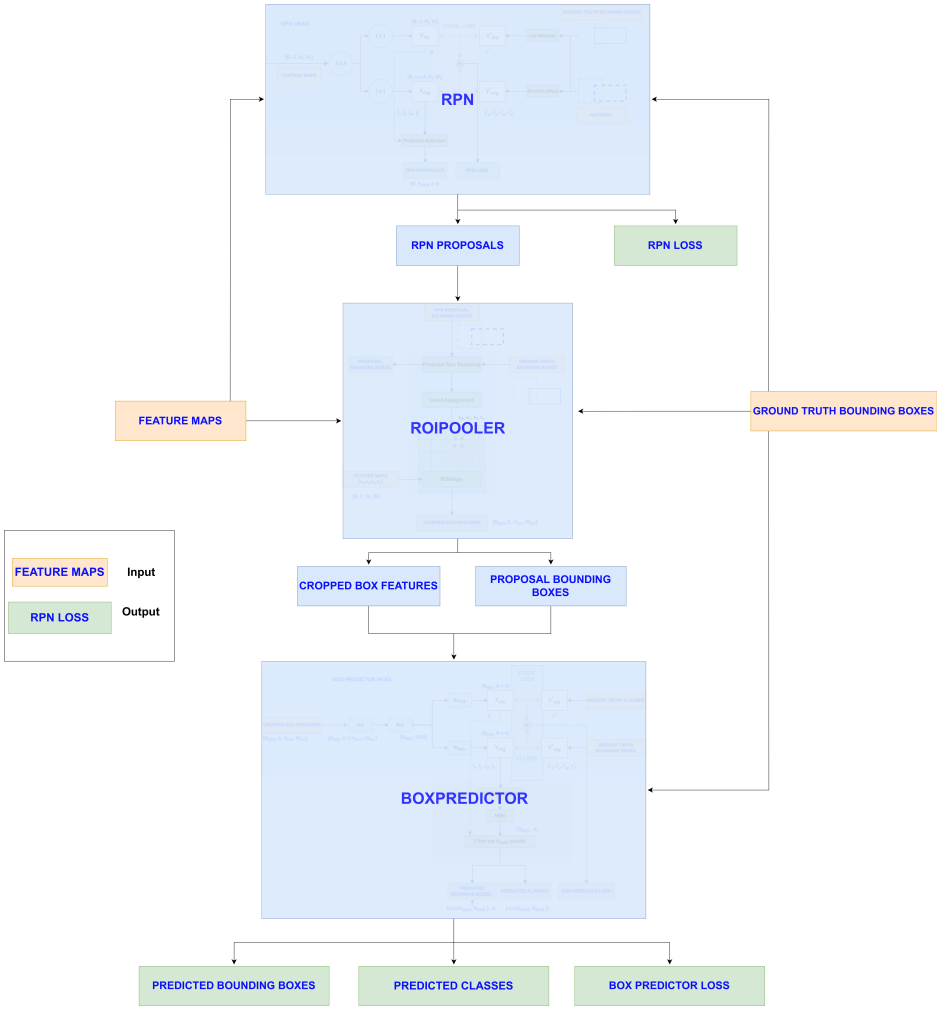


Figure 2.6: BoxHead Architecture

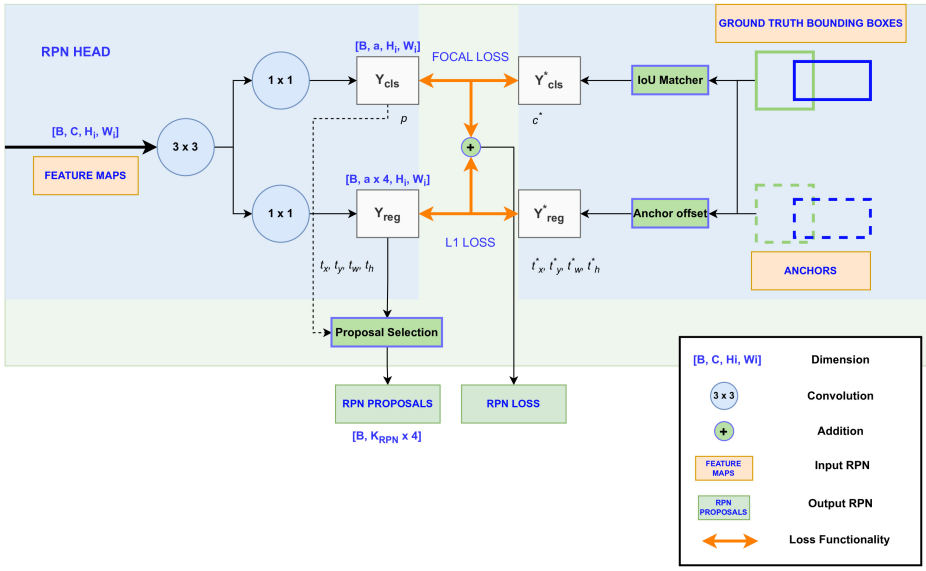


Figure 2.7: RPN Architecture

window to a lower-dimensional feature. The generated feature is then fed into two fully connected layers: the box regression layer (reg) and the box classification layer (cls). The output feature maps are the following:

1. Y_{RPN_cls} $(B, num\_anchors, H_i, W_i)$: probability map of object existence in an anchor (objectness score).

2. Y_{RPN_reg} $(B, num\_anchors \times 4, H_i, W_i)$: relative box shape to anchors.

Here, B stands for batch size, and H_i and W_i correspond to the spatial dimensions of the respective multi-scale feature maps. The output Y can be mathematically defined as:

$$\begin{cases} Y_{RPN\_cls} = h_a(\mathcal{H}(x_j)), & j \in \{0, 1, 2, 3\} \\ Y_{RPN\_reg} = h_{a\times 4}(\mathcal{H}(x_j)), & j \in \{0, 1, 2, 3\}. \end{cases} \tag{2.9}$$

Here, $\mathcal{H}(\cdot)$ denotes a 3×3 convolution operation, and $h(\cdot)$ indicates a 1×1 convolution operation that adjusts the number of channels in the output feature map to a and $a \times 4$, respectively. The variable a represents the number of anchors at every pixel location,

and $a \times 4$ represents the number of predicted anchor delta parameters for all anchors.

Anchors. The network predicts several region proposals at each sliding window location that are parameterized with respect to predefined anchors. These anchors are reference boxes centered at the sliding window and are associated with a scale and aspect ratio (Ren et al.; 2016). Each anchor is represented by a set of four coordinates, which specify its size relative to the spatial location of the network's sliding window.

Ground Truth Boxes and Anchors. During training, the RPN is trained to adjust the predicted anchor coordinates to better align with the ground truth bounding boxes in the image. To train the RPN network, each anchor is assigned a binary class label, indicating whether it corresponds to an object or not. The IoU overlap between the anchors and the ground truth boxes is calculated to assign the labels, and a predefined threshold is used to evaluate whether an anchor is positive or negative. For each anchor, the network also predicts four regression parameters t_x, t_y, t_w, and t_h that describe the location and size of the predicted bounding boxes relative to the anchor. These regression parameters and a set of target parameters t_x^*, t_y^*, t_w^*, and t_h^*, also referred to as "deltas," are defined as:

$$\begin{aligned} t_x &= \frac{x - x_a}{w_a}, \quad t_y = \frac{y - y_a}{h_a} \\ t_w &= log(\frac{w}{w_a}), \quad t_h = log(\frac{h}{h_a}) \\ t_x^* &= \frac{x^* - x_a}{w_a}, \quad t_y = \frac{y^* - y_a}{h_a} \\ t_w^* &= log(\frac{w^*}{w_a}), \quad t_h = log(\frac{h^*}{h_a}). \end{aligned} \tag{2.10}$$

Here, t_i represents a vector representing the four anchor deltas of the predicted bounding box, while t_i^* corresponds to the anchor deltas of a ground truth box associated with a positive anchor. The parameters x, y, w, and h denote the box's center coordinates, width, and height. The predicted bounding box, anchor box, and ground truth box are denoted by x, x_a, and x^*; the same applies for y, w, and h. The process can be viewed as a bounding box regression, whereby the predicted bounding box for the given anchor is adjusted to better match the ground truth box associated with the concerned anchor.

Loss Calculation. The RPN is optimized by minimizing a multi-task loss function L_{RPN}. Drawing inspiration from the Faster R-CNN implementation (Ren et al.; 2016),

the loss function is defined as:

$$\mathcal{L}_{RPN}(p,t) = \frac{1}{N_{cls}} \sum_i \mathcal{L}_{cls}(p_i, c_i^*) + \lambda \frac{1}{N_{reg}} \sum_i c_i^* \mathcal{L}_{reg}(t_i, t_i^*), \tag{2.11}$$

where i represents the index of an anchor in the mini-batch, p_i is the predicted probability that the anchor i is an object, and c_i^* is the corresponding ground truth label, i.e., 1 if the anchor is positive, and 0 if the anchor is negative. Equation 2.11 is normalized by N_{cls} and N_{reg} and weighted by the parameter λ.

$\mathcal{L}_{reg}$ is the bounding box regression loss and is defined as a smooth localization loss (L1) function adopted from Fast R-CNN (Girshick; 2015):

$$\mathcal{L}_{reg}(t_i, t_i^*) = smooth_{L1}(t_i - t_i^*), \tag{2.12}$$

in which

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if \quad |x| < 1 \\ |x| - 0.5 & otherwise. \end{cases} \tag{2.13}$$

$\mathcal{L}_{cls}$ refers to the logarithmic loss for classification between two classes, namely "object" and "not an object." In the Faster R-CNN paper (Ren et al.; 2016), $\mathcal{L}_{cls}$ is specified as the binary cross-entropy loss. The Faster R-CNN paper demonstrates remarkable performance for datasets that feature a greater amount of larger objects, such as the PASCAL VOC and MS COCO datasets. In contrast, the HISAS-1030 sonar dataset, as discussed in Section 2.2.6, is significantly imbalanced. The ground truth bounding boxes, on average, occupy only approximately 0.03% of the total image area, resulting in most of the cropped SAS input pairs lacking relevant changes. To tackle the imbalance problem in the Siam R-CNN architecture, the cross-entropy loss is substituted with the focal loss function defined in Equation 2.6. The focal loss aims to mitigate the bias introduced by the dominant "background" class and facilitate the convergence of the RPN network. Note that the bounding box regression loss $\mathcal{L}_{reg}$ is only calculated for foreground classes where the anchor corresponds to a ground truth object. This approach ensures that the class imbalance present in the SAS dataset does not affect the regression loss component.

Proposal Selection. Proposal selection involves reversing Equation 2.10 to obtain bounding box proposals $\{x, y, w, h\}$ from predicted anchor deltas $\{t_x, t_y, t_w, t_h\}$. These

proposals are then ranked based on their predicted objectness score, with the K_{RPN} highest-ranked proposals being selected and forwarded to the ROIPooler for further processing. This approach is inspired by the Detectron2 implementation of Faster R-CNN, where the proposals are treated as fixed during joint training with the BoxPredictor (Wu et al.; 2019). This strategy is known as "approximate joint training" in the Faster R-CNN paper (Ren et al.; 2016), and it is simple to implement and integrate into the Siam R-CNN architecture.

2.2.4.2 ROIPooler

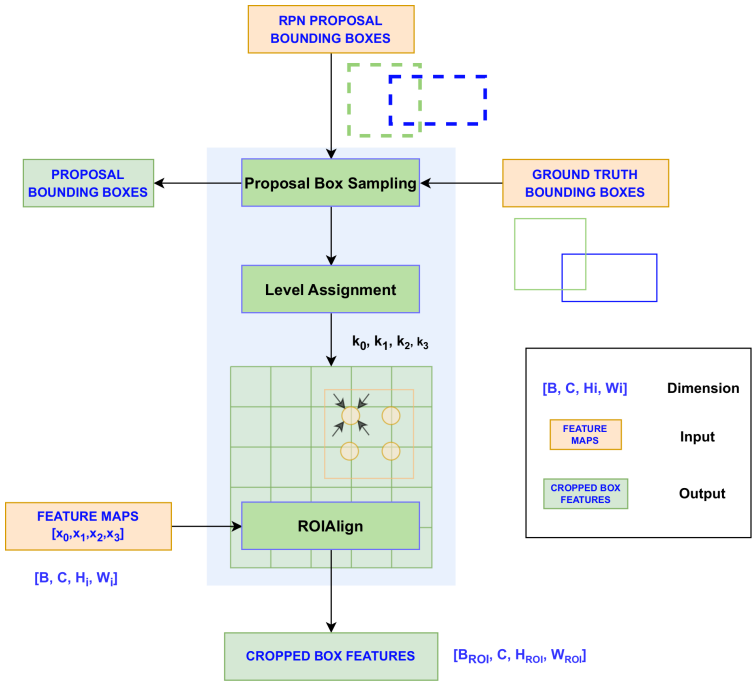


Figure 2.8: ROIPooler Architecture

The ROIPooler module plays a crucial role in Siam R-CNN by cropping the regions of interest (ROIs) from feature maps and feeding them to the bounding box predictor. The functionality of the ROIPooler is depicted in Figure 2.8, and it is inspired by the Detectron2 ROIPooler architecture (Wu et al.; 2019). The ROIPooler can be divided into

three stages: proposal box sampling, level assignment, and ROIAlign.

1. **Proposal Box Sampling.** The first stage of the ROIPooler module involves matching the proposal boxes from the RPN with the ground truth boxes. Proposals with a higher Intersection over Union (IoU) than a predefined threshold are labeled positive, while others are labeled negative. During training, the predicted proposals are augmented with the ground truth boxes to speed up the learning process. Moreover, the boxes are balanced to maintain a predefined fraction of positive samples.

2. **Level Assignment.** Following this stage, the proposal boxes are assigned to the appropriate feature map, considering that the feature maps from FeatureNet are of multiple scales. Lin et al. (2017) has proposed a level assignment equation for feature pyramid networks (FPNs) which can be mathematically expressed as:

$$k = \lfloor k_o + \log_2(\sqrt{w * h}/C) \rfloor, \tag{2.14}$$

   where w and h are the width and height of the bounding box in the original difference image, k_o is the feature map level index on which a canonically-sized box should be placed, and C is the canonical box size in pixels.

3. **ROIAlign.** ROI pooling is a technique used to extract a small feature map, such as 7×7, from each region of interest. However, the ROIPool-operation presented in Fast R-CNN (Girshick; 2015) has a misalignment problem between the input feature map and the ROI. To solve this issue, the ROIAlign technique proposed by Mask R-CNN (He et al.; 2017) uses bilinear interpolation to obtain exact pixel-level feature maps for each region proposal. The ROIAlign operation is explained in greater detail in the Mask R-CNN paper (He et al.; 2017). The ROIAlign layer outputs cropped instance features, which include balanced positive and negative regions of interest. The output has a size of $[B_{ROI}, C, H_{ROI}, W_{ROI}]$, where B_{ROI} is the number of ROIs in the batch (batch size $\times$ predefined number of ROIs per image), C is the channel number, and H_{ROI} and W_{ROI} represent the width and height, respectively.

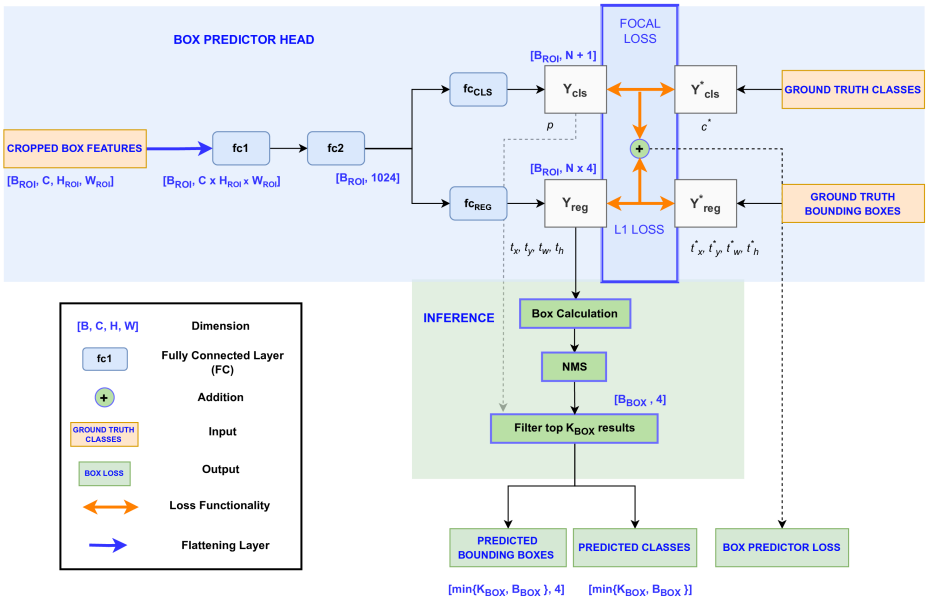


Figure 2.9: BoxPredictor Architecture

2.2.4.3 BoxPredictor

After ROI pooling, the cropped feature vectors are fed into the bounding box predictor head of BoxPredictor. The BoxPredictor architecture is depicted in Figure 2.9. The feature vectors are passed through a series of fully connected layers, which are subdivided into two output layers. The first output layer computes the softmax probability estimate over N object classes, denoted as Y_{cls}. The second output layer produces the refined bounding box position for the N object classes, represented as Y_{reg} (Girshick; 2015). The output tensors generated by the final layers can be expressed as:

$$\begin{cases} Y_{cls} = \mathcal{L}_{(1024,N+1)}(y_{FC}), & \text{shape: } (B_{ROI}, N+1) \\ Y_{reg} = \mathcal{L}_{(1024,N\times 4)}(y_{FC}), & \text{shape: } (B_{ROI}, N\times 4), \end{cases} \tag{2.15}$$

where y_{FC} is mathematically defined as

$$y_{FC} = \mathcal{L}_{(C\times H\times W,1024)}(\mathcal{F}(Y_{ROI})). \tag{2.16}$$

Here, $\mathcal{L}_{x,y}(\cdot)$ refers to a fully connected layer that conducts a linear transformation, modifying the dimensions of the tensor from x to y. Additionally, $\mathcal{F}$ is a flattening layer that reshapes the multidimensional input into a one-dimensional tensor. The symbol Y_{ROI} denotes the box features that are generated by the ROI pooling layer.

Loss Calculation. Two loss functions are applied to the final output tensors. The total loss from the BoxPredictor output can be expressed as:

$$\mathcal{L}_{box}(\mathbf{p}, \mathbf{c}^*, \mathbf{t}, \mathbf{t}^*) = \frac{1}{N_{cls}} \sum_{i=1} L_{cls}(\mathbf{p_i}, c_i^*) + \lambda \frac{1}{N_{reg}} \sum_{i=1} L_{reg}(\mathbf{t_i}, \mathbf{t_i^*}), \tag{2.17}$$

where i represents the index of the mini-batch, $\mathbf{p_i}$ represents the unnormalized logits for each class and c_i^* represents the ground truth class index. $\mathbf{t_i}$ represents the foreground bounding box delta predictions, while $\mathbf{t_i^*}$ corresponds to the foreground ground truth bounding box deltas associated with the proposals that match the features that were used to compute predictions.

Here, $\mathcal{L}_{reg}$ denotes the bounding box regression loss and is defined as the smooth localization loss (L1) function in Equation 2.13.

The loss function $\mathcal{L}_{cls}$ compares the prediction scores $\mathbf{p_i}$ with the ground truth

class index c_i^*. In the Faster R-CNN paper (Ren et al.; 2016), $\mathcal{L}_{cls}$ is implemented using a softmax cross-entropy loss function. However, as discussed in Section 2.2.4.1, the SAS dataset introduces a significant imbalance between positive and negative classes, requiring a loss function that accounts for the imbalanced classes. Therefore, similar to the RPN loss functionality, L_{cls} is implemented as a focal loss function, as defined in Equation 2.6. The term L_{reg} is only calculated for foreground proposals that are matched to a ground truth bounding box, and there is no need to make additional adjustments to account for class imbalance.

Inference. During the model evaluation phase, the final box coordinates are derived from the prediction deltas Y_{reg} to assess performance. The bounding box coordinates undergo three stages of processing. Firstly, boxes with low detection scores are filtered out. Secondly, non-maximum suppression (NMS) is applied to eliminate overlapping boxes for each class. NMS is a post-processing technique that selects only the most confident predictions while removing redundant bounding boxes that significantly overlap with other boxes. Finally, if the number of remaining boxes exceeds a predefined value K_{BOX}, the top K_{BOX} results are selected for performance evaluation.

2.2.5 Attention Mechanism

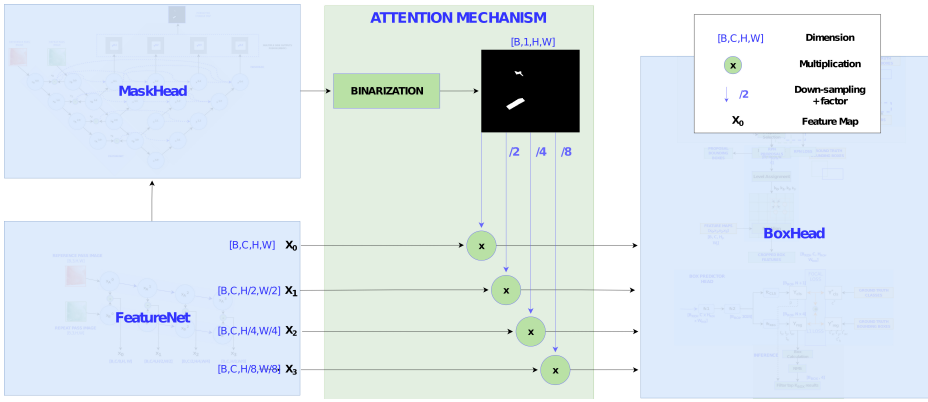


Figure 2.10: Attention Mechanism

In order to leverage the learned change map to its fullest potential, a straightforward yet efficient attention mechanism is incorporated into the network. This attention

mechanism is derived from SpotNet, a self-attention multi-task network introduced by Perreault et al. (2020), which combines the object detection and segmentation task. After obtaining the final change map output from MaskHead, down-sampling is performed to match the spatial dimensions of the original feature maps. To reduce the impact of regions that are unlikely to contain new objects, an adjustment is made to the feature maps from FeatureNet before they are passed to BoxHead. Specifically, each channel of the feature maps is multiplied by the change map. This multiplication helps minimize false positives in irrelevant areas and diminishes the influence of non-relevant regions on the BoxHead results. Figure 2.10 details the proposed attention mechanism within the Siam R-CNN architecture (see Figure 2.2).

2.2.6 Loss Functionality

During training, Siam R-CNN computes a multi-task loss on each image pair. This loss function, denoted as $\mathcal{L}$ incorporates three individual losses:

$$\mathcal{L} = \lambda_{RPN}\mathcal{L}_{RPN} + \lambda_{box}\mathcal{L}_{box} + \lambda_{mask}\mathcal{L}_{mask}. \tag{2.18}$$

The loss formulations for $\mathcal{L}_{mask}$, $\mathcal{L}_{RPN}$, and $\mathcal{L}_{box}$ are defined in Equations 2.8, 2.11, and 2.17, respectively. The total loss is obtained by adding up all the individual losses, where the weights λ_{RPN}, λ_{box}, and λ_{mask} are important hyperparameters for optimizing the multi-task loss.

For most neural networks, training multiple tasks is difficult without finding the correct balance between those tasks. Searching for optimal weighting is prohibitively expensive and difficult to resolve with manual tuning. However, as detailed in Section 1.2.6, systematic methods have been introduced to address the issue of balancing task-specific weights. To adjust the weights λ_{RPN}, λ_{mask}, and λ_{box}, two approaches will be implemented: gradient normalization (GradNorm) and dynamic weight averaging (DWA). The task-specific weights for GradNorm and DWA are derived using equations 1.11 and 1.12, respectively. GradNorm adjusts the task-specific weights based on the training rates and gradient magnitudes of the loss, providing a more robust option with less manual tuning. On the other hand, DWA computes the weights solely based on the loss gradients, eliminating the need for additional backward passes. Both approaches will be evaluated and compared in terms of their effectiveness in optimizing the multi-

task architecture of Siam R-CNN. The evaluation results will be presented in Chapter 3.

The uncertainty weighting scheme presented in Section 1.2.6 will not be evaluated in this thesis due to compatibility challenges with Siam R-CNN.

2.3 Pretext Task Architecture

The following sections present the different pretext tasks used for pre-training FeatureNet. The auxiliary tasks are designed such that it aims to help the model learn features that are expected to be useful in the downstream change detection task. FeatureNet will be pre-trained on a set of unlabeled image pairs to enable it to be fine-tuned efficiently on a smaller labeled set of SAS image pairs.

2.3.1 Pretext Task 1

The first pretext task involves performing binary classification, where the network predicts whether a pair of patches are overlapping or not. This task is inspired by Leenstra et al. (2019), that exploited the temporal consistency of Sentinel-2 imagery to obtain a self-supervised learning signal. The task enables the FeatureNet model to ignore irrelevant variations, such as rocks and internal waves on the seafloor. Instead, it should focus on relevant spatial dissimilarities between patches, ultimately reducing the number of false alarm detections in the SAS difference image. Each training example contains a patch pair $\{(p_A, p_B), y\}$, where y denotes the associated pseudo label. y is 0 for spatially overlapping pairs and 1 for non-overlapping pairs. It is worth noting that neither patch includes any newly employed objects or changes intended to facilitate learning features in areas where no changes have occurred.

2.3.1.1 Network Architecture

For pretext task 1, the network architecture comprises FeatureNet as the feature extractor, followed by a classifier that consists of three fully connected layers. The classifier's output is the raw logits, representing the probability that the patch pair is spatially overlapping or non-overlapping. Figure 2.11 presents the network architecture for the first pretext task.

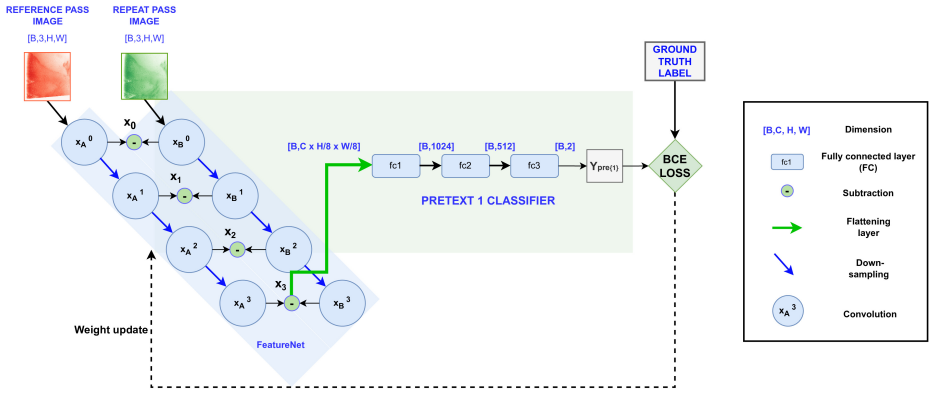


Figure 2.11: Pretext 1 Network Architecture

2.3.1.2 Loss Functionality

The optimization of the network parameters is achieved by minimizing the binary cross-entropy loss, which is commonly preferred when dealing with well-balanced datasets. This preference can be attributed to the smooth gradient it provides for the backpropagation process and the straightforward interpretation of the model's output, as stated by Leenstra et al. (2021). The binary cross-entropy loss is defined by the following equation:

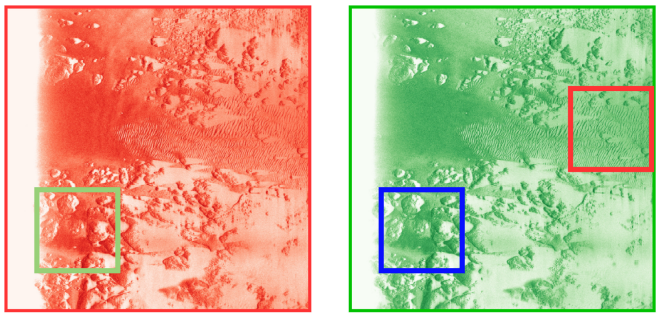
$$\mathcal{L}_{bce} = \frac{1}{N}\sum_{n=1}^{N} -(y_n \log(P(y_n)) + (1 - y_n)\log(1 - P(y_n)), \tag{2.19}$$

where y_n and $P(y_n)$ are the target values and input probabilities, respectively. N is the batch size.

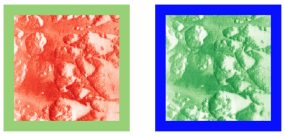
2.3.2 Pretext Task 2

The second pretext task is a contrastive method, aiming to learn image representations that place overlapping patches closer together in the high-dimensional feature space and non-overlapping patches far apart. As for pretext task 1, this approach draws inspiration from the self-supervised Sentinel-2 change detection paper by Leenstra et al. (2021). Each training sample consists of a set of triplets (p_A, p_B^1, p_B^2), where patches p_A and p_B

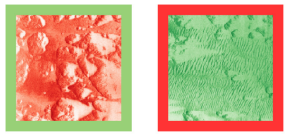
are spatially overlapping, while p_A and p_B^2 are not. The objective of this pretext task aligns with pretext task 1: to enable the change detection network to acquire features that map unchanged pixel pairs into the same region of the feature space, thus making unchanged areas more similar. The patch sampling strategy for both pretext tasks is illustrated in Figure 2.12.



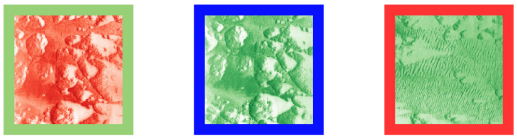
(a) Location of patches in the SAS image pair



(b) Pretext 1 Overlapping Patch Pair



(c) Pretext 1 Non-Overlapping Patch Pair



(d) Pretext 2 Patch Triplet

Figure 2.12: Pretext Patch Sampling Strategy

2.3.2.1 Network Architecture

Figure 2.13 presents the network architecture for the second pretext task. While bearing a resemblance to FeatureNet, this architecture deviates by featuring three branches instead of the original Siamese structure's two branches. Additionally, unlike the

concatenation approach in FeatureNet, the three feature branches in this architecture remain separate. As a result, the network produces three down-sampled feature vectors, which are subsequently optimized using the loss function described in the following section.

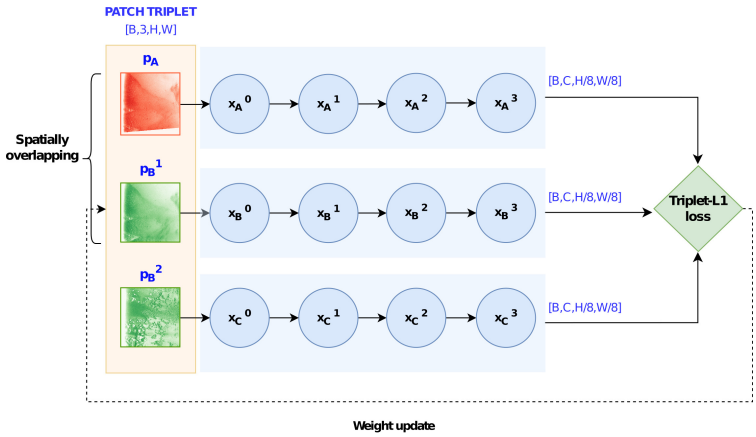


Figure 2.13: Pretext 2 Network Architecture

2.3.2.2 Loss Functionality

To optimize the mapping of similar patches closer together and dissimilar patches far apart in the feature space, a combination of triplet margin loss and L1 loss is employed. The complete loss function can be defined as follows:

$$\mathcal{L}_{triplet\_L1} = max(||\mathbf{f^1} - \mathbf{f^2}||_2 - ||\mathbf{f^1} - \mathbf{f^3}||_2 + \alpha, 0) + \gamma \cdot |\mathbf{f^1} - \mathbf{f^2}|. \tag{2.20}$$

Here, $\mathbf{f^i}$ represents the feature vector for patch i. The term α denotes the margin between positive and negative pairs, determining the desired separation distance. The parameter γ is a hyperparameter that balances the influence of the triplet loss and L1 loss. In the article by Leenstra et al. (2021), the values of α and γ were experimentally set to 1 for optimal performance. Minimizing this loss function during training allows the network to effectively differentiate between similar and dissimilar patches in the feature space.

Chapter 3

Experiments and Results

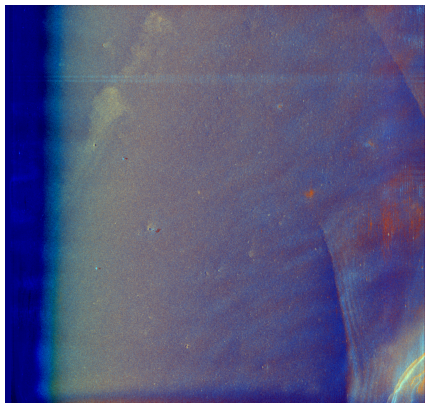
This chapter provides the details of the experiments conducted and their results. It first gives a detailed description of the SAS dataset, covering how the training and ground truth data are generated. It then proceeds to describe the implementation details and justifies the selected parameters. Following that, an ablation study is presented, discussing the effects of different components, including self-supervised pre-training, multi-task learning, and attention integration. Finally, the experimental results are compared to other change detection methods.

3.1 Dataset and Evaluation Criteria

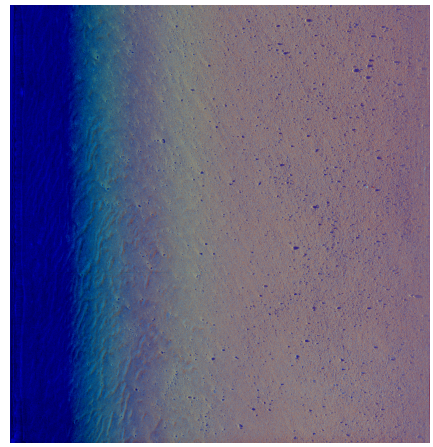
3.1.1 Dataset

The SAS data used in this thesis was collected from various sites in Norwegian and international coastal waters by deploying several HUGIN autonomous underwater vehicles (AUVs) equipped with a HISAS-1030 sonar. The SAS dataset comprises 275 images that capture the seafloor with varying degrees of texture, stone, and bathymetry. Some of the images have noticeable changes in the form of mine-sized objects that have been placed between the two surveys. These images have a pixel size of approximately 4×4 cm and are converted to $3 \cdot 8$ bit PNG files. The color coding of the images is as follows: the *red* channel represents the reference image mapped to the pixel coordinates

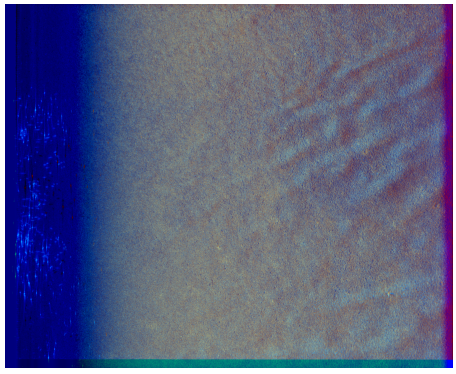
of the repeat-pass image, the *green* channel represents the repeat-pass image, and the *blue* channel represents the difference image. The different image results from subtracting the transformed reference image from the repeat-pass image. Examples from the HISAS-1030 SAS dataset are depicted in Figure 3.1. Each image in the figure is accompanied by a caption that provides information about the seafloor location and changes observed in the image.



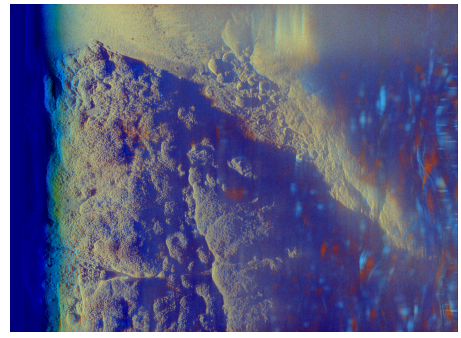
(a) Larvik, Norway - 4 deployed targets.



(b) Nordleksa, Norway - 2 deployed targets.



(c) Biodola, Italy - no deployed targets.



(d) Bonassola, Italy - no deployed targets.

Figure 3.1: HISAS-1030 SAS Images

The images have undergone preprocessing steps, including a logarithmic transform to emphasize sonar highlights and shadows more equally, followed by anisotropic

diffusion filtering to reduce image speckle and improve data-driven co-registration. The images are also resampled from slant-range to ground-range coordinates using high-resolution bathymetry from interferometric sonar data processing (Midtgaard; 2018). These steps improve the quality and clarity of the images and facilitate further analysis.

A large fraction of the SAS dataset does not contain deployed objects between the reference and repeat-pass survey. To make use of these images, the pretext tasks are specifically designed to learn features from the unchanged areas within the SAS images. Therefore, the dataset is divided into two subsets: 225 images that do not contain known changes and 50 images with known changes. The first dataset of 225 images is utilized for training and evaluating the pretext weights, while the second dataset of 50 images is employed to train and evaluate the downstream change detection task.

3.1.1.1 Ground Truth Preparation

The downstream change detection task adjusts its parameters and biases by minimizing the error between the predicted and ground truth output. To facilitate this, ground truth annotations were created for each learning task using a computer vision annotation tool, CVAT (CVAT.ai Corporation; 2022). This involved drawing polygons and bounding box rectangles around the regions of change in the SAS difference image. In sonar images, a highlight region is often followed by a shadow region for most targets of interest. Therefore, both the echo from the changed objects and the corresponding shadow were marked as changed regions.

However, some of the HISAS-1030 difference images contained unwanted semicircular and successive echoes from the center of the deployed target. Figure 3.2 displays a cropped difference image from the HISAS-1030 Bonassola survey demonstrating the effect. The semicircular echoes that extend from the target position are caused by sidelobes. These sidelobes represent undesired energy that appears around the main beam, resulting in echoes that deviate from the intended direction of the signal. The successive echoes in the range direction are attributed to multipath reflections, which occur when transmitted signals take multiple paths and experience different delays before reaching the receiver due to reflections, scattering, or diffraction. Ideally, for operational purposes, it is preferable to annotate only the actual objects as changes. This is because SAS artifacts depend on the accuracy of the sonar and available survey tech-

nology. However, due to a significant class imbalance in the SAS dataset, considering these artifacts as changes could potentially enhance training performance. Therefore, annotations with and without the sonar artifacts will be evaluated to compare their performance.

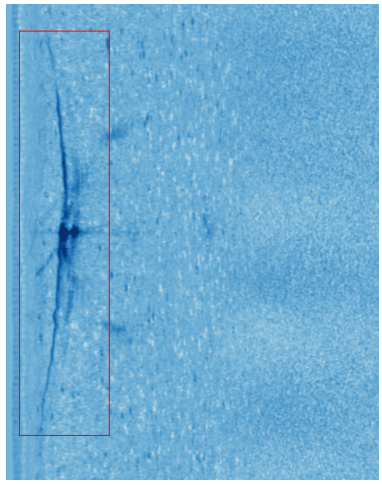
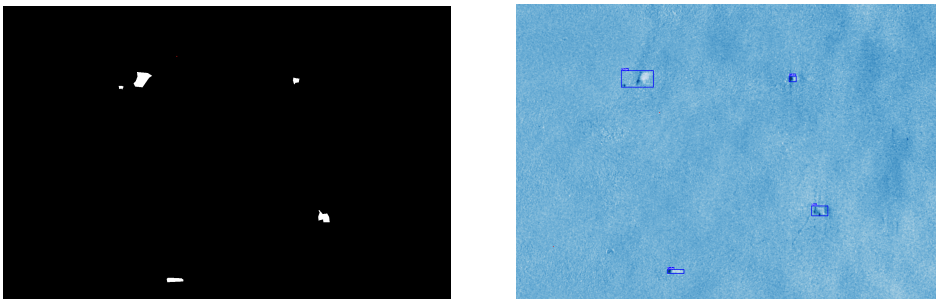


Figure 3.2: HISAS-1030 Bonassola Difference Image with SAS Artifacts

The annotated polygons were further processed to generate a change map with black and white pixels representing unchanged and changed areas. Bounding box coordinates and object class assignments were converted into a JSON file in MS COCO annotation format, compatible with the Detectron 2 Faster R-CNN framework.

A visual representation of a generated change map and ground truth bounding box rectangles is shown in Figure 3.3. In Figure 3.3, four deployed objects can be observed on the seafloor after the reference survey.

As mentioned in Section 1.2.5, a self-supervised predictive learning strategy aims to alleviate the annotation bottleneck by using a programmatically derivable label when training the network. Hence, for pretext task 1 presented in Section 2.3.1, the labels are derived from the input images during run-time. The input pairs are randomly selected from the pool of unlabeled image pairs and are equally divided into two classes: overlapping ($y = 0$) and non-overlapping ($y = 1$). Pretext task 2 is a self-supervised contrastive method that eliminates the need for ground truth labels.



(a) Change map for MaskHead (b) Bounding boxes for BoxHead

Figure 3.3: SAS Ground Truth Generation Example

3.1.1.2 Training Data Preparation

The data preparation process described in this section applies to pretext and downstream change detection tasks. The reference and repeat-pass image pairs and the corresponding ground truth instances are divided into smaller patches to allow the network to focus on local details within the image. The patches used for training are $256x256$ pixels, which is a size that is suitable for detecting a deployed mine-sized object. To ensure consistency, the input patches are normalized by subtracting the mean and dividing by the standard deviation of the SAS dataset. Additionally, several data augmentation techniques are applied randomly to the SAS images during training. More specifically, each image has a 75% chance of being rotated by a multiple of 90 degrees and a 50% chance to be flipped in horizontal and vertical directions. These augmentations increase the diversity and effective size of the dataset, helping to prevent overfitting and improve the model's generalization ability.

The HISAS-1030 images used in this study were captured in various environments with different degrees of complexity. Partitioning these images into training and testing sets can significantly impact the evaluation of the model's performance. To address this challenge, the K-fold cross-validation technique is utilized. This technique involves dividing a dataset into k subsets, where $(k-1)$ subsets are used for training, and the remaining fold is used for evaluation. This process is repeated for each fold, ensuring that all folds are used once as the validation subset. Generally, a larger value of k (e.g., 5, 10, or 20) is preferred to utilize a larger number of patterns for training purposes

(Anguita et al.; 2012). In this case, since the pretext and change detection datasets consist of 225 and 50 image pairs, respectively, a 5-fold cross-validation technique is employed. The model's overall performance can be determined by calculating the average validation results from the 5 validation subsets.

3.1.2 Evaluation Criteria

The evaluation of Siam R-CNN results involves both visual qualitative comparisons and quantitative measures. The SAS dataset is heavily imbalanced and requires appropriate evaluation metrics to obtain reliable interpretations of the change detection results. Different metrics are employed to evaluate the predicted change map and bounding boxes.

Change Map Evaluation Metrics. A set of five evaluation metrics is utilized to evaluate the performance of the pixel-based change maps against the ground truth change maps. These metrics include precision (P), recall (R), overall accuracy (OA), F1 score (F1), and Cohen's Kappa score (Kap). The evaluation process involves analyzing the logits score of each pixel with respect to the ground truth pixels, which allows us to determine the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) samples. Specifically, TP represents the number of changed pixels correctly classified as changed, FP represents the number of unchanged pixels incorrectly flagged as changed, TN represents the number of unchanged pixels correctly detected as unchanged, and FN represents the number of changed pixels incorrectly classified as unchanged. Based on these quantities, the five evaluation metrics are defined as follows:

$$P = \frac{TP}{TP + FP} \tag{3.1}$$

$$R = \frac{TP}{TP + FN} \tag{3.2}$$

$$F1 = \frac{2 \times P \times R}{P + R}, \tag{3.3}$$

$$OA = \frac{TP + TN}{TP + TN + FP + FN}, \tag{3.4}$$

$$Kap = \frac{OA - PE}{1 - PE}, \tag{3.5}$$

where

$$PE = \frac{(TP + FP) \times (TP + FN)}{(TP + TN + FP + FN)^2}. \tag{3.6}$$

A higher precision value implies fewer false alarms, while a higher recall value indicates a lower rate of incorrect detection. The overall accuracy metric is the ratio of correctly detected pixels to all pixels in the image. However, these three metrics can provide a misleading overestimate of the result when the amount of changed pixels is a small fraction of the image. The F1 score provides a comprehensive evaluation metric by considering precision and recall simultaneously. In some cases, however, the F1 score fails to provide a concise representation of the model's performance. Recall, precision, and the F1 score do not account for the number of true negative samples, leading to problems when evaluating samples without any changes.

A more precise measure of the model's performance can be obtained using Cohen's Kappa score (Cohen; 1960) as a complementary evaluation metric. The Kappa score reflects the level of agreement between the predicted change map and the ground truth map and considers the true negative samples when calculating the score. This makes it a more accurate measure than simple percent accuracy for imbalanced datasets. By incorporating the Kappa score into the evaluation process, it is possible to gain a more well-rounded view of the model's capabilities. Note that higher F1 and Kappa values represent better overall performance.

Bounding Box Evaluation Metrics. Similar to the pixel-wise evaluation metrics, the core components TP, FP, TN, and FN are combined to evaluate the model's bounding box detection performance. However, an auxiliary metric known as Intersection over Union (IoU) is introduced for object-based detection algorithms to compute these quantities. The IoU metric quantifies the degree of overlap between the predicted bounding box and the ground truth bounding box. It is calculated as the area of their intersection divided by the area of their union, as defined in Equation 3.7.

$$IoU = \frac{area(B_{pred} \cap B_{gt})}{area(B_{pred} \cup B_{gt})}. \tag{3.7}$$

The IoU value is compared to a threshold parameter denoted by t to determine if a bounding box detection is correct or incorrect. Specifically, the detection is considered correct if $IoU \geq t$. Conversely, if the IoU value is less than the threshold value t, $IoU < t$,

then the detection is regarded as incorrect.

The BoxHead output consists of a list of bounding boxes, confidence levels, and corresponding classes. In order to evaluate the overall performance of the model's output, the Average Precision (AP) is computed. The AP metric is commonly used for object detection evaluation and facilitates comparison with other deep learning-based detectors. It is based on the precision-recall curve, constructed by varying the confidence threshold for detection and computing precision and recall values at each threshold (Everingham et al.; 2010). The AP score for a specific IoU threshold is obtained by calculating the area under the precision-recall curve using interpolation. The overall AP score for a class is computed by averaging the AP scores over a set of predetermined IoU thresholds. Mathematically, the total AP score can be expressed as:

$$AP = \frac{1}{I_{tot}} \sum_{i \subseteq iou} AP_i, \tag{3.8}$$

where iou and I_{tot} represent the set and the total number of IoU thresholds, respectively. AP_i is defined as:

$$AP_i = \frac{1}{IP_{tot}} \sum_{r \subseteq ip} p_{interp}(r), \tag{3.9}$$

where ip and IP_{tot} denote the set and the total number of interpolation recall points. $p_{interp}(r)$ denotes the interpolated precision at each recall level r.

The AP metric is commonly used to rank submitted works in object detection competitions. However, the selection of IoU thresholds and interpolation precision may vary depending on the object detection challenge, which is often specific to a dataset such as the PASCAL VOC (Everingham et al.; 2010) or Microsoft COCO dataset. In the absence of a challenge or a specifically designed AP score for sonar datasets, the PASCAL VOC design details are adopted primarily to facilitate the comparison of internal results. The PASCAL VOC challenge uses an IoU threshold of 0.5, meaning a predicted bounding box is considered correct if it has an IoU of 0.5 or higher with the ground truth bounding box (R. Padilla et al.; 2020). Additionally, the PASCAL VOC employs 11 recall thresholds for interpolating the precision-recall curve.

However, detecting small objects often results in lower IoU values due to the relatively smaller intersection area compared to the union area. Moreover, variations in sonar sensor angles relative to the deployed objects contribute to inconsistent inten-

sity values, shapes, shadows, and sizes in the dataset. These factors pose challenges for the neural network to accurately predict the correct boundaries of the bounding boxes. Therefore, two AP scores are calculated to account for a higher partial overlap between predicted and ground truth boxes in the SAS dataset: AP@0.25 and AP@0.5. The AP@0.25 metric considers an IoU threshold of 0.25 in addition to the PASCAL VOC IoU threshold of 0.5.

Statistical significance testing has not been conducted on the evaluation results presented in this thesis due to time constraints. While this limits the ability to make definitive conclusions about the observed findings, the study still provides valuable insights into using deep learning for change detection in SAS imagery.

3.2 Implementation Details

The pretext and change detection task architecture described in Chapter 2 is implemented using the PyTorch framework and a single NVIDIA A100-40GB GPU. AdamW is selected as the optimizer for both the pretext and downstream tasks. As discussed in Section 1.2.4.3, AdamW offers explicit weight decay handling, which is beneficial in preventing overfitting on the relatively small SAS dataset. Therefore, it is preferred over Adam and SGD as an optimizer. The initial learning rate is set to $5e-4$, decaying by a factor of 0.5 every eight epochs.

3.2.1 Pretext Implementation

The Pretext 1 network takes two tensors as input, each with dimensions $B \times 3 \times 256 \times 256$, where B represents the batch size initially set to 4. Similarly, for the Pretext 2 network, three input tensors (anchor, positive, and negative samples) are provided, following the exact dimensions and batch size as Pretext 1. The values for the triplet loss margin α and parameter γ in Equation 2.20 in the Pretext 2 network are determined empirically and set to 1.

The pretext tasks training process is performed for 50 epochs to ensure thorough data processing while maintaining reasonable training time.

3.2.2 Siam R-CNN Implementation

The Siam R-CNN network takes two input tensors with dimensions $B \times 3 \times 256 \times 256$, where B denotes the batch size. Initially, B is set to 4 to ensure the model encounters a wide range of SAS samples during training. MaskHead produces an output tensor of size $B \times 256 \times 256 \times 2$, representing the logits indicating the probability of unchanged or changed pixels. BoxHead has two output tensors: one with dimensions $min(K_{box}, B_{box}) \times 4$ representing the predicted bounding boxes, and another with dimensions $min(K_{box}, B_{box})$, representing the predicted classes. If the number of predicted bounding boxes B_{box} exceeds a predefined value K_{box}, the top K_{box} boxes are selected for evaluation. Initially, K_{box} is set to 2 to ensure efficient performance. It is worth noting that none of the patch pairs in the SAS dataset, each of size 256×256, contains more than two deployed targets.

The initial values for the multi-task loss weights, λ_1, λ_2, and λ_3, are all set to 1. These weights are automatically adjusted through the DWA or GradNorm weighting scheme. In the DWA weighting scheme, T in Equation 1.12 is set to 2 as suggested in the DWA paper (Liu et al.; 2019). In the GradNorm weighting scheme, the initial value of α in Equation 1.11 is set to 1.5, which exceeds the recommended value proposed in the GradNorm paper (Chen et al.; 2018). This adjustment is made empirically to account for the distinct learning dynamics between the tasks.

The Siam R-CNN training is conducted for a total of 30 epochs. Due to the significant number of parameters, approximately 29 million, in the Siam R-CNN architecture, extending the number of epochs becomes impracticable despite the potential for enhancing model performance.

The software implementation of Siam R-CNN, including the network architecture, loss function, and training and evaluation process, is included as an attachment to this thesis. The implementation is inspired by the publicly available code of Siamese Nested U-Net (Li et al.; 2020) and Detectron2 (Wu et al.; 2019) and is modified and adapted specifically for this study.

3.2.2.1 FeatureNet Implementation Details

The FeatureNet parameters and biases are initialized with the final, best-performing weights from the self-supervised pretext task. Referring to Figure 2.3, each convolutional

layer down-sample features by a max pooling layer with a stride of 2. The number of convolutional filters in the encoder is set to $\{32, 64, 128, 256\}$. In the specialization project, the number of convolutional filters was set to $\{64, 128, 256, 512\}$, based on the performance gain observed in the Siamese Nested U-Net model (Li et al.; 2020). However, the Siam R-CNN architecture introduces additional parameters, leading to longer training times and increased computational resource requirements. Therefore, the number of convolutional filters in the encoder is reduced to address these challenges.

3.2.2.2 MaskHead Implementation Details

The MaskHead decoder of the Siam R-CNN network (see Figure 2.5) uses convolutional filters of sizes $\{32, 64, 128, 256, 512\}$ to restore the features extracted from FeatureNet. The details of the loss function can be found in Section 2.2.3.1, where the weights w_j in Equation 2.8 are assigned values of $\{0.5, 0.5, 0.75, 0.75, 1.0\}$ and λ is set to 0.5. Considering the imbalanced nature of the SAS dataset, the focal loss parameters in Equation 2.6 are initially set to $\alpha_t = 0.01$ and $\gamma = 4$.

3.2.2.3 BoxHead Implementation Details

RPN. The architecture of the RPN is shown in Figure 2.7. Initially, 15 anchors are placed at every pixel location, generated based on sizes $\{32, 64, 128, 256, 512\}$ and aspect ratios $\{0.5, 1, 2\}$. The choice of anchor sizes is motivated by handling all variations of object sizes in the SAS dataset. To classify anchors as foreground or background, the minimum and maximum IoU thresholds are set to 0.3 and 0.7, respectively. Anchors with overlaps falling between thresholds are ignored. The number of top-scoring RPN proposals to keep after applying NMS, denoted as K_{RPN}, is set to 1000, following the default configuration in Detectron2 (Wu et al.; 2019). The loss function for the RPN is defined by Equation 2.11, where the weights N_{cls} and N_{reg} are defined as $B_{RPN} \times B$, representing the total number of samples used for computing the loss. The focal loss parameters of L_{cls} in Equation 2.11 are initially set to $\alpha_t = 0.01$ and $\gamma = 4$. Additionally, the parameter λ is set to 1.

ROIPooler. The ROIPooler module is illustrated in Figure 2.8. During the first stage of proposal box sampling, the IoU threshold between positive and negative boxes is set to 0.5. Moreover, the predefined positive sample fraction is set to 0.25 to ensure a

balanced representation. The assignment of proposal boxes to the appropriate feature map is determined by the level assignment equation, as defined in Equation 2.14. In this equation, the values of k_0 and C are initially set to 4 and 224, respectively, following the established canonical values used in the FPN architecture introduced by Lin et al. (2017). For the ROIAlign operation, a pooler resolution of 7 is chosen, corresponding to extracting a feature map of size 7×7 from each region of interest. This resolution selection is based on the default configuration from Detectron2 (Wu et al.; 2019).

BoxPredictor. The BoxPredictor module, detailed in Section 2.2.4.3, plays a critical role in the model evaluation phase by deriving the final box coordinates from the prediction deltas Y_{reg}. The dimensions of the prediction deltas Y_{reg} and classification scores Y_{cls} are $B_{ROI} \times (N \times 4)$ and $B_{ROI} \times (N+1)$, respectively. Here, B_{ROI} represents the number of ROI instances sampled from each image in a mini-batch, and N denotes the number of object classes. In this thesis, the primary focus is on improving binary change detection performance. Therefore, the number of classes N is set to 1, referring to the "changed object" class. Including multiple classes could result in poor performance due to the limited data available for each class. Following the suggestion in the Faster R-CNN paper (Ren et al.; 2016), the value of B_{ROI} is set to 512. The BoxPredictor loss is defined by Equation 2.17, where N_{cls} and N_{reg} are defined as 1 and $B_{ROI} \times B$, respectively, where the latter refers to the total number of region samples used for loss computation. The focal loss parameters of L_{cls} in Equation 2.17 are initially set to $\alpha_t = 0.01$ and $\gamma = 4$.

During inference, a class score threshold of 0.1 is empirically set to filter out low-scoring bounding boxes predicted by the BoxPredictor. This threshold helps ensure the quality of the predicted bounding boxes.

3.3 Experiment Pipeline

To evaluate the performance of Siam R-CNN, a series of experiments are conducted. Initially, the two pretext tasks are trained to assess their performance and prepare the pre-trained weights for the downstream change detection task. Subsequently, Siam R-CNN is fine-tuned with the pre-trained weights obtained from the pretext experiments. Following that, an ablation study is carried out, where specific influential components of Siam R-CNN are removed during training to evaluate their impact on the overall system performance. Finally, the performance of Siam R-CNN is further evaluated by fine-

tuning it with the pre-trained transfer learning weights obtained from the specialization project (Nyegaarden; 2022).

3.4 Results and Analysis

3.4.1 Efficiency of SAS Dataset Annotation

As discussed in the section on ground truth data preparation (Section 3.1.1.1), some images contain additional SAS artifacts around the changed objects. The decision to include or exclude these artifacts as changes can potentially affect the change detection performance of the network. In these experiments, two annotated datasets, one with

Table 3.1: Effects of SAS dataset annotation on Siam R-CNN change detection performance

Annotation set	OA	Precision	Recall	F1	Kappa	AP@0.25	AP@0.5
With artifacts	99.808	0.579	0.554	0.503	0.549	0.417	0.202
Without artifacts	**99.835**	**0.636**	**0.599**	**0.555**	**0.551**	**0.471**	**0.241**

additional artifacts and one without, are used to train the Siam R-CNN architecture. Siam R-CNN has been pre-trained using pretext task 1 for these experiments. Table 3.1 summarizes the change detection results using the two annotation sets. The results suggest that using the set that only annotates the actual objects as changes yield better overall change detection performance. The intention behind annotating the additional artifacts was to address the class imbalance between changed and unchanged areas. However, due to the limited occurrence of artifacts within the SAS dataset, effectively learning the features associated with them becomes challenging. Consequently, the decreased performance can be attributed to the network's inability to accurately predict the surrounding artifacts of the deployed target. Figure 3.4 presents a visual comparison of the bounding box performance for a deployed cube from the Larvik A1 survey. The visual comparisons of bounding box performance display the ground truth bounding boxes in blue and the predicted bounding boxes in red. As observed in Figure 3.1b, Siam R-CNN trained with annotated artifacts fails to predict the ground truth sidelobe echo, thus supporting the findings from Table 3.1.

Consequently, the annotation set without the SAS artifacts will be evaluated for the remaining part of the ablation study.

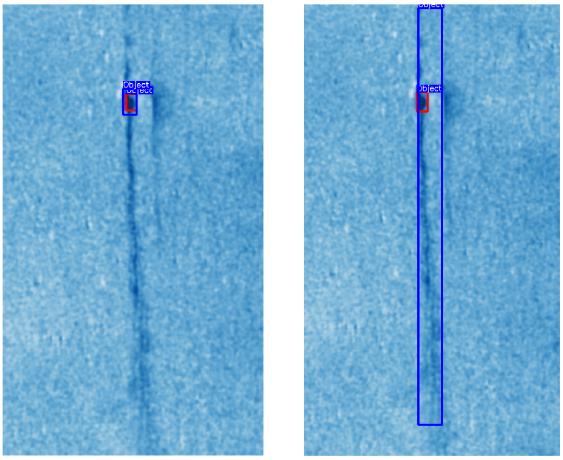


Figure 3.4: Visual comparison of Siam R-CNN bounding box performance applying different annotation sets. Ground truth boxes are shown in blue, and predicted bounding boxes are in red.

3.4.2 Efficiency of Self-Supervised Pre-Training

Pretext Task Performance. Table 3.2 reports the validation results for the pretext tasks. For pretext task 1, the average validation accuracy was consistently high: the model could correctly predict whether the patches were overlapping in 96.051% of the patch pairs. The average validation binary cross-entropy loss after 30 epochs was 0.107. Regarding pretext task 2, the average validation accuracy was 90.951% and a triplet-L1 validation loss of 0.108. The validation accuracy for pretext task 2 was determined by calculating the proportion of validation triplets where the feature distance between the anchor and positive patch was smaller than the distance between the anchor and negative patch, considering the additional margin γ.

These results suggest that the pretext tasks were relatively easy to solve. Furthermore, this indicates that the feature extractor effectively captures meaningful patterns

and structures within the SAS patch pairs, potentially enhancing performance in downstream change detection tasks.

Table 3.2: Performance of pretext tasks expressed in validation accuracy (%).

Pretext task	Validation loss	Validation accuracy
1	0.107	96.051
2	0.108	90.951

Change Detection Task Performance. The main objective of this thesis is to evaluate the effectiveness of self-supervised pre-training in improving SAS change detection performance. To accomplish this, Siam R-CNN is pre-trained using pretext task 1 and pretext task 2. Additionally, the change detection performance of the self-supervised pre-trained models is compared with that of a model trained without pre-trained weights.

Table 3.3: Effects of self-supervised pretext tasks on Siam R-CNN change detection performance

Pretext task	OA	Precision	Recall	F1	Kappa	AP@0.25	AP@0.5
No pretext task	99.634	0.582	0.597	0.514	0.507	0.401	0.167
1	**99.835**	**0.636**	0.599	**0.555**	0.551	**0.471**	**0.241**
2	99.817	0.618	**0.632**	0.554	**0.596**	0.458	0.226

The change detection results are presented in Table 3.3. Siam R-CNN performed best when pre-trained with weights from pretext task 1, showing an increase of 8.0% in the F1 score and 18% in AP@0.25 compared to not pre-training the network. Additionally, pretext task 2 demonstrated superior performance compared to not pre-training the network, improving F1 and AP values. These results indicate that Siam R-CNN benefits from the knowledge acquired during the self-supervised pre-training.

Figure 3.5 and Figure 3.6 present a visual comparison of the predicted change map and bounding boxes for two images from the SAS dataset. These images were carefully chosen from the Larvik A1 survey as they represent different levels of complexity in the seafloor environment. The first image presents a non-complex environment with easily

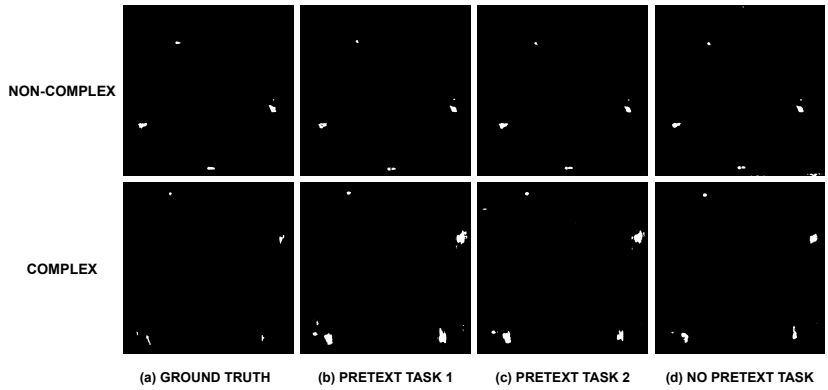


Figure 3.5: Visual comparison of Siam R-CNN change map performance applying different self-supervised pretext tasks

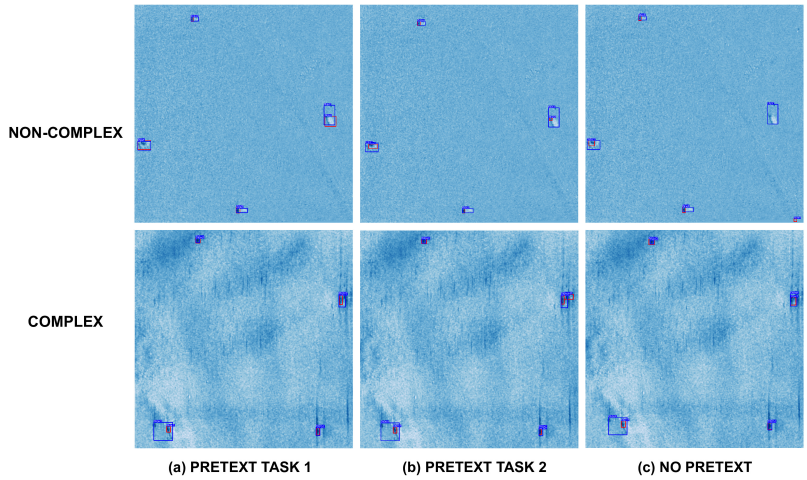


Figure 3.6: Visual comparison of Siam R-CNN bounding box performance applying different self-supervised pretext tasks. Ground truth boxes are shown in blue, and predicted bounding boxes are in red.

detectable targets. In contrast, the second image introduces additional complexities, including rocks and internal waves in the water column. Moreover, two of the targets in this image are occluded by sidelobe and multi-path echoes. Finally, both images feature a variety of deployed targets with different shapes and sizes, making them suitable for evaluating overall change detection performance. Therefore, these images are used throughout the thesis to visualize the results. Note that the images have been cropped to focus on the differences between ground truth and predicted changes.

The visual comparison further supports the qualitative evaluation metrics, indicating that pretext task 1 produces the most accurate change map and bounding box predictions. In Figure 3.6, it can be observed that the predictions of Siam R-CNN from all three experiments struggle to detect the light shadow following the deployed targets. Specifically, Siam R-CNN without pre-trained weights (third column) fails to predict the deployed glider on the right side of the first image and produces a false alarm in the bottom right corner. Among all targets, the deployed glider exhibits the smallest object-to-shadow ratio, making it particularly difficult for Siam R-CNN to detect without pre-training. The false alarm in Figure 3.6 is presented as a number of falsely predicted unchanged pixels (false negatives) in the resulting change map displayed in the third column of Figure 3.5. This occurrence can be attributed to a white line present at the bottom of the reference-pass image, introduced as an attempt to align it with the repeat-pass image that covers a larger area. This white line represents an irrelevant change that is falsely detected by the network lacking self-supervised pre-training.

Based on these findings, pre-trained weights from pretext 1 will be the focus of further analysis in subsequent experiments.

3.4.3 Efficiency of Multi-Task Learning

In this set of experiments, the efficiency of the multi-task architecture is evaluated and compared to the performance of the individual tasks performed independently. First, Siam R-CNN is trained with a multi-task approach, incorporating MaskHead and BoxHead for change map and bounding box predictions. Then, Siam R-CNN is trained in a single-task manner, where MaskHead and BoxHead are trained separately without shared feature representations.

It is expected that learning the change map task jointly with the bounding box detection task can be mutually beneficial since both tasks have a large overlap in what

needs to be learned. Table 3.4 demonstrates that sharing representations between the MaskHead and BoxHead improves performance in both tasks. The F1 and AP@0.25 scores increase by 8.4% and 170%, respectively, compared to the single-task framework. The significant improvement in bounding box performance can be attributed to the integrated attention mechanism. By leveraging the learned change maps, the model is able to enhance its ability to generate more precise bounding box predictions. The attention mechanism effects will be further evaluated in Section 3.4.6.

Table 3.4: Effects of multi-task learning on Siam R-CNN change detection performance

Tasks	OA	Precision	Recall	F1	Kappa	AP@0.25	AP@0.5
Single-task	99.819	0.598	0.507	0.512	0.587	0.175	0.055
Multi-task	**99.835**	**0.636**	**0.599**	**0.555**	**0.551**	**0.471**	**0.241**

Figure 3.7 and Figure 3.8 provide a visual comparison of the change map and bounding box performance. In Figure 3.8, it is evident that the single-task trained network (second column) faces difficulties in detecting the deployed objects, in contrast to the multi-task network. Specifically, the single-task network fails to detect the deployed cube located in the upper left corner of the second image. Instead, it predicts a false alarm of irrelevant seafloor changes. Moreover, when studying the first image in Figure 3.8, the single-task network only manages to detect the darkest areas of the deployed targets, consistently failing to identify the brighter shadow. These observations support the multi-task bounding box performance enhancement presented in Table 3.4.

Conversely, in Figure 3.7, there are no obvious differences in change map performance between the multi-task and single-task networks. Both networks successfully predict the deployed targets with little presence of falsely predicted pixels. This suggests that the change map task is not as dependent on the shared multi-task framework compared to the bounding box task. This is substantiated by the relatively smaller increase in the multi-task change map evaluation metrics, as shown in Table 3.4.

3.4.4 Efficiency of Multi-Task Loss Weighting

Appropriate weighting of the multi-task loss components in Equation 2.18 plays an important role in achieving accurate and reliable change detection results. This thesis

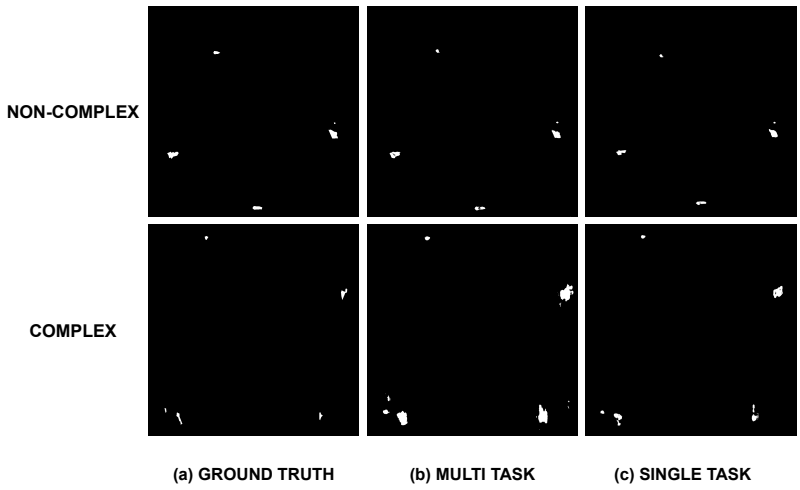


Figure 3.7: Visual comparison of single-task and multi-task change map performance

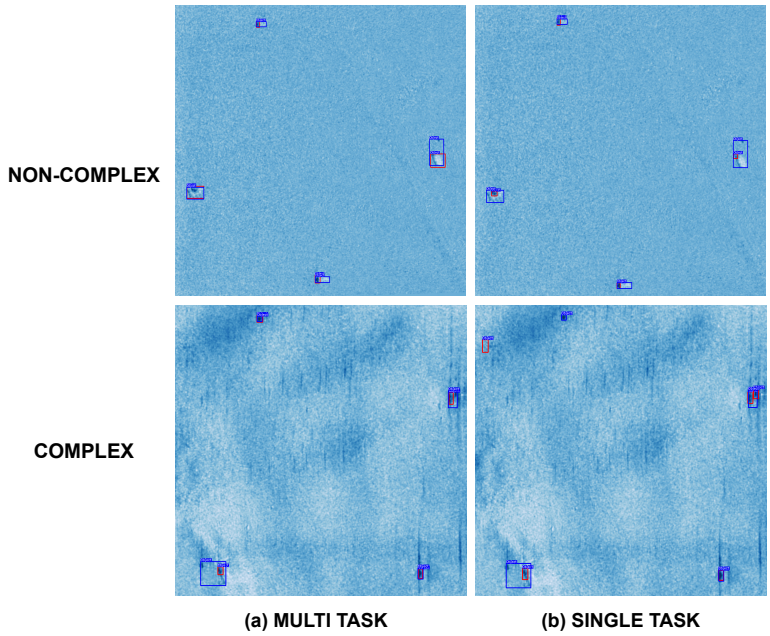


Figure 3.8: Visual comparison of single-task and multi-task bounding box performance. Ground truth boxes are shown in blue, and predicted bounding boxes are in red.

explores two specific weighting schemes, GradNorm and Dynamic Weight Averaging (DWA), explained in detail in Section 1.2.6. To evaluate the effectiveness of multi-task loss weighting, Siam R-CNN is trained using three different methods: GradNorm, DWA, and equally assigned weights.

Table 3.5 presents the experimental results for the Siam R-CNN architecture across all loss function weighting schemes. The results presented in Table 3.5 emphasize the importance of automatically adjusting the weights for multiple loss components. Specifically, the best-performing weighting scheme, DWA, improves the F1 score and AP@25 values by 6.7% and 5.8%, compared to the model trained with equally assigned weights.

Table 3.5 demonstrates a slight performance gap between the DWA and GradNorm weighting schemes. DWA dynamically adjusts the loss weights based on the relative losses between tasks. By assigning higher weights to tasks with larger loss gradients, DWA implicitly allows tasks to learn at different speeds. In contrast, GradNorm aims to balance the gradients across tasks, ensuring that each task receives equal attention during training. Considering the superior performance demonstrated by DWA, it appears that the multi-task framework benefits from different learning rates between tasks. This can be explained by a task imbalance between the change map and the bounding box tasks, where the shallower BoxHead network might require more training time and consequently lower learning rates than the MaskHead architecture.

Based on these findings, DWA is applied to adjust the multi-task loss components in the rest of the ablation studies.

Table 3.5: Effects of multi-task weighting scheme on Siam R-CNN change detection performance

Weighting scheme	OA	Precision	Recall	F1	Kappa	AP@0.25	AP@0.5
Equal Weights	99.745	0.604	0.597	0.520	0.523	0.445	0.218
DWA (T = 2)	**99.835**	**0.636**	0.599	**0.555**	**0.551**	**0.471**	**0.241**
GradNorm (α = 1.5)	99.783	0.604	**0.640**	0.549	0.514	0.460	0.223

3.4.5 Efficiency of Loss Functionality

This thesis proposes a focal loss function to address imbalanced classes and obtain better classification results. The focal loss in Equation 2.6 is applied to optimize all three classification tasks in $\mathcal{L}_{RPN}$, $\mathcal{L}_{box}$, and $\mathcal{L}_{mask}$ in Equation 2.18. To verify the efficiency of using focal loss, the results are compared to optimizing Siam R-CNN with binary cross-entropy loss. This is equivalent to changing the value of the focal loss parameter γ to 0. The results are summarized in Table 3.6. These results indicate that when employing binary cross-entropy loss, Siam R-CNN consistently predicts the majority class of "not changed" in nearly all scenarios. The higher values of overall accuracy (OA) and Kappa score can be attributed to a substantial proportion of true negative predictions in Equation 3.3 and 3.5, respectively. This emphasizes the significance of incorporating additional evaluation metrics, such as the F1 score, to obtain an accurate measure of the model's performance. Additionally, the results demonstrate the significant impact of class imbalance on change detection performance, emphasizing the necessity of employing a weighted loss function, such as focal loss, to effectively address this issue.

Table 3.6: Effects of focal loss parameters on Siam R-CNN change detection performance

Loss Functionality	OA	Precision	Recall	F1	Kappa	AP@0.25	AP@0.5
Binary cross-entropy loss	**99.988**	0.000	0.000	0.000	**0.964**	0.001	0.000
Focal loss	99.835	**0.636**	**0.599**	**0.555**	0.551	**0.471**	**0.241**

3.4.6 Efficiency of Attention Mechanism

Table 3.4 indicates that multi-task learning enhances the Siam R-CNN change detection performance. However, a more significant improvement is observed when combining it with the attention mechanism, as demonstrated in Table 3.7. The implemented attention mechanism accelerates the multi-task learning process by leveraging the learned change map to make more precise bounding box predictions. This substantially increases the AP@0.25 and AP@0.5 metrics, with improvements of 68% and 80%, respectively. The attention mechanism also has a positive impact on the change map performance. Specifically, the F1 score increases by 8.8%. By enabling the attention mechanism, BoxHead is able to generate more accurate bounding boxes, thereby reducing the bounding box

loss. Consequently, the multi-task weighting scheme adjusts the weightings to prioritize minimizing the change map prediction loss. This adjustment effectively enhances the overall performance of the change map task.

Table 3.7: Effects of attention mechanism on Siam R-CNN change detection performance

Attention mechanism	OA	Precision	Recall	F1	Kappa	AP@0.25	AP@0.5
With	**99.835**	**0.636**	0.599	**0.555**	**0.551**	**0.471**	**0.241**
Without	99.739	0.622	**0.695**	0.510	0.539	0.280	0.134

Figure 3.9 compares the Siam R-CNN bounding box performance with and without the attention mechanism. The visual comparison in Figure 3.9 specifically focuses on comparing the bounding box predictions with the corresponding ground truth, considering that the attention mechanism brings about the most significant improvement in bounding box performance. Figure 3.9 verifies the performance gap of the two networks in predicting the deployed targets. The network trained without the attention mechanism (second column) fails to detect the cube in the upper left corner of the second image. In contrast, the attention-trained network successfully predicts the cube. Furthermore, studying the predictions made by the attention-free Siam R-CNN reveals consistently less accurate bounding boxes that only encompass small areas of the deployed targets. This is particularly noticeable in the predicted bounding boxes of the bag and glider, positioned on the left and right sides of the first image in Figure 3.9.

3.4.7 Efficiency of Augmentation

Due to the large number of parameters in the network, it is essential to use data augmentation techniques to prevent overfitting and enhance the system's generalization ability. In previous experiments, the images were augmented by randomly rotating and flipping them in horizontal and vertical directions. Table 3.8 demonstrates the effectiveness of augmenting the SAS dataset. These results emphasize the significance of implementing data augmentation techniques to improve overall performance, revealing an increase of 8.2% in the F1 score and 1.9% in AP@0.25.

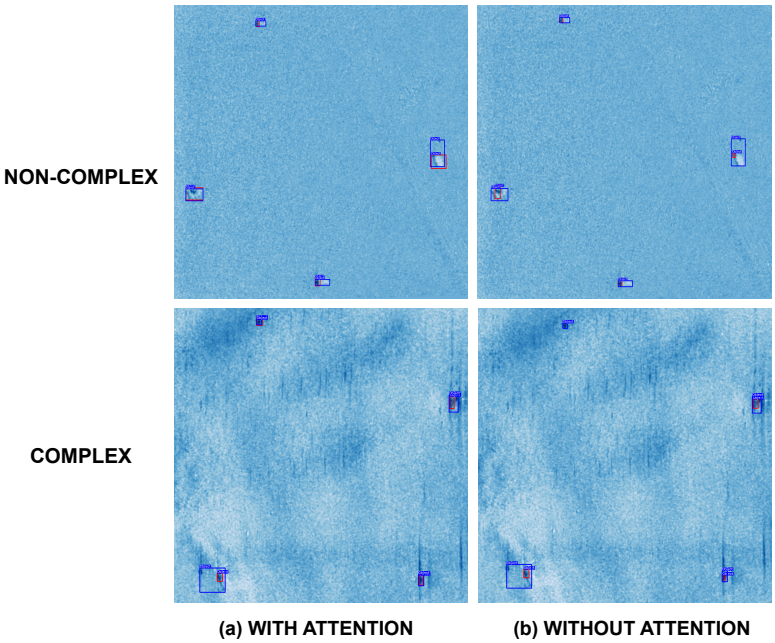


Figure 3.9: Visual comparison of Siam R-CNN bounding box performance with and without attention mechanism. Ground truth boxes are shown in blue, and predicted bounding boxes are in red.

Table 3.8: Effects of augmentation on Siam R-CNN change detection performance

Augmentation	OA	Precision	Recall	F1	Kappa	AP@0.25	AP@0.5
With	99.835	**0.636**	**0.599**	**0.555**	0.551	**0.471**	**0.241**
Without	**99.938**	0.631	0.545	0.513	**0.703**	0.462	0.239

3.5 Comparison

The following sections will compare the Siam R-CNN architecture and self-supervised pre-training technique to existing SAS change detection methods.

3.5.1 Comparison to Deep Learning-Based CD Methods

Currently, no publicly available deep learning-based change detection methods are specifically designed for SAS imagery. One possible explanation is the absence of publicly available benchmark datasets containing temporal SAS data. These datasets are crucial for researchers to develop competitive network architectures and participate in competitions and publications. Benchmark image datasets such as ImageNet, PASCAL VOC, and COCO contribute to the frequent development of state-of-the-art object detection models aiming to achieve superior performance. That being the case, it is difficult to compare the Siam R-CNN architecture to other deep learning-based change detection methods. However, one of the main objectives of this thesis is to serve as a starting point for future comparisons and provide a framework for future research in this area.

In the specialization project (Nyegaarden; 2022), a Siamese U-Net++ architecture was pre-trained on a larger dataset of very high-resolution (VHR) training data. This transfer learning approach improved change detection performance compared to training the network with randomly initialized weights. To further evaluate the self-supervised pre-training technique proposed in this thesis, the following experiments compare using self-supervised pretext weights with transfer learned VHR weights for pre-training the Siam R-CNN architecture. In the specialization project, the transfer learning approach involved using a larger filter size of $\{64, 128, 256, 512\}$ compared to the filter size of $\{32, 64, 128, 256\}$ in the Siam R-CNN architecture. To make use of the transfer learned weights, the filter size of Siam R-CNN is adjusted accordingly. Table 3.9 presents a qualitative comparison of the different pre-training methods, including pretext task 1, pretext task 2, and the transfer learning technique presented in the specialization project.

The results in Table 3.9 demonstrate that pre-training the Siam R-CNN model with self-supervised weights from pretext task 1 yield the highest overall performance compared to the other two methods. Specifically, pre-training of pretext task 1 leads to an

Table 3.9: Comparison of SAS change detection performance with different pre-training techniques

Pre-training technique	OA	Precision	Recall	F1	Kappa	AP@0.25	AP@0.5
Pretext task 1	**99.835**	**0.636**	0.599	**0.555**	0.551	**0.471**	**0.241**
Pretext task 2	99.817	0.618	**0.632**	0.554	**0.596**	0.458	0.226
Transfer learning	99.816	0.610	0.593	0.522	0.595	0.448	0.170

improvement of 6.3% and 5.1% in the F1 and AP@0.25 scores, respectively, compared to the transfer learning technique. Additionally, pretext 2 demonstrated superior performance compared to the transfer learning technique, improving F1 and AP values. These results can be attributed to the substantial domain differences between the VHR and SAS datasets encountered in the transfer learning approach. The self-supervised pretext tasks effectively address these domain discrepancies by pre-training the model using an unlabeled SAS dataset.

However, multiple factors suggest that the transfer learning technique has a more favorable starting point for achieving better change detection performance than the self-supervised methods. Firstly, in the transfer learning approach, both the encoder (FeatureNet) and decoder (MaskHead) are pre-trained. The self-supervised pretext tasks, however, are specifically designed for training only the encoder module of the network. Therefore, a slight performance enhancement could be expected by the additional pre-training of MaskHead. Secondly, the transfer learned Siam R-CNN uses a larger filter size compared to the self-supervised pre-trained framework. This allows the network to learn more complex and high-level features, leading to improved change detection performance.

These considerations emphasize the remarkable performance superiority of the pretext tasks, yielding promising results for developing self-supervised pre-training techniques for SAS change detection.

Figure 3.10 and Figure 3.11 present a visual comparison confirming the qualitative results. Specifically, Figure 3.11 demonstrates that the transfer learning technique fails to detect the deployed glider situated in the bottom left corner of the second image, whereas the self-supervised pre-trained networks accurately predict a small region of the glider.

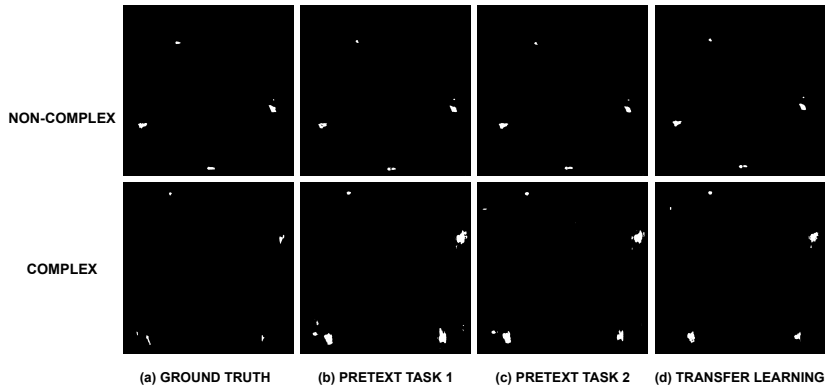


Figure 3.10: Visual comparison of Siam R-CNN change map performance applying different pre-training techniques

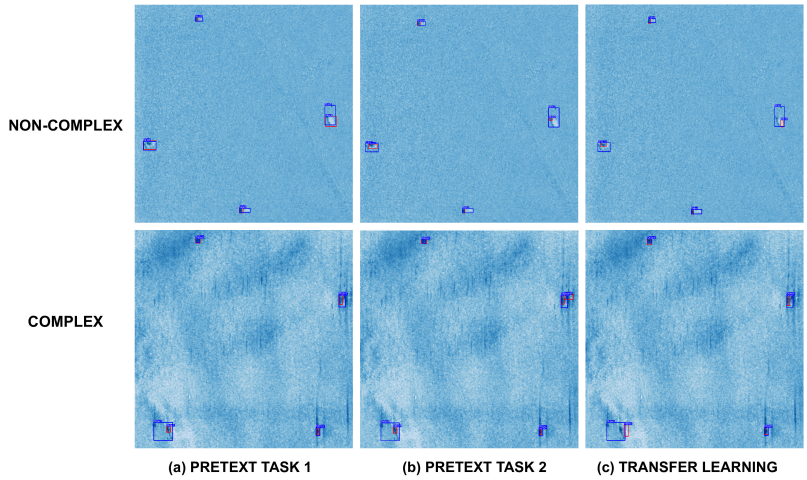


Figure 3.11: Visual comparison of Siam R-CNN bounding box performance applying different pre-training techniques. Ground truth boxes are shown in blue, and predicted bounding boxes are in red.

3.5.2 Comparison to Traditional CD Methods

Section 1.2.3 outlines some of the most promising automatic change detection methods (ACD) for SAS imagery. While several comparable ACD methods have been developed for mine-hunting, they use different SAS datasets that complicate a meaningful comparative analysis. The main reason for this is the absence of benchmark datasets that include temporal SAS data. Existing defense-related research papers use SAS data with classified mine information, which cannot be shared for comparison purposes. This creates difficulties in comparing Siam R-CNN with traditional change detection methods.

However, as mentioned in Section 1.1, the research paper by Midtgaard (2018) presents an ACD method for naval mine hunting. This method uses HISAS-1030 sonar data collected during the MANEX'14 sea trials. Three survey images used to evaluate the ACD method are also a part of the SAS dataset used to evaluate Siam R-CNN in this thesis. These images originate from the survey area in Bonassola Bay, where each reference pass image contains a deployed ballast.

The reported results in the research paper by Midtgaard (2018) represent a fusion of results obtained from multiple passes of the same deployed target. This multi-view fusion approach helps eliminate many false alarms observed in single-view surveys. However, the SAS dataset used in this thesis only includes single-pass repeat and reference surveys, making it unsuitable for direct comparison with the multi-view ACD method. To address this limitation and enable comparison, single-pass results have been obtained from the author of Midtgaard (2018). The single-pass results are presented in Table 3.10. In this table, the reference and repeat-pass image pairs are indexed numerically. The confidence value in the table represents a numerical measure indicating the level of confidence associated with a specific detection. The findings in Table 3.10 demonstrate that two out of three ballasts were successfully detected in the single-pass survey.

To assess and compare the performance of the ACD technique with the Siam-RCNN framework, Siam R-CNN is evaluated on the Bonassola Bay image pairs. Table 3.11 presents the numerical results obtained from Siam R-CNN, while Figure 3.12 visually represents the bounding box predictions generated by Siam R-CNN for the three image pairs.

The evaluation demonstrates that Siam R-CNN successfully detects all three deployed

Table 3.10: ACD results from a single-pass survey in Bonassola Bay

SAS image pair	Deployed target	Confidence value	Detection
1	Ballast 2	0.86	True
2	Ballast 2	N/A	False
3	Ballast 1	0.46	True

objects without any false alarms. In contrast, the ACD technique fails to identify the ballast present in the second image. These findings indicate that Siam R-CNN achieves superior performance on the three Bonassola data samples. However, it is important to note that these results alone do not provide a conclusive assessment of the overall performance of the two techniques, given the extremely limited and homogeneous nature of the data. Therefore, a comparison with a larger and more diverse SAS dataset is necessary to comprehensively assess the performance. Nonetheless, these results indicate that Siam R-CNN is a promising and competitive alternative to traditional methods for change detection in SAS imagery.

Table 3.11: Siam R-CNN results from a single-pass survey in Bonassola Bay

SAS image pair	Deployed target	Confidence value	Detection
1	Ballast 2	0.32	True
2	Ballast 2	0.22	True
3	Ballast 1	0.25	True

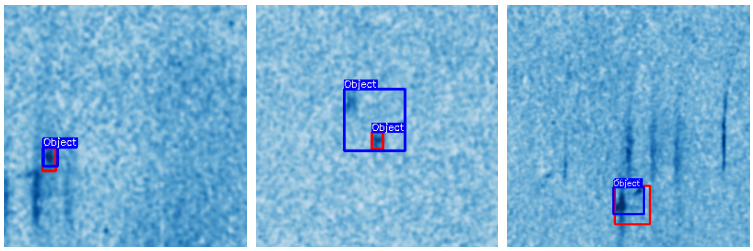


Figure 3.12: Visual results of Siam R-CNN bounding box performance from a single-pass survey in Bonassola Bay

Chapter 4

Conclusions and Future Work

This master's thesis introduces a novel self-supervised learning framework called Siam R-CNN for change detection of SAS imagery. Siam R-CNN is specifically designed to overcome the challenges associated with a limited labeled SAS dataset. The framework leverages unlabeled SAS data to pre-train the network using pretext tasks that exploit the temporal consistency between images. Additionally, the framework incorporates a multi-task attention-based design that simultaneously learns object-based bounding box change detection and pixel-based change map generation.

The experimental results demonstrate a significant performance improvement when pre-training the network using the pretext tasks proposed in this thesis. Moreover, the self-supervised pre-training approach demonstrates superior performance compared to the transfer learning technique introduced in the specialization project (Nyegaarden; 2022).

The multi-task framework benefits both change detection tasks, resulting in more accurate predictions for change maps and bounding boxes. Notably, incorporating the attention mechanism achieves the most substantial performance gains, resulting in a remarkable increase of 68% and 80% in AP@0.25 and AP@0.5, respectively.

Finally, this thesis presents one of the first comparisons within the scientific community between a deep learning-based and traditional SAS change detection method. The results suggest that Siam R-CNN offers a promising alternative to existing ACD technology, reducing the reliance on human supervision and providing more accurate

change predictions.

4.1 Future Work

In this master's thesis, two pretext tasks were designed to pre-train the Siam R-CNN framework. These tasks aimed to learn feature representations that could benefit the feature extractor in downstream change detection tasks. Although these pretext tasks achieved satisfactory results, there are other promising pretext candidates (see Section 1.2.5) that have the potential to further enhance the change detection performance of Siam R-CNN. For future work, it would be valuable to evaluate pretext tasks that pre-train the entire network instead of just the feature extractor. For example, exploring self-supervised generative autoencoders that pre-train both the encoder and decoder components of the MaskHead framework could be worth investigating. This approach would enable the network to learn more comprehensive representations and potentially lead to improved change detection results.

One of the main challenges in SAS change detection is the lack of benchmark datasets that can be used to compare the performance of different change detection methods. This limitation makes it difficult to conduct a thorough comparison of the Siam R-CNN framework against alternative methods. However, with the rapid advancements in remote sensing technology, it is foreseeable that a benchmark SAS temporal dataset will be established for research purposes. This development would provide an opportunity for a more comprehensive evaluation of the Siam R-CNN framework. Additionally, the availability of a benchmark SAS dataset would enable modifications and extensions to the Siam R-CNN framework, facilitating the development of state-of-the-art change detection architectures. This would contribute to further advancements in the SAS change detection research.

References

Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L. and Ridella, S. (2012). The 'K' in K-fold Cross Validation, *ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* .

Bai, T., Wang, L., Yin, D., K, S., Chen, Y., Li, W. and Li, D. (2022). Deep learning for change detection in remote sensing: a review, *Geo-Spatial Information Science* pp. 1–27.

Callow, H. J., Hagen, P., Hansen, R., Sæbø, T. and Pedersen, R. B. (2012). A new approach to high-resolution seafloor mapping, *The Journal of Ocean Technology* **7**(2): 13–22.

Chen, H. and Shi, Z. (2020). A Spatial-Temporal Attention-Based Method and a New Dataset for Remote Sensing Image Change Detection, *Remote Sensing* **12**.

Chen, Z., Badrinarayanan, V., Lee, C. and Rabinovich, A. (2018). Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks, *ICML* .

Cipolla, R., Gal, Y. and Kendall, A. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 7482–7491.

Cohen, J. (1960). A coefficient of agreement for nominal scales, *Educational And Psychological Measurement* (20): 37–46.

CVAT.ai Corporation (2022). Computer vision annotation tool (cvat), `http://cvat.ai/`.

Everingham, M., Van Gool, L., Williams, C., Winn, J. and Zisserman, A. (2010). The PASCAL visual object classes (VOC) challenge, *Journal of Computer Vision* **88**(2): 303–338.

G-Michael, T., Marchand, B., Tucker, J., Marston, T., Sterhnlicht, D. and Azimi-Sadjadi, M. (2016). Image-Based Automated Change Detection for Synthetic Aperture Sonar by Multistage Coregistration and Canonical Correlation Analysis, *IEEE Journal of Oceanic Engineering* **41**(3).

Girshick, R. (2015). Fast R-CNN, *Proceedings of the IEEE international conference on computer vision* pp. 1440–1448.

Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation, *2014 IEEE Conference on Computer Vision and Pattern Recognition Sensing* pp. 580–587.

Guo, M.-H., Xu, T.-X., Liu, J., Liu, Z., Jiang, P., Mu, T., S-H, Z., Martin, R., Cheng, M. and Hu, S. (2022). Attention mechanisms in computer vision: A survey, *Computational Visual Media* **8**(3): 331–368.

Hansen, R. (2011). Introduction to Synthetic Aperture Sonar, *in Sonar Systems edited by N.Z.Kolev* pp. 3–28.

Hansen, R. (2022). Change detection on shipwrecks using synthetic aperture sonar, *Technical Report 22/02441*, FFI.

Hansen, R., Sæbø, T., Callow, H. and Synnes, S. (2011). Challenges in Seafloor Imaging and Mapping With Synthetic Aperture Sonar, *IEEE transactions on geoscience and remote sensing* **49**(10): 3677 – 3687.

He, K., Gkioxari, G., Dollár, P. and Girshick, R. (2017). Mask R-CNN, *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* .

He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep Residual Learning for Image Recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 770–778.

Jaturapitpornchai, R., Matsuoka, M., Kanemoto, N., Kuzuoka, S., Ito, R. and Nakamura, R. (2019). Newly built construction detection in SAR images using deep learning, *Remote Sensing* **11**(12): 1–24.

Jean, N., Wang, S., Samar, A., Azzari, G., Lobell, D. B. and Ermon, S. (2018). Tile2vec: Unsupervised representation learning for spatially distributed data, *AAAI* **33**: 3967–3974.

Jian, L., Pu, Z., Zhu, L., Yao, T. and Liang, X. (2022). SS R-CNN: Self-Supervised Learning Improving Mask R-CNN for Ship Detection in Remote Sensing Images, *Remote Sensing* **14**(17).

Jiang, H., Peng, M., Zhong, Y., Xie, H., Hao, Z., Lin, J., Ma, X. and Hu, X. (2022). A Survey on Deep Learning-Based Change Detection from High-Resolution Remote Sensing Images, *Remote Sensing* **14**: 1552.

Keskar, N. S. and Socher, R. (2017). Improving generalization performance by switching from Adam to SGD, *ArXiv* **abs/1712.07628**.

Khelifi, L. and Mignotte, M. (2020). Deep Learning for Change Detection in Remote Sensing Images: Comprehensive Review and Meta-Analysis , *IEEE Access* **8**.

Kingma, D. and Ba, J. (2015). Adam: A Method for Stochastic Optimization, *3rd International Conference for Learning Representations, San Diego* .

Kongsberg Maritime (2010). HISAS 1030 - High Resolution Interferometric Synthetic aperture sonar, `https://www.kongsberg.com/globalassets/maritime/km-products/product-documents/high-resolution-interferometric-synthetic-aperture-sonar---hisas-1030`.

Kongsberg Maritime (2022). Synthetic Aperture Sonar - HISAS, `https://www.kongsberg.com/ru/maritime/products/ocean-science/mapping-systems/sonars/SAS/`.

Leenstra, M., Marcos, D., Bovolo, F. and Tuia, D. (2021). Self-supervised pre-training enhances change detection in Sentinel-2 imagery, *International Conference on Pattern Recognition* pp. 578–590.

Li, K., Li, Z. and Fang, S. (2020). Siamese NestedUNet Networks for Change Detection of High Resolution Satellite Image, *2020 International Conference on Control, Robotics and Intelligent System (CCRIS 2020)* pp. 42–48.

Lin, T.-Y., Dollàr, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. (2017). Feature Pyramid Networks for Object Detection, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 2117–2125.

Liu, S., Johns, E. and Davison, A. (2019). End-to-End Multi-Task Learning with Attention , *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 1871–1880.

Loshchilov, I. and Hutter, F. (2017). Fixing Weight Decay Regularization in Adam, *ArXiv* **abs/1711.05101**.

Midtgaard, O. (2013). Change Detection Using Synthetic Aperture Sonar Imagery With Variable Time Intervals, *Proc. 1st Underwater Acoustic Conference, Jun. 2013* .

Midtgaard, Ø. (2018). Automated Change Detection in Streaming SAS Imagery, *Proceedings of the 4th International Conference on Synthetic Aperture Sonar Synthetic Aperture Radar* **40**(2).

Murugan, P. and Durairaj, S. (2017). Regularization and optimization strategies in deep convolutional neural network, *CoRR* pp. 1–15.

Nyegaarden, N. (2022). Transfer Learning with Deep Convolutional Neural Networks for Change Detection in Synthetic Aperture Sonar Imagery, *NTNU* .

Pang, Y., Xie, J., Khan, M. H., Anwer, R., Khan, F. and Shao, L. (2019). Mask-guided attention network for occluded pedestrian detection, *Proceedings of the IEEE International Conference on Computer Vision* p. 4967–4975.

Peng, B., Huang, Q., Vongkusolkit, J., Gao, S., Wright, D. B., Fang, Z. N. and Qiang, Y. (2021). Urban flood mapping with bitemporal multispectral imagery via a self-supervised learning framework, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **14**: 2001–2016.

Peng, D., Zhang, Y. and Guan, H. (2019). End-to-End Change Detection for High Resolution Satellite Images Using Improved UNet++, *Proceedings of the 4th International Conference on Synthetic Aperture Sonar Synthetic Aperture Radar* **40**(2): 115–122.

Perreault, H., Bilodeau, G.-A., Saunier, N. and Hèritier, M. (2020). SpotNet: Self-Attention Multi-Task Network for Object Detection, *17th Conference on Computer and Robotic Vision (CRV)* .

R. Padilla, S., Netto, L. and da Silva, E. A. B. (2020). A Survey on Performance Metrics for Object-Detection Algorithms,, *International Conference on Systems, Signals and Image Processing (IWSSIP)* pp. 237–242.

Ramkumar, V. R. T., Arani, E. and Zonooz, B. (2022). Differencing based self-supervised pretraining for scene change detection, *ArXiv* **abs/2208.05838**.

Ramkumar, V. R. T., Bhat, P., Arani, E. and Zonooz, B. (2021). Self-supervised pretraining for scene change detection, *35th Conference on Neural Information Processing Systems (NeurIPS 2021)* .

Ren, S., He, K., Girshick, R. and Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **39**(6): 1137–1149.

Ruder, S. (2016). An overview of gradient descent optimization algorithms, *ArXiv* **abs/1609.04747**.

Shafique, A., Cao, G., Khan, Z., Asad, M. and Aslam, M. (2022). Deep Learning-Based Change Detection in Remote Sensing Images: A Review, *Remote Sensing* **14**(4): 871.

Singh, A. (1989). Review article digital change detection techniques using remotely-sensed data, *International Journal of Remote Sensing* **10**(6): 989–1003.

Själander, M., Jahre, M., Tufte, G. and Reissmann, N. (2019). EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure.

Vandenhende, S., Georgoulis, S., Proesmans, M., Dai, D. and Van Gool, L. (2020). Revisiting multi-task learning in the deep learning era, *ArXiv* **abs/2004.13379**.

Wang, J., Yang, X., Yang, X., L, J. and Fang, S. (2020). Unsupervised change detection between SAR images based on hypergraphs, *ISPRS Journal of Photogrammetry and Remote Sensing* **164**: 61–72.

Wang, Q., Zhang, X., Chen, G., Dai, F., Gong, Y. and Zhu, K. (2018). Change detection based on Faster R-CNN for high-resolution remote sensing images, *Remote Sensing Letters* **9**(10): 923–932.

Wang, Y., Albrecht, C., Braham, N., Mou, L. and Zhu, X. (2022). Self-supervised learning in remote sensing: A review, *IEEE Geoscience and Remote Sensing Magazine* pp. 2–36.

Warakagoda, N. and Midtgaard, Ø. (2018). Transfer-learning with deep neural networks for mine recognition in sonar images, *Proceedings of the Institute of Acoustics* **40**(2).

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y. and Girshick, R. (2019). Detectron2, `https://github.com/facebookresearch/detectron2`.

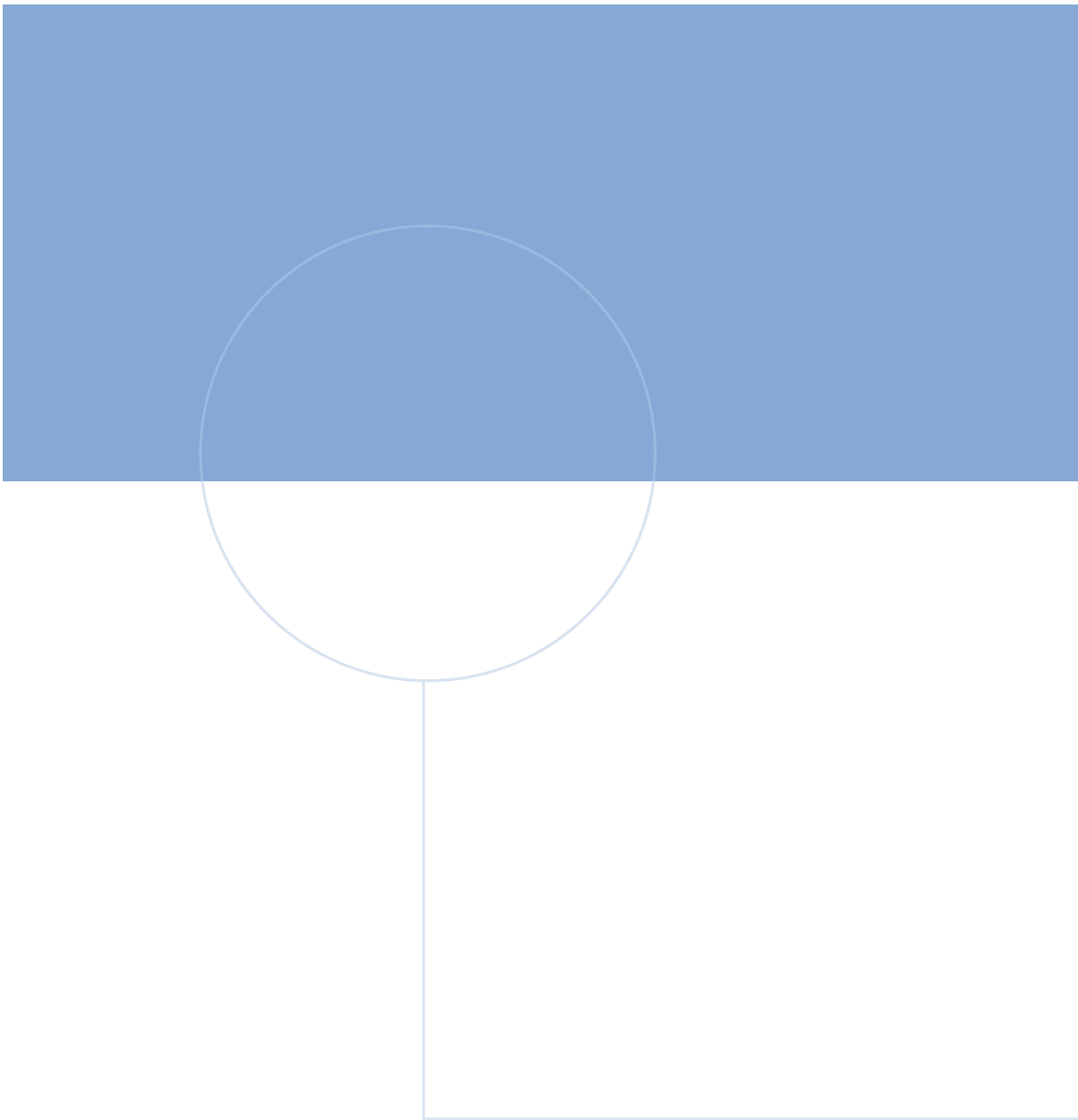
Zbontar, J., Jing, L., Misra, I., LeCun, Y. and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction, *Proceedings of the 38th International Conference on Machine Learning, PLMR* **139**.

Zhang, Y. and Yang, Q. (2017). A survey on multi-task learning, *IEEE Transactions on Knowledge and Data Engineering* **34**: 5586–5609.

Zhou, Z., Siddiquee, M., Tajbakhsh, N. and Liang, J. (2020). UNet++: Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation, *Journal of IEEE Transactions on Medical Imaging* **39**(6): 1856–1867.

Zhu, Q., Guo, X., Li, Z. and Li, D. (2022). A review of multi-class change detection for satellite remote sensing imagery, *Geo-spatial Information Science* pp. 1–15.

Zhu, W., Huang, Y., Zeng, L., Chen, X., Liu, Y., Qian, Z., Du, N., Fan, W. and Xie, X. (2019). AnatomyNet: Deep learning for fast and fully automated whole-volume segmentation of head and neck anatomy, *Medical Physics* **46**(2).



NTNU
Norwegian University of
Science and Technology

FFI Norwegian Defence
Research Establishment