UiT The Arctic University of Norway

Faculty of Science and Technology, Department of Physics and Technology

Affinity–Guided Image–to–Image Translation for Unsupervised Heterogeneous Change Detection

Mads Adrian Hansen

FYS–3941 Master's thesis in applied physics and mathematics – 30 ECTS – December 2019

Abstract

Change detection in earth observation remote sensing images can be used to describe the extent of natural disasters, e.g., forest fires and floods. When time is of the essence, the ability to utilize heterogeneous images is fundamental, i.e., images that are not directly comparable due to the sensors used or the capturing conditions.

The recent advances in machine learning have dispersed into the field of change detection in earth observation remote sensing images, and several methods utilizing machine learning principles have been proposed.

One promising paradigm to approach heterogeneous change detection from is paired image–to–image translation. If images captured with different sensors under varying conditions can be adequately mapped between their respective imaging domains to compare them directly, can changes be highlighted.

Performing change detection in an unsupervised setting is crucial for the current state of the art methods, as the inference models are trained to do change detection on one particular dataset, i.e., the models do not have generalization capabilities. A production system must thus be able to describe a current natural disaster without access to ground truth, i.e., it must perform an unsupervised sample selection to train the image–to–image translation maps.

Luppino *et al.* [2] proposed an unsupervised change detection method utilizing affinity norms, which was later improved in [1]. This affinity norm method was used to produce initial change maps (ICMs), used for sample selection in the training of two convolutional neural network (CNN) architectures: ACE-net and X-net [1]. These image–to–image translation CNNs were trained using a cross–domain loss term weighted with the ICM, and a loss term that enforces cyclic consistency.

Affinity matrices describe neighborhood structures and are used in computer vision to solve e.g., foreground–background separation problems. Inspired by the use of affinity matrices to produce initial change maps [2], we had the idea that affinity matrices could also be used during the training phase of the image translation CNNs. The core realization is that for an image $\boldsymbol{X} \in \mathcal{X}$ mapped with an image translation CNN $T_{\mathcal{X}} : \mathcal{X} \to \mathcal{Y}$ to produce $\hat{\boldsymbol{Y}} = T_{\mathcal{X}}(\boldsymbol{X})$, the affinity structure should be retained, i.e., the affinity matrix of $\boldsymbol{X}$ and $\hat{\boldsymbol{Y}}$ should be similar.

Based on this realization, we herein propose an affinity–guiding loss term for training paired image–to–image translation maps.

The loss term is used to train the Affinity–guided X-net (AX-net), and its performance is evaluated and compared to X-net [1] in an ablation study. Ablation studies are crucial for deep learning research [3] and aim to identify parts of a machine learning model that does not contribute to its inference. This ablation study aims to isolate the contribution of the three loss terms in the optimization of the CNNs.

The experimental results indicate that the affinity–guiding loss term is beneficial, but increases the optimization time significantly. Specifically, the affinity–guiding loss term can replace the cyclic consistency term. If that would be the case, one can consider simplifying the model by removing an entire CNN as the ability to cycle the image translation is not any longer needed.

Ablation studies are crucial for deep learning research
– François Chollet

Acknowledgments

I want to express my sincerest gratitude to you, Stian, for helping me find the way. For your able guidance. For your patience and determination in helping me complete this thesis. Further, I extend my gratitude to you, Luigi, for our discussions, your know-how, and your Italian temper. It has been a pleasure working with both of you.

I also want to thank Gabriele and Filippo for the fruitful discussions leading to the methodology proposed in this thesis. And thank you, Thomas, for the technical guidance and all pointers in the right direction.

To Daniel and Andreas: thank you for our friendship. Thank you for the inspiration you have given me, the discussions we have had, our collaboration, and everything you have taught me. Thank you for the grit you have forced me to show to keep up with you. You have been invaluable.

I also want to express due gratitude towards the members of the UiT Machine Learning Group, and the other faculty and fellow students who have contributed to my journey.

Lastly, to Amalie and the rest of my family: thank you for love and support.

Mads Adrian Hansen
Tromsø, December 2019

Contents

List of Figures

List of Tables

List of Algorithms

Notation

$\mathcal{X} \subseteq \mathbb{R}^{h \times w \times c_x}$	Domain, e.g., feature space
$\boldsymbol{X} \in \mathcal{X}$	Matrix
$T_{\mathcal{X}} : \mathcal{X} \to \mathcal{Y}$	Map from $\mathcal{X}$ to $\mathcal{Y}$
$\hat{\boldsymbol{Y}} = T_{\mathcal{X}}(\boldsymbol{X})$	Matrix, result of transformation from $\mathcal{X}$ to $\mathcal{Y}$
$\{1, \dots, n\}$	A set with n elements
$\mathcal{O}(1, N, log(N), N^2, \dots)$	Big O, time complexity or memory consumption
$(g \circ f)(x) = g(f(x))$	Function composition of g and f
$\boldsymbol{A} \odot \boldsymbol{B}$	Element-wise (Hadamard) product of $\boldsymbol{A}$ and $\boldsymbol{B}$
$\oplus$	Addition like operation, e.g., a weighted average

The notation $\mathbb{E}_X[\cdot]$ is a convenient shorthand for $\mathbb{E}_{X \sim p_{\text{data}}(x)}[\cdot]$, where p_{data} is the distribution of the training data, and will be used throughout.

Abbreviations

Adam	Adaptive Moment Estimation
CD	Change Detection
CNN	Convolutional Neural Network
cGAN	conditional Generative Adverserial Network
AE	AutoEncoder
EM	Expectation Maximization
EO	Earth Observation
FCM	Fuzzy C-Means
FLICM	Fuzzy Local Information C-Means
GAN	Generative Adverserial Networks
GKIT	Generalized Kittler-Illingworth threshold
KIT	Kittler-Illingworth threshold
I2I	Image–to–Image
ICM	Initial Change Map
MDS	MultiDimensional Scaling
MRF	Markov Random Field
PCC	Post Classification Comparison
RS	Remote Sensing
SCCN	Symmetric Convolutional Coupling Networks
SGD	Stochastic Gradient Descent
SAR	Synthetic Aperture Radar
VAE	Variational AutoEncoder

1. Introduction

This section describes the motivation of the thesis. It explains the societal impact of change detection in heterogeneous remote sensing images for earth observation and the added value of being able to utilize heterogeneous images for this purpose. It also gives an overview of the technical challenges that we are faced with in this endeavor. The main hypothesis to be tested and the central research tasks are presented. An overview of the thesis is laid out, and the main contributions are highlighted.

1.1. Motivation

Nature is in flux, as natural and human–caused trends and events changes the surface of the Earth. Change detection encompasses the quantification of such temporal phenomena [4]. Observing and describing change on the surface of the Earth have many practical uses such as environmental monitoring including desertification, deforestation and glaciers, landscape monitoring including urban areas, forests and wetland, and monitoring of natural disasters including forest fires, drought, floods and landslides [5]. Change detection systems are designed to automatically identify changes between satellite images captured in the same area at different times.

For change detection after natural disasters the response time is of the essence, and it is desirable to utilize images that are captured as close in time to the event as possible. This often means using images captured with different sensors, i.e., heterogeneous images. This poses two technical challenges, as the images cannot be directly compared, and no ground truth is available to train the system, i.e, the system must be trained in an unsupervised manner.

In the last decade, mapping of change phenomena in multi-source remote sensing images has gained increasing attention, and the algorithmic breakthroughs in machine learning have accelerated this. Nevertheless, unsupervised change detection in heterogeneous images is still very much an open research topic, and several approaches based on image–to–image translation have been proposed [1, 2, 6–9] in the last couple of years.

Affinity matrices encode pixel-to-pixel similarities in images. It was recently shown that affinity information can be used to produce decent change maps on its own, which have further been used as priors to guide unsupervised training [1, 2]. The core hypothesis of this thesis is that affinity information can also be used more explicitly to directly guide the training of image translation maps as part of the loss function. The core insight motivating the proposed loss term is that the affinity information should be consistent through an image-to-image translation.

1.2. Hypothesis

The hypothesis of this Master's thesis is:

> *It is useful to explicitly include affinity information in the loss function when training paired image-to-image translation models for change detection in earth observation remote sensing images.*

1.3. Research Tasks

Hypothetico-deductive method is used in an attempt to falsify this hypothesis. In this process the problem has been decomposed into the following research tasks:

- Introduce the difference of the affinity matrices of the cross-domain images as a loss term in the training of paired image-to-image translation networks;
- Use this affinity–guiding loss term as part of the X-net model [1];
- Perform an ablation study to decompose the contributions of the various loss terms in the affinity–guided X-net;
- Evaluate whether affinity guiding is beneficial. The desired improvement is to increase the average κ-score without adding much variance or greatly increase the time or memory complexity, compared against X-net[1];
- Retain the relatively simple loss function in [1].

1.4. Organization of the Thesis

This thesis is organized in three main parts, which again are divided into sections.

Part I offers a summary of background theory related to the proposed affinity–based image–to–image translation loss term and change detection model. Change detection is defined in Section 2, and described in the context of heterogeneous remote sensing in earth observation images. A summary of related works is also provided, with emphasize on machine learning methods. Section 3 provides definitions of proximity measures, and a description of proximity graphs and affinity matrices. Section 4 gives a brief introduction to the machine learning methodology necessary to understand the proposed model and the related models presented in Section 2.

In Part II is first the affinity–guiding loss term proposed in Section 5. A description of X-net [1] and its affinity–guided extension follows in Section 6, which also contains a subsection with implementational details.

In Part III is first the experimental setup for the ablation study presented in Section 7. A description of the remote sensing earth observation datasets used follows in Section 8. In Section 9 are results and observations from the ablation study presented.

Part IV offers some concluding remarks and ideas for future work.

1.5. Contributions

My contributions through this thesis and my work with it includes:

- A considerable contribution in discussions on ongoing change detection projects in the research group.
- Formulation of the research questions for this thesis.
- Propososition of the affinity–guiding loss term for image–to–image translation problems.
- Implementation of the affinity–guided X-net[1] model.
- Proposition of an ablation study in order to understand the contributions of the loss terms in the proposed change detection model.
- Ported large parts of the code for the project from TensorFlow 1.04 to TensorFlow 2.0. It is now an extensive code-base with good documentation, which is used in several ongoing research projects in the research group.

[1] The code is available at github.com/MadsAdrian/MastersThesis.

Part I – Technical Background

2. Change Detection

Detecting changes in images is conceptually simple. Radke *et al.* [10] describes the goal of change detection as identifying pixels that are "significantly different" between a pair of images of the same scene taken at different times. Although this describes the algorithmic goal, it does not describe the goal of the change detection process. The underlying assumption is that some event has caused the scene or some object in it to change, and the ultimate goal is to describe or quantify the extent of this change. Furthermore, identifying pixel differences presuppose that the only significant differences between the images are caused by the event of interest, which does not account for e.g. noise. Thus, there is a fundamental difference between identifying changes in the pixels representing an object and changes in the objects state. Singh [11] phrases change detection as *the process of identifying differences in the state of an object or phenomenon by observing it at different times.* This better captures the subtlety of the problem. Inspired by these formulations, the following definition will be used for the context of this thesis.

Definition 1 (Change Detection) *Assume that an object or scene is in one state at time t_1 and in a different state at time t_2. Given a representation of the object or scene in domain $\mathcal{X}$ at time t_1 and in domain $\mathcal{Y}$ at time t_2,* change detection *is the process of identifying the differences in the representations due to the changed state of the object.*

Identifying such differences can help describe the changed state of the object. Change detection will herein be discussed in the context of remote sensing for earth observation. The applications of change detection in images are not limited to remote sensing, but can also be relevant for e.g. medical diagnosis and treatment, surveillance, civil infrastructure, underwater sensing and driver assistance systems [10].

In remote sensing for earth observation, the scene is typically an area on the surface of the Earth, and different natural or anthropogenic[2] events or effects can cause changes in the scene. Changes can be dramatic and abrupt, or more subtle and gradual, and change can thus be understood either in a binary fashion or as a continuum [4].

The changed state of the object can be the result of sudden events such as forest fires, drought, floods or landslides [5]. It can also be the result of diffuse, long term effects causing changes in land-use and land-cover, landscape, urban areas, forest or vegetation, wetland, or glaciers, or other

[2] Caused by humans

environmental changes such as desertification or deforestation. Change detection is also used for crop monitoring and shifting cultivation monitoring. Lu *et al.* [5] presents a comprehensive list of use-cases and references to relevant articles.

2.1. Resolution

Methods and algorithms are affected by spatial, spectral, temporal and thematic constraints [4]. Depending on the event or effect of interest, different temporal, spatial and spectral resolutions are needed. Temporal resolution refers to the length of the change interval $t_2 - t_1$, spatial resolution refers to the area each pixel represents, and spectral resolution refers to which electromagnetic frequency bands the sensor capture.

2.1.1. Temporal Resolution

There are limitations to the time resolution that can be achieved for remote sensing data. The revisit period of the sensor determines how frequently it is possible to acquire an image of a certain area. Weather conditions, seasonal changes, and other temporal effects further influence whether the event of interest can be observed in a pair of images captured at certain times.

Definition 1 considers bi-temporal change detection. This requires a pair of images, and is more apt for exploring sudden changes, e.g. the effects of natural disasters. For such scenarios, it is desirable that the images are captured as close up to the event as possible. This ensures that the detected changes are due to the event of interest, and not other factors.

Bi-temporal methods are generally indifferent to what image was captured first. "A change is a change", and without extra information it is not possible to determine which way the change happened. Semantic information can in some cases determine what way the change happened, but can be hard to incorporate into an automatic system. If a a forest fire is observed, burnt forest will make it evident what image was captured first. It will take years or decades for the forest to recover. For a flooding event, this is not as obvious, as the changes introduced by a flood can be restored in a matter of weeks.

Bi-temporal change detection can also be used to explore long term effects, but has limited value for this use case, as it is not possible to describe the rate of change. A related problem is *trend detection* or *change trajectories*, which can be phrased as multi-temporal change detection. The methodology and challenges are similar, but rather than a pair, a series of images is used. This is more apt to explore long term effects, as the rate of change can be described.

The concepts of bi-temporal or multi-temporal also relates to whether change is understood in a binary fashion or as a continuum. If change is understood as a continuum, the extent of the change is quantified on pixel level, with a description of how much the pixel is changed rather than if it is changed. This quantification is easier in the multitemporal case, and many multi-temporal methods can thus consider change as a continuum. A continuous description of change can be thresholded to produce a binary description.

For brevity, change detection will be discussed in the bi-temporal context for the remainder of this chapter. The descriptions and reasoning can often be extended to the multi-temporal framework, and trend detection can be interpreted as a series of change detection problems.

2.1.2. Spatial Resolution

The change detection capabilities of any system are intrinsically limited by the spatial resolution of the images [4]. The sensor must be able to capture the required features to detect the desired changes. If the physical extent/manifestation of a change is much smaller than the area a pixel represents, the change might not influence the pixel enough to change its value significantly.

A too high resolution can complicate change detection, as small errors in the registration[3] becomes more notable, and pixel signatures are more affected by the caption angle. In optical images can shadows cause issues, which is more notable with a high resolution. A coarser resolution will smooth the shadows.

Detecting e.g. deforestation of large forests can be done with a coarse resolution. For other applications, can a finer scale be necessary. A lower bound on the error margins will be similar in order of magnitude to the scale of the pixels. On change boundaries, residual misregistration at the below-pixel level commonly degrades the assessment of the change event [4].

2.1.3. Spectral Resolution

Different properties of the Earth's surface contribute to the reflection of energy in different bands [12]. This holds true for both passive and active sensors. Thus are there limitations to what changes can be detected with a certain sensor. If the physical property changed by an event or effect is not captured by the sensor, the change does not influence the representation of the scene.

Specifically, if an area in a scene truly belongs to class a at time t_1 and class b at time t_2, but the representation of a and b are indistinguishable in domain $\mathcal{X}$, the change cannot be detected by any system.

[3] Same pixel in both images representing the same location on earth. Same spatial resolution.

2.2. Representation Domains

The homogeneity or heterogeneity of the representation domains $\mathcal{X}$ and $\mathcal{Y}$ results in two main branches of change detection methodology. Homogeneity refers to both the kind of sensor used to capture the data, and the sensor configuration and physical conditions at the time of acquisition.

Homogeneous and heterogeneous change detection refers to change detection performed on homogeneous and heterogeneous data respectively. It does not describe the method per se, but rather which data the method is used on. Homogeneous change detection is the standard setup, but during the last decade have heterogeneous setups become more common [13, 14].

2.2.1. Homogeneous Change Detection

Homogeneous refers to data captured by comparable sensors under similar conditions. This can be understood as domains $\mathcal{X}$ and $\mathcal{Y}$ being the same, or at least *directly comparable*. Despite not fully describing these assumptions, unimodal[4] is used synonymous with homogeneous.

Lu *et al.* [5] suggests that homogeneous data should be captured by the same sensor with the same spectral and spatial resolution. Further, should the acquisitions be (near) annual to handle effects from external sources as Sun angle, and seasonal and phenological[5] differences. These criteria might be unnecessarily strict for the purpose of having comparable domains, however they indicate what measures are required to be able to call the data homogeneous.

These strong assumptions allows for fairly simple change detection methods. It is probably harder to make the data fully adhere to these strong assumptions, than detecting changes if the assumptions are fulfilled.

Whether two images acquired at different times can indeed be homogeneous is a philosophical discussion. However, data have successfully been assumed homogeneous with precise registration, radiometric and atmospheric calibration, and normalization [5]. These measures can be sufficient to make unimodal images homogeneous. This is probably the reason why the terms are used synonymously.

For homogeneous data it is assumed that the images have similar statistical characteristics [15]. This includes both the noise model and how each class is represented in each image. Furthermore, the introduction of new classes in one of the images could violate this assumption. For practical purposes, the assumptions for homogeneous change detection can be summarized

[4] I.e. one image modality
[5] Related to periodic biological phenomena

as the data being directly comparable after preprocessing. If the data is unimodal, but the conditions for the two acquisitions are very different, preprocessing might not be sufficient. These cases should be considered a heterogeneous problem.

2.2.2. Heterogeneous Change Detection

Heterogeneous data indicates that the domains $\mathcal{X}$ and $\mathcal{Y}$ are not directly comparable. This can be the result of highly variable unimodal conditions, or completely different sensors that capture different physical properties. The latter case is also referred to as bimodal, multimodal, multisource, multisensor, cross sensor, and information unbalanced data [2].

For unimodal images, it can be hard to preprocess away highly varying conditions. Such cases can be considered as a heterogeneous problem to alleviate the need for meticulous preprocessing. For multimodal images, it is not possible to eliminate with preprocessing the fact that different sensors measures different physical properties, and the representations lie in different domains.

Heterogeneous change detection is in general a much harder problem than homogeneous change detection, and the two problems requires different conceptual solutions. In heterogeneous change detection, the local statistics of the data can be radically different [13], and the limited information quality between heterogeneous images makes change detection difficult to accomplish [16].

Heterogeneous change detection have strong ties to domain adaptation, feature learning and image–to–image translation [1]. The dominant approach is to somehow map one or both of the images to another domain where direct comparison is possible. In this domain, homogeneous approaches can be applied, as the representations are assumed directly comparable. This involves obtaining maps either between the input domains, $T_{\mathcal{X}} : \mathcal{X} \to \mathcal{Y}$ and/or $T_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{X}$, or maps from the input domains to a common feature space $\mathcal{Z}$, $S_{\mathcal{X}} : \mathcal{X} \to \mathcal{Z}$ and $S_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{Z}$.

The main motivation for heterogeneous methods is data availability. The time resolution dictated by the revisit time of a given instrument may be insufficient in many applications. Data availability may be further constrained by for instance weather conditions (cloudiness preventing acquisition of optical images) or by practical constraints such as conflicts in acquisition scheduling due to other uses of the sensor. For sudden events, it is desirable to use images acquired as close in time as possible before and after the event [17].

For sudden events, it is important to exploit the first available acquisition of the area, independent of its modality [14]. Synthetic Aperture Radar

(SAR) images can be captured at night and with less sensitivity to weather and atmospheric conditions. The images captured closest in time to the event will thus often be taken with different sensors, with the typical case of one optical and one SAR image. Preevent optical images can be collected from archives, while the first postevent images are SAR images for technical reasons [13]. Thus is it desirable to compare images captured in different domains, and will typically have different statistical behavior and different noise models.

2.3. Heterogeneous Change Detection Pipeline

The challenge of heterogeneous change detection is twofold; a) the different sensors provide different descriptions of the same truth [18], and moreover b) said truth changes.

The pipeline for change detection methods can roughly be divided into three main steps:

a) preprocessing including calibration and registration,

b) change extraction including feature extraction, mapping between domains as well as difference image computation, and

c) postprocessing, i.e. change map computation.

2.3.1. Preprocessing

Preprocessing is essential to achieve good change detection [4]. The goals of the preprocessing process is to establish a more direct link between the data and the biophysical phenomena it represents [4]. This can include the steps image registration, removal of data acquisition errors and image noise, aligning the noise models of the images (e.g. logarithmic transformation on SAR images to obtain near-Gaussian class distributions) and/or normalization. Other steps are masking of contaminated (e.g. cloudy) and/or irrelevant (e.g. water bodies when looking at changes in vegetation) scene fragments [4]. This can be summarized as preparing the data for change detection by bringing the representations as close as possible to the truth they represent. According to Lu *et al.* [5] the most important preprocessing steps for change detection are multi-temporal image registration and radiometric and atmospheric corrections.

Accurate image registration is absolutely essential [4], and misregistration will most likely be detected as a change by many change detectors [14].

Definition 2 (Image Registration) *Image registration is the process of geometrically aligning two or more images of the same scene obtained at different times, from different viewpoints, and/or by different sensors [19].*

Zitová and Flusser [19] list four steps for typical image registration methods; feature detection, feature matching, transform model estimation, and image resampling and transformation. Deep learning approaches to image registration are increasingly common [20], and Dong *et al.* [21] propose a Super Resolution CNN (SRCNN) approach which can be used to recover spatial details from the higher resolution images after registration [22].

Other calibration steps attempts to align the images in the pixel feature space because a common radiometric response allows quantitative analysis of more images [4]. Inherent noise will affect the change detection capabilities of a system and can even create unreal change phenomena [4]. Absolute radiometric correction is not necessary for successfully change detection, but the radiometric properties of the subject image need to be adjusted to those of the reference image [4].

This thesis is mainly concerned with change extraction, and preprocessing will not be covered in more detail. The required preprocessing of the data used in the experiments are considered sufficient to solve the isolated problems at hand.

2.3.2. Change Extraction

The goal of the change extraction process is to produce a difference image or two images that can be reasoned about to produce the final change map. In difference image approaches [23], homogeneous data allows for direct, pixel-wise comparison of the images through pixel difference or ratio. For optical multi-spectral images, comparison is usually performed with a pixel by pixel, band by band image difference [14]. Due to the multiplicative nature of speckle noise, the image ratio is more common when working with SAR images [14].

For heterogeneous change detection, the change extraction process does in general include transfering the images to a domain where they are directly comparable. This domain can be either, or both, of the input domains, as well as a common high dimensional or categorical feature space. Approaches to do this include feature extraction [9], image–to–image translation [1] and pixel distribution transformations [24]. After the pair of images are brought to a homogeneous space, an image difference or ratio can be performed to to produce a *difference image* or *ratio image*. This image is the output of the change extraction process. The change map computation then involves thresholding the difference or ratio image to produce the binary *change map*.

Reasoning approaches generally belong to the class of Post Classification Comparison (PCC) [16, 25, 26], where the pixels of each image are classified to produce a categorical feature space. Arithmetic operations in this feature space is not suited to infer changes, and decision theory [27] is often used to produce change maps from this space.

2.3.3. Change Map Computation

Change detection techniques can, based on their output, be divided into two groups [5]: a) techniques that output a binary change map, indicating if a pixel is changed; and b) techniques that output a detailed 'from-to' change map, indicating pixel class membership at both times, and thus whether it is changed.

A from-to change map is higher order, and can be transformed into a binary change map if the class information is not needed. The change extraction process must provide the necessary information to produce the desired type of change map, and from-to change maps are typically associated with post classification comparison schemes.

For difference image approaches, the change map computation can include different filtering schemes to take neighborhood information into account, and finally thresholding of the filtered difference image.

2.3.3.1. Threshold Methods

A myriad of image thresholding methods exist. The most well known is probably Otsu's thresholding method [28]. The method aims at minimizing the intra-class variance, and is commonly implemented as a search for the midpoint between the modes of the pixel graylevel histogram.

Another common method is the Kittler-Illingworth threshold (KIT) [29], which minimize the error assuming the pixel graylevels are normally distributed. Moser and Serpico [30] later generalized this method to account for the non-Gaussian distribution of SAR amplitude ratio images, dubbed Generalized Kittler-Illingworth threshold (GKIT).

Other thresholding methods include the algorithms presented in [31–35]. Melgani and Bazi [36] proposed an ensemble approach where a majority vote of these five thresholding methods is used to determine the final change map. Others have used this ensemble approach with a subset of the five algorithms.

2.4. Related Work

There are several natural groupings of change detection methods. Methods are designed a) for either homogeneous or heterogeneous data, b) from traditional signal processing, statistical, or machine learning principles, and c) in a supervised or unsupervised fashion [1]. For brevity, only heterogeneous methods will be covered here. A relevabt assortement of supervised and unsupervised techniques are presented in this section, and for each paradigm, a selection of signal processing and machine learning methods are summarized. Emphasis is put on unsupervised deep learning methods

formulated for bitemporal multisensor images, as the methodology proposed in this thesis is put in this setting.

Different sensors provide different representations of the same truth. These representations are generally incomparable in the low-dimensional spaces where they are observed. However, the same underlying truth indicate that the multi-source data is comparable in some high dimensional feature space in where the truth lies [8]. This insight motivates two different approaches to heterogeneous change detection;

a) map the multi-source data into a high dimensional feature space for comparison. That is to construct maps $S_{\mathcal{X}} : \mathcal{X} \to \mathcal{Z}$ and $S_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{Z}$ where $\mathcal{Z}$ is the common feature space which is similar to the space in which the truth lie, or

b) use the assumption that this common feature space exists to construct maps between the low-dimensional observation spaces for comparison. The common feature space $\mathcal{Z}$ is implicitly or explicitly considered a latent space for the maps $T_{\mathcal{X}} : \mathcal{X} \to \mathcal{Y}$ and/or $T_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{X}$.

The goal is to transform the representations to a space where they can be directly compared. When direct comparison is possible, methods for homogeneous change detection can be used to extract the changes.

2.4.1. Supervised Methods

The first methods for heterogeneous change detection were supervised methods [2]. Under the assumption that some dependence exists between the unchanged areas of two registered, heterogeneous images, Mercier *et al.* [14] proposed a quantile regression approach to detect changes. Based on copula theory, the regression model aims at estimating the local statistics of the first image as if it had been observed with the acquisition conditions of the second image. In terms of Definition 1, the method yields an estimate of the local statistics of the image captured in $\mathcal{X}$ as if it was captured in $\mathcal{Y}$. A symmetric Kullback-Leibler-based [37, 38] comparison of these local statistics is applied to define a change measure.

Another approach that maps the images between the input domains is the Homogeneous Pixel Transformation (HPT) method proposed by Liu *et al.* [39]. This kernel regression scheme [2] consists of two operations; a forward transformation $T_{\mathcal{X}} : \mathcal{X} \to \mathcal{Y}$, and a backward transformation $T_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{X}$, which are trained in a supervised manner based on the unchanged pixels. To achieve noise tolerance, a weighted sum of each pixel's k-nearest neighbors is used to estimate the transformed pixel value. After transforming $\boldsymbol{X}$ to $\hat{\boldsymbol{Y}} = T_{\mathcal{X}}(\boldsymbol{X})$ and $\boldsymbol{Y}$ to $\hat{\boldsymbol{X}} = T_{\mathcal{Y}}(\boldsymbol{Y})$, a difference image $\boldsymbol{D} = \|\hat{\boldsymbol{X}} - \boldsymbol{X}\| + \|\hat{\boldsymbol{Y}} - \boldsymbol{Y}\|$ is computed. Fuzzy C-Means (FCM) [40] is used to cluster the pixels of the difference image into changed and unchanged

pixels, and a filter based on Dempster–Shafer theory [27] is used to update the FCM result based on the assumption that changed pixels seldom appear alone spatially.

A class of methods relying on a common feature space is post classification comparison. In this approach a segmentation[6] is produced for each of the representations in domain $\mathcal{X}$ and $\mathcal{Y}$. Then a pixel-by-pixel or region-by-region comparison is performed to detect changes between the representations. The classification step can be interpreted as a categorical feature extraction, which map the data to a low dimensional categorical feature space. The representations are directly comparable in this categorical space. The accuracy of the post-classification comparison is totally dependent on the accuracy of the initial classifications, and the final accuracy closely resembles that resulting from multiplying the accuracies of the individual classification. This may be considered intrinsically low [4].

Different classification schemes have been used to produce the categorical feature space. Camps-Valls *et al.* [42] proposed a family of kernel classifiers. Further is Fuzzy C-Means (FCM) [40], and variations of it such as Fuzzy Local Information C-Means (FLICM) [43] and Evidential C-Means (ECM) [44], commonly used.

Liu *et al.* [16] propose a general multidimensional evidential reasoning (MDER) approach to perform post classification comparison. The framework is agnostic to the classification step, and is concerned with the change extraction in the categorical feature space. The approach builds on their previous work on evidential reasoning in [25, 26], and can be considered as an extension [16] of the classical evidential reasoning frameworks Dempster–Shafer Theory [27] and Dezert–Smarandache Theory [41].

Prendes *et al.* [17] propose a manifold learning approach using a sliding window similarity measure. Based on a statistical model for homogeneous areas in optical and SAR images, the marginal distributions of such areas are defined. These marginals are used to derive the joint distribution for multiple sensors in a homogeneous area, which again is used to define the distribution for multiple sensors in a sliding window. A mixture of C distributions, where C denotes the number of pixel classes, is constructed, and Expectation Maximization (EM) is used to optimize the parameters. This mixture defines a no change manifold $\mathcal{M}$ when the parameter estimation is performed with only unchanged areas. A test statistic on the distance between an image patch and the manifold is used in the final change detection step.

Prendes *et al.* [46] later extended this approach by introducing a Bayesian nonparametric framework to deal with an unspecified number of objects and a Markov Random Field to account for spatial correlation between

[6] Classify each pixel or pixel region

neighboring pixels. The statistically inclined reader is encouraged to further study these thorough works.

Other approaches to supervised heterogeneous change detection include the common meta-Gaussian distributed feature space approach of Storvik *et al.* [24], the graph matching approach of Tuia *et al.* [47], and the approach of Volpi *et al.* [48] using a kernel extension of canonical correlation analysis (CCA).

Most of the supervised methods for heterogeneous change detection are not *supervised* in the machine learning use of the term. Although a priori information about what pixels have changed is used, the changed/unchanged labels are not used as training targets. Rather they guide what areas of the images are used to train the maps between the domains, which can be considered a higher order supervision or *semi-supervision*. Most of the methods can be considered unsupervised or *self-supervised* if some scheme for unsupervised sample selection, e.g. the approach of Gong *et al.* [18] or Luppino *et al.* [2], is employed. The distinction between supervised and unsupervised training is thus not as significant for heterogeneous change detection as for other problem domains.

2.4.2. Unsupervised Methods

The main challenge in unsupervised heterogeneous change detection is to identify unchanged areas which can be used to define maps or projections between domains [2]. For many of the methods covered in this section, the sample selection is done using an Initial Change Map (ICM). Different change detection algorithms have their own merits and no single approach is optimal and applicable to all cases [22].

2.4.2.1. Signal Processing Methods

Gong *et al.* [18] propose an iterative coupled dictionary learning (ICDL) approach for unsupervised heterogeneous change detection. The model aims to establish a pair of coupled dictionaries $S_{\mathcal{X}} : \mathcal{X} \to \mathcal{Z}$ and $S_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{Z}$, where $\mathcal{Z}$ is a common feature space. The dictionaries produce sparse codes for co-located image patches. As only unchanged patches are used to training the dictionary atoms, the atoms of the two dictionaries can be aligned by the one-to-one correspondence at each location. The training of the dictionaries is done under an iterative scheme for unsupervised sample selection to keep the purity of the training sample for the dictionaries. Starting from a random initialization, the scheme selects unchanged pairs of image patches based on the reconstruction error with respect to the current dictionaries.

Touati and Mignotte [49] define a similarity feature map with a set of linear equality constraints on each pixel pair. The feature map represents the

difference between the multitemporal images. A nonlocal pairwise energy-model is used to estimate the over-constrained problem defined by the equality constraints. The estimation is based on a MultiDimensional Scaling (MDS) [50] mapping technique, which achieves linear complexity. The ensemble of five thresholding algorithms proposed by Melgani and Bazi [36] is used to produce the final change map from the difference image produced by the energy-model.

Touati *et al.* [51] propose a de-texturing approach using feature vectors constructed from local histograms. The de-texturing map is constructed based on the insight that two non-adjacent pixels with the same local texture should have the same intensity in the feature space. Each pixel is characterized by a feature vector gathering the values of a coarse gray level histogram and the values of a gradient magnitude histogram in four directions (vertical, horizontal, two diagonals). The histograms are computed for a neighborhood of each pixel. For each image, the extracted features are reduced to a one-channel, gray-level image using MDS. A double histogram matching [52] is done on these images to adjust for possible a scale factor between them. A difference image is computed from these representations, and an ensemble of three thresholding methods [32, 33, 35] is used to produce the final change map.

2.4.2.2. Machine Learning Methods

The advances in machine learning the last decade [53–60] have shifted the focus of change detection methodology from statistical or signal processing models towards (convolutional) neural network models [2]. The machine/deep learning approaches to change detection is mainly concerned with domain adaptation, feature learning and image–to–image translation [1]. The most used models are AutoEncoders (AEs), Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs). These deep learning models are described in more detail in Section 4.

Autoencoders are mainly used for feature extraction. The notation $U_{\mathcal{X}} : \mathcal{X} \to \mathcal{X}'$, where $\mathcal{X}'$ is the feature space, is used to refer to the encoder part of an autoencoder throughout this section. It is optimized in conjunction with a decoder, but only the encoder is used for the feature extraction.

Zhang *et al.* [22] propose a mapping neural network (MNN) $T_{\mathcal{X}'} : \mathcal{X}' \to \mathcal{Y}'$ to make heterogeneous images commensurable. One stacked denoising autoencoder is trained for each domain. These autoencoders, $U_{\mathcal{X}} : \mathcal{X} \to \mathcal{X}'$ and $U_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{Y}'$ are used for feature extraction. In the feature space, an initial change map (ICM) is produced using the post classification comparison scheme of Chen *et al.* [61]. A subset of the pixels in $\boldsymbol{X}'$ and $\boldsymbol{Y}'$ determined by the initial change map is used to train $T_{\mathcal{X}'}$, and a feature similarity analysis is done on $\boldsymbol{Y}'$ and $\hat{\boldsymbol{Y}}'$ using a cosine similarity metric to

produce a change map. This change map is segmented using Fuzzy Local Information C-Means (FLICM) [43] to produce the final change map.

Zhan *et al.* [62] propose an iterative feature mapping network (IFMN) to make the images comparable. The feature mapping is trained to simultaneously minimize the the difference between unchanged pixels and maximize the difference between changed ones. The scheme is threefold. First a feature extraction using one stacked denoising autoencoder for each domain, $U_{\mathcal{X}} : \mathcal{X} \to \mathcal{X}'$ and $U_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{Y}'$, is performed. Then the IFMN $T_{\mathcal{X}'} : \mathcal{X}' \to \mathcal{Y}'$ is trained. The training scheme of the IFMN relies on two initial change maps $P_u(i,j)$ and $P_c(i,j)$, which are binary functions indicating a belief whether the pixel i,j is unchanged or changed respectively. The ICMs are randomly initialized and later set based on thresholds α, β on the difference between $\boldsymbol{Y}'$ and $\hat{\boldsymbol{Y}}'$. An iterative optimization of $T_{\mathcal{X}}$ and (P_u, P_c) is used to train the FMN in an expectation-maximization fashion. The third part of the scheme is a hierarchical clustering component based on FLICM to detect changes. An channel-wise angular distance metric [63] is used on the pixels of the difference image to detect changes, and the hierarchical structure allows detection of both major and minor changes.

Su *et al.* [64] propose a method where a ternary change map [65] which categorizes pixels as one of $H = \{\text{unchanged, positive changed, negative changed}\}$. This scheme also relies on feature extraction using one stacked denoising autoencoder for each domain, $U_{\mathcal{X}} : \mathcal{X} \to \mathcal{X}'$ and $U_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{Y}'$. An initial change map is produced by clustering $\boldsymbol{Y}' - \boldsymbol{X}'$ with Fuzzy C-Means (FCM), assigning an initial $\eta \in H$ to each pixel. Then three maps $F^{(\eta)}_{\mathcal{X}'} : \mathcal{X}' \to \mathcal{Y}'$, one for each class in H, is trained to transform the pixels of a optical image to the ones of a SAR image based on the pixel's initial class. With these three maps, a difference image with three channels $|\boldsymbol{Y}' - \hat{\boldsymbol{Y}}'_\eta|$, where $\hat{\boldsymbol{Y}}'_\eta = F^{(\eta)}_{\mathcal{X}'}(\boldsymbol{X}')$, is produced. The pixels of this difference image are clustered using FCM to produce the final change map.

Zhan *et al.* [67] propose a method to detect changes between an optical and a SAR image. The method is based on a joint feature extraction using an autoencoder. The optical image and the log-transformed SAR image is concatenated in the channel dimension, and a stacked denoising autoencoder $V_{(\mathcal{X},\mathcal{Y})} : (\mathcal{X}, \mathcal{Y}) \to (\mathcal{X}, \mathcal{Y})'$ is trained. In the feature space, the channels are split to again represent the images in $\mathcal{X}'$ and $\mathcal{Y}'$. An initial change map is produced in the feature space using a post classification comparison scheme. The noise robust density-based clustering scheme of Ester *et al.* [68] is used to separately classify the pixels of $\boldsymbol{X}'$ and $\boldsymbol{Y}'$, and a difference image $\boldsymbol{D} = |1 - L_{\mathcal{Y}'}/L_{\mathcal{X}'}|$ is computed, where $L_{\mathcal{X}'}$ and $L_{\mathcal{Y}'}$ denote the classification maps of the respective domains. This difference image is used to produce an initial change map with FCM. The initial change map have the classes $H = \{\text{unchanged, changed, uncertain}\}$. The set of unchanged and changed pixel pairs are used to train a classifier, that is used to give a final label to the uncertain pixels.

Liu *et al.* [9] propose a method using CNNs dubbed Symmetric Convolutional Coupling Networks (SCCN). Two maps $S_{\mathcal{X}} : \mathcal{X} \to \mathcal{Z}$ and $S_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{Z}$ are trained to bring the images to a common feature space $\mathcal{Z}$. These maps each contain one $k \times k$ convolutional layer and an arbitrary number of 1×1 convolutional layers. The kernel size k and the number of convolution filters n_f of the first layer should be the same for each map to achieve symmetry. The 1×1 layers are dubbed coupling layers, and these components gives the method its name. The number of coupling layers $l_{\mathcal{X}}, l_{\mathcal{Y}}$ is allowed to vary to adapt to the different properties of the input domains. The two maps are pretrained as encoders in an autoencoder setting, and an ICM is randomly initialized. Later the ICM is updated based on the difference between the current outputs of the maps, as the maps are trained and the ICM is updated in an alternating fashion. A variation of this approach was proposed by Zhao *et al.* [69], slightly modifying the objective function and the ICM updating procedure.

Gong *et al.* [8] propose a model dubbed CouPling Translation Network (CPTN). The model combines concepts from Variational AutoEncoders (VAEs) [70, 71] and Generative Adverserial Networks (GANs) [54]. It involves two autoencoders with shared weights and a common code space, which are trained with both a cyclic reconstruction term, a code space aligning VAE loss term, and an adversarial loss term aiming to distinguish in which space the decoded images originated. An ICM computed on patch level using the Generalized Kittler-Illingworth Threshold (GKIT) [30] is used for sample selection.

Niu *et al.* [7] propose a conditional Generative Adversarial Network (cGAN) based translation network that aims to translate an optical image with a SAR image as a target, and an approximation network that approximates the SAR image to the translated one by reducing their pixelwise difference. Similarly to Gong *et al.* [8], GKIT is used to produce the ICM, and FMC is used to cluster the difference image to produce the final change map.

Li *et al.* [6] propose a self-supervised CNN approach to heterogeneous change detection. FCM is employed to produce an ICM which is used in an sample selection scheme based on Spatial FCM [45]. The selected samples are stacked in the channel dimension and used to train a classifying CNN.

Luppino *et al.* [2] propose a method to produce an ICM based on affinity matrices. The ICM procedure is covered in detail in Section 3.6. The proposed ICM is used for sample selection to train a selection of regression functions including Gaussian process regression, support vector regression and random forest regression.

Luppino *et al.* [1] propose an improvement of the affinity based ICM. This ICM is used to weight the cross domain loss terms in two image–to–image translation networks.

One of the methods, dubbed X-net [1], uses a pair of CNNs $T_{\mathcal{X}} : \mathcal{X} \to \mathcal{Y}$ and $T_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{X}$ to translate the images between the input domains. The two networks are trained with a cross domain loss term weighted by the ICM and a cyclic loss term. The translation networks are used to map the images between the domains to produce two difference images that are averaged and thresholded with Otsu's method [28] to produce the final change map.

The other method, dubbed Adversarial Cyclic Encoders Network (ACE-Net) [1], uses two pairs of encoder–decoder CNNs ($S_{\mathcal{X}} : \mathcal{X} \to \mathcal{Z}$, $S'_{\mathcal{X}} : \mathcal{Z} \to \mathcal{X}$), ($S_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{Z}$, $S'_{\mathcal{Y}} : \mathcal{Z} \to \mathcal{Y}$)) to map both to a common feature space $\mathcal{Z}$ and between the input domains. The four networks are trained with a direct reconstruction loss term, a cross domain loss term weighted by the ICM, and a cyclic reconstruction loss term. The architecture also uses an adversarial loss term on the code space with a discriminator trying to distinguish what input encoder produced the current code. The change map computation is the same as for X-net.

3. Affinities

An affinity matrix is a generic matrix that describes how close, or similar, two points are in some space [75]. The space can be a feature space or a combined feature and spatial space, and the similarity between two points (i, j) is typically encoded with an affinity value $a_{ij} \in [0, 1]$. The affinity matrix is the matrix with these affinities as elements, $\boldsymbol{A} = [a_{ij}]$, $i, j = 1, \ldots, n$, where n denotes the number of data points. We shall use symmetric affinities, where $a_{ij} = a_{ji}$, hence the affinity matrix is also symmetric. The use of affinity matrices is well-known from spectral clustering [76, 77], graph methods [78], and computer vision [75, 79, 80].

An advantage of affinity matrices is that their structure is independent of the space or domain the described data points lie in. For similar spatial structures in different domains (that is, captured by different sensors or sensing parameters), the affinity matrices should still be similar.

The main novelty proposed in this thesis is to utilize the information held by such affinity matrices for the training of deep neural networks for image–to–image translation. A loss term comparing the affinity matrices of an image and its translated counterpart should enforce that the pixel similarity structure is preserved in the transformation.

This section gives an introduction to proximity measures and affinity matrices from a graph-theoretical point of view.

3.1. Proximity Measures

Proximity measures are used to describe how *similar* and *dissimilar* two points in a space are. This section follows the definitions from Theodoridis and Koutroumbas [12]. $\mathcal{D}$ denotes a dataset $\{\boldsymbol{x}_i\}_{i=1}^n$ throughout.

Definition 3 (Dissimilarity Measure [12]) *A* dissimilarity measure d *on* $\mathcal{D}$ *is a function*

$$d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$$

such that for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{D}$

$$\begin{gathered} \exists d_0 \in \mathbb{R} : d_0 \leq d(\boldsymbol{x}, \boldsymbol{y}), \\ d(\boldsymbol{x}, \boldsymbol{x}) = d_0 \end{gathered}$$

and

$$d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x}).$$

In words, a dissimilarity measure on a dataset $\mathcal{D}$ is a function $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ that is symmetric and has a lower bound $d_0 \in \mathbb{R}$ achieved for equal vectors.

If the measure adheres to the additional constraint that d_0 is only achieved for equal vectors and also the *triangular inequality* in (1), then the measure is a *metric* dissimilarity measure. This gives

Definition 4 (Metric Dissimilarity Measure [12]) *A* metric dissimilarity measure *is a dissimilarity measure where for all* $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{D}$*;*

$$d(\boldsymbol{x}, \boldsymbol{y}) = d_0 \text{ if and only if } \boldsymbol{x} = \boldsymbol{y}$$

and

$$d(\boldsymbol{x}, \boldsymbol{z}) \leq d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z}) \tag{1}$$

Similar definitions exist for similarity measures, which in a sense exhibit inverse properties of dissimilarity measures.

Definition 5 (Similarity Measure [12]) *A* similarity measure s *on* $\mathcal{D}$ *is a function*

$$s : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$$

such that for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{D}$

$$\begin{aligned} \exists s_0 \in \mathbb{R} : s(\boldsymbol{x}, \boldsymbol{y}) &\leq s_0, \\ s(\boldsymbol{x}, \boldsymbol{x}) &= s_0 \end{aligned}$$

and

$$s(\boldsymbol{x}, \boldsymbol{y}) = s(\boldsymbol{y}, \boldsymbol{x}).$$

Note that s_0 is an upper bound. Further is

Definition 6 (Metric Similarity Measure [12]) *a* metric similarity measure *a similarity measure where for all* $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{D}$*;*

$$s(\boldsymbol{x}, \boldsymbol{y}) = s_0 \text{ if and only if } \boldsymbol{x} = \boldsymbol{y}$$

and

$$s(\boldsymbol{x}, \boldsymbol{y}) s(\boldsymbol{y}, \boldsymbol{z}) \leq [s(\boldsymbol{x}, \boldsymbol{y}) + s(\boldsymbol{y}, \boldsymbol{z})]\, s(\boldsymbol{x}, \boldsymbol{z})$$

Dissimilarity measures can be turned into similarity measures, and vice versa, through a monotonic transform [12], i.e., a non-increasing function.

3.1.1. Common Proximity Measures

The most common proximity measure [12] (for real valued vectors) is the weighted L_p norm;

$$L_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} w_i |x_i - y_i|^p \right)^{1/p} \tag{2}$$

where x_i, y_i denote the d elements of $\boldsymbol{x}, \boldsymbol{y}$ respectively and $w_i \geq 0$ is a weight coefficient. When $w_i = 1, i = 1, \ldots, d$, this is often referred to as the unweighted L_p norm or simply L_p norm.

Variants of the weighted L_p norm include [12]

- the *Manhattan norm*; the (weighted) L_1 norm,
- the unweighted Euclidean distance; the unweighted L_2 norm,
- the Mahalanobis distance; the L_2 norm weighted by the variance in each dimension, $w_i = 1/\sigma_i^2$. This can further be generalized to the form
$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})' \, \boldsymbol{S}^{-1} \, (\boldsymbol{x} - \boldsymbol{y})},$$
where $\boldsymbol{S} \in \mathbb{R}_+^{d \times d}$ is the positive semi-definite covariance matrix. A matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ is *positive semi-definite* if $\boldsymbol{z}^\top \boldsymbol{M} \boldsymbol{z} \geq 0$ for all non-zero $\boldsymbol{z} \in \mathbb{R}^n$.
- The (weighted) L_∞ norm
$$L_\infty(\boldsymbol{x}, \boldsymbol{y}) = \max_{i=1,\ldots,d} w_i |x_i - y_i|$$

3.2. Graphs

Affinity matrices have strong ties to proximity graphs. The study of graphs have ties to pure mathematics, set theory and computer science.

Definition 7 *A* simple graph *is an object consisting of two sets called its* vertex set *and its* edge set. *The vertex set is a finite nonempty set. The edge set may be empty, but otherwise its elements are two-element subsets of the vertex set. [82, Definition 5]*

A graph is denoted

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\},$$

where $\mathcal{V}$ and $\mathcal{E}$ denotes the vertex set and the edge set respectively. The terms vertex and node are used interchangeably. Typically, the vertex set is a set of integers

$$\mathcal{V} = \{1, \ldots, n\},$$

where each element can be considered an index for the members of the system or data the graph represents. The edge set is

$$\mathcal{E} = \{\{v_i, v_j\} : v_i, v_j \in \mathcal{V}\},$$

where an edge is represented as a set of vertices. A simple graph, as defined in Definition 7, is an *undirected graph*, where the relationship between v_i and v_j is symmetric, which will suffice for our discussion of affinity matrices

A graph can be built from an arbitrary dataset, e.g., an image, where each pixel is associated with a node, and edges are formed between all pixels.

3.3. Affinity Matrix

The edges of a graph can be associated with an edge weight w to produce a *weighted graph*. Each edge is associated with a weight w_{ij} which describes the importance or strength of the individual connection. Normally, the weights are positive and every edge is associated with a weight; that is $w : \mathcal{E} \to \mathbb{R}_+$.

An *affinity matrix* $\boldsymbol{A} \in \mathbb{R}_+^{n \times n}$ of a weighted graph, with elements

$$\boldsymbol{A} = [a_{ij}] \quad i, j = 1, \ldots, n,$$

encodes the similarity between each pair of node, e.g., it describes local and global similarity structures or neighborhood structures.

Often, the weights are limited to $a_{ij} \in [0, 1]$. A zero entry in the affinity matrix indicates that there is no edge, that is $a_{ij} = 0 \Leftrightarrow \{v_i, v_j\} \notin \mathcal{E}$.

An associated matrix is the *degree matrix*, which is the diagonal matrix with the row sum of the affinity matrix $\boldsymbol{A}$ for weighted graphs. The degree of a vertex describes its total connection to the other vertices.

3.4. Proximity Graphs

Some types of data can naturally be represented by a graph structure, e.g., social networks and communication networks. For these data, the graph nodes and edges are given directly from the system the graph represents, and can often be weighted by some natural quantity in the data.

For other types of data, some design choice must be made to encode the data in a graph. One example of this is image data, which can be represented as a graph by considering how close the pixels are to each other, either in the feature space or in a composite space of the feature space and the physical distance space. The feature space is typically the color cube, and the graph can be built by assigning each pixel to a node and weight

the edges between them by the pairwise distance of all the pixels in this space. A more complex composite similarity measure, such as the one used by Shi and Malik [83], can also be used. Here

$$w_{ij} = \exp\left(\frac{-\|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2^2}{\sigma_I^2}\right) \cdot \begin{cases} \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{\sigma_x^2}\right), & \text{if } \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 < \epsilon \\ 0, & \text{else,} \end{cases}$$

where $\boldsymbol{f}_i$ denotes the intensity vector of pixel i, $\boldsymbol{x}_i$ denotes the pixel position of pixel i, σ_I and σ_X denotes bandwidth parameters, and ϵ is a threshold for the distance in the pixel coordinate space.

Other choices can be made to build a proximity graph, by creating a *fully-connected graph*, where all pairwise similarities are encoded. For a dataset with n observations of dimension d, $\mathcal{D} \in \mathbb{R}^{n \times d}$, each observation is associated with a node and the complete graph is constructed. It is weighted by some proximity measure which adequately describes the neighborhood structure in the feature space, $w_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

This complete graph is often processed further to make it sparser. Removing edges "across the feature space" makes the analysis of the graph easier. Common processing steps include

- the *ϵ-threshold graph*, where edges

$$w_{ij} = \begin{cases} 1, & \text{if } d(\boldsymbol{x}_i, \boldsymbol{x}_j) < \epsilon \\ 0, & \text{else ,} \end{cases}$$

- the *k-nearest neighbor graph*, where

$$w_{ij} = w_{ji} = \begin{cases} s(\boldsymbol{x}_i, \boldsymbol{x}_j), & \text{if } \boldsymbol{x}_i \in k\text{NN}(\boldsymbol{x}_j) \text{ or } \boldsymbol{x}_j \in k\text{NN}(\boldsymbol{x}_i) \\ 0, & \text{else.} \end{cases}$$

  where $\boldsymbol{x}_i \in k\text{NN}(\boldsymbol{x}_j)$ indicates whether $\boldsymbol{x}_i$ is among the k closest points to $\boldsymbol{x}_j$. Note that the nearest neighbor condition is not symmetric, and this is the undirected version of the directed kNN graph.

- the *mutual k-nearest neighbor graph*, where

$$w_{ij} = w_{ji} = \begin{cases} s(\boldsymbol{x}_i, \boldsymbol{x}_j), & \text{if } \boldsymbol{x}_i \in k\text{NN}(\boldsymbol{x}_j) \text{ and } \boldsymbol{x}_j \in k\text{NN}(\boldsymbol{x}_i) \\ 0, & \text{else.} \end{cases}$$

  This is a stronger condition than on the k-nearest neighbor graph, and is generally sparser.

- *Radial basis function graphs*, most commonly with a Gaussian kernel

$$w_{ij} = \begin{cases} \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{\sigma_x^2}\right), & \text{if } \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 < \epsilon \\ 0, & \text{else.} \end{cases} \tag{3}$$

Note that all the proximity graphs described above are similarity graphs, or affinity graphs.

3.5. Affinity Scheme Used Herein

While any of these affinity measures could in principle have been used, we have chosen to use affinities computed with the very common Gaussian kernel for this first assessment of the value of affinity information in image-to-image translation and heterogenous change detection.

Let h, w, and c denote the height, width and number of channels for an image $\boldsymbol{X}$. This corresponds to a dataset $\mathcal{D} \in \mathbb{R}^{(hw)\times c}$. Let each pixel be associated with a node v_i, $i = 1, \ldots, hw$ in a fully connected graph. The affinity matrix is constructed as in Equation (3).

The bandwidth σ_x of the Gaussian kernel is set using a kNN heuristic. For each pair of datapoints is the distance $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2$ computed, and the average distance to the kth neighbor is used to set the bandwidth. The heuristic value $k = \frac{3}{4}hw$ is set after observing empirically that the structure of the affinity matrix is more stable across image domains for high values of k. This scheme is the same as in [1].

3.6. Affinity Norm Difference Image

Luppino *et al.* [2] proposed an affinity–based method to produce a difference image, for which an improvement was proposed in [1].

The rationale is that the edges of the pixel proximity graphs $\boldsymbol{A_X}$ and $\boldsymbol{A_Y}$ of the images $\boldsymbol{X}$ and $\boldsymbol{Y}$ should have a) similar weights for unchanged pixels, and b) dissimilar weights for changed pixels.

These relationships can be described through the vertex degree of the proximity graph $\boldsymbol{A_D} = |\boldsymbol{A_X} - \boldsymbol{A_Y}|$. That is to say, the row sum of row i in $\boldsymbol{A_D}$ quantifies a belief in whether pixel i is changed. A larger value indicates a stronger belief that the pixel is changed. This is used to produce a difference image $\boldsymbol{D}$, where pixel $i = 1, \ldots, hw$ has the value of the vertex degree of v_i in $\boldsymbol{A_D}$.

The affinity matrices $A_{(\cdot)}$ are computed as described in Section 3.5 for the the input images $\boldsymbol{X}$ and $\boldsymbol{Y}$. Due to memory constraints must the affinity computations be performed on overlapping patches of size $k \times k$, and the values of $\boldsymbol{D}$ is averaged over the patches. For increased efficiency is the computation done with a stride Δ. The computation is done on three scales to account for both highly local, local and global neighborhood information.

Note that the difference image produced is not used as to produce the final change map, but as an initial change map (ICM) to perform unsupervised sample selection.

4. Machine Learning

This section provides an introduction to the machine learning principles central to understand the contributions of this thesis, as well as an overview of the principles used in the more similar of the related methods presented in Section 2.4.

The motivation to use machine learning for heterogeneous change detection can be found in the Universal Approximation Theorem.

Theorem 1 (Universal Approximation Theorem [57, 84–86])
A feed-forward network with a linear output layer and at least one hidden layer with any "squashing" activation function can approximate any Borel measurable[7] *function from one finite-dimensional space to another with any desired nonzero amount of error, provided that the network is given enough hidden units. The derivatives of the feed-forward network can also approximate the derivatives of the function arbitrarily well [84].*

This lends that a feed-forward neural network with sufficient expressive power is able to approximate any function f^*. Although, it does not guarantee that any training algorithm is able to learn the appropiate parameters of the neural network [57].

Considering the domains in Definition 1 (Change Detection) on page 5: If the representations in $\mathcal{X}$ and $\mathcal{Y}$ are captured at the same time t, i.e. captured with no change, the two representations, although different, would represent the same, unchanged truth. Assuming there exists some perfect map between the representation domains, the difference present in this case would only be due to capturing noise. If such a perfect map could be found, it would presumably be able to highlight true changes that have occurred for images not captured at the same time. Under the assumption that such a map between the domains $\mathcal{X}$, $\mathcal{Y}$ exists, Theorem 1 motivates the use of neural networks to find it.

4.1. Training Paradigms

There are several training paradigms in machine learning. The classical case is *supervised learning*, where the samples presented in the learning process are labeled, i.e. (manually) associated with the desired output. A sample consists of a value $\boldsymbol{x}$ and a label y, and a collection of samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ constitutes the training data. N denotes the cardinality of the training data. The model is trained to reproduce the correct output from a given input.

[7] *"The concept of Borel measurability is beyond the scope of this [thesis]; for our purposes it suffices to say that any continuous function on a closed and bounded subset of* $\mathbb{R}^n$ *is Borel measurable and therefore may be approximated by a neural network"* [57].

The other main paradigm is *unsupervised learning*, where the samples are not associated with a label. The training data is just $\{\boldsymbol{x}_i\}_{i=1}^N$, and the goal is for the model to infer some structure in the data. The hope is that the inferred structure is useful for solving the problem at hand.

There are several sub-paradigms, the most prominent being *semi-supervised* learning where some labeled data $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ and some unlabeled data $\{\boldsymbol{x}_i\}_{i=1}^M$ is combined, where in general $N \ll M$. The intuition is that the labeled examples can guide the unsupervised process to converge faster and also to solve the desired problem.

4.2. Neural Networks

Neural networks are the quintessential models in machine learning and deep learning [57]. The goal of a neural network is to approximate some function $f^* : \boldsymbol{x} \to \boldsymbol{y}$ by defining a mapping $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ and learn the parameters $\boldsymbol{\theta}$ that best approximates f^*. A feed-forward neural network[8] is associated with a Directed Acyclic Graph (DAG) [57], where each computation node of the DAG represents a layer in the network. Let L denote the number of layers and $\{f^{(l)}\}_{l=1}^L$ denote the set of computation nodes in the DAG. The neural network f can then be represented by composing together these layers as

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = (f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(2)} \circ f^{(1)})(\boldsymbol{x}), \tag{4}$$

where $\circ$ denotes function composition, e.g. $(f^{(2)} \circ f^{(1)})(\boldsymbol{x}) = f^{(2)}(f^{(1)}(\boldsymbol{x}))$.

Each layer $f^{(l)}$ can be interpreted as a vector-to-vector function. Underlying is a set of vector-to-scalar computation nodes, or neurons, acting in parallel. Each neuron produce one index in the output vector. Each layer is on the general form

$$f^{(l)}(\boldsymbol{x}^{(l-1)}; \boldsymbol{\theta}^{(l)}) = \phi\left(\boldsymbol{W}\boldsymbol{x}^{(l-1)} + \boldsymbol{b}\right), \quad l = 1, \ldots, L \tag{5}$$

where

- $\boldsymbol{x}^{(0)}$ denotes the input vector and $\boldsymbol{x}^{(l-1)}$ denotes the output of layer $l-1$;
- $\boldsymbol{\theta}^{(l)} = \{\boldsymbol{W}, \boldsymbol{b}\}$ denotes the set of *trainable parameters* for the layer;
- $\phi(\cdot)$ denotes an vector valued activation function, see Section 4.2.1;
- $\boldsymbol{W} \in \mathbb{R}^{k_l \times k_{l-1}}$ denotes the weight matrix, where k_l denotes the number of neurons in layer l; and
- $\boldsymbol{b} \in \mathbb{R}^{k_l}$ denotes a bias vector.

The input dimension and output dimension k_0 and k_L are governed by the dimensions of the input and output data respectively. The internal or *hidden* dimensions are set as part of the model specification, and governs the expressive power of the network. The number of layers L is referred to as the *depth*, and largest k_l is referred to as the *width* of the network.

[8] The information flows in only one direction.

4.2.1. Activation Function

The vector valued activation function $\phi(\cdot)$ in Equation (5) is an important part of the problem solving capabilities of a neural network. This non-linear function is what takes the function composition in Equation (4) from a linear transformation into a nonlinear one, and thus gives the neural network the capability of assuming a nonlinear function.

The activation functions are typically non-linearities performed element-wise. Common choices include

- the linear function $\phi(\boldsymbol{x}) = \boldsymbol{x}$,
- the logistic sigmoid function $\phi(\boldsymbol{x};\, \sigma) = (1 + \mathrm{e}^{\sigma \boldsymbol{x}})^{-1}$,
- the hyperbolic tangent function, $\phi(\boldsymbol{x}) = \tanh \boldsymbol{x}$, and
- the generalized Rectified Linear Unit (ReLU)

$$\phi(\boldsymbol{x};\, \alpha) = \begin{cases} \boldsymbol{x}, & x \geq 0 \\ \alpha \boldsymbol{x}, & \text{otherwise.} \end{cases}$$

  Special cases include [55]

  - $\alpha = 0$ which is the ordinary (i.e. non-generalized) ReLU,
  - $\alpha = -1$ which yields the absolute value $\phi(\boldsymbol{x}) = |\boldsymbol{x}|$, and
  - $\alpha \leq 1$ which is equivalent to $\phi(\boldsymbol{x};\, \alpha) = \max(\boldsymbol{x}, \alpha \boldsymbol{x})$.

4.2.2. Parameter Optimization

To learn the set of trainable parameters of the network $\boldsymbol{\theta} = \{\boldsymbol{\theta}^{(l)}\}_{l=1}^{L}$ such that the network reach its goal, means of evaluating how well $f(\boldsymbol{x};\, \boldsymbol{\theta})$ approximate $f^*(\boldsymbol{x})$ is needed. A loss function is used to describe the problem the network is intended to solve. Considering the supervised case, where pairs of $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ are provided, the loss function $\mathcal{E}$ is typically some (metric) dissimilarity measure between the target $\boldsymbol{y}_i$ and the networks output $\hat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i)$,

$$\mathcal{E}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i). \tag{6}$$

The formal objective of the optimization process is to minimize the expected value of the loss function under the distribution of the training data. Let J denote the objective function

$$J(\boldsymbol{\theta}) = \mathbb{E}_X \left[\mathcal{E} \right].$$

The notation $\mathbb{E}_X [\cdot]$ is a convenient shorthand for $\mathbb{E}_{X \sim p_{\mathrm{data}}(x)} [\cdot]$, where p_{data} is the distribution of the training data, and will be used throughout. The optimal network parameters $\boldsymbol{\theta}^*$ are obtained by

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{7}$$

4.2.2.1. Loss Functions

Depending on the problem at hand, different loss functions can be applied. The most common option is probably the weighted Mean Squared Error (wMSE)

$$\mathcal{E}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = w_i \cdot \|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\|_2^2,$$

where $\|\cdot\|_2$ denotes the L_2 norm and $w_i \in \mathbb{R}$ is a weight associated with each training sample. With this loss function, the objective function becomes

$$J(\boldsymbol{\theta}) = \mathbb{E}_X\left[\mathcal{E}(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{\theta}))\right] = \frac{1}{N}\sum_{i=1}^{N} w_i \cdot \|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\|_2^2.$$

Other popular loss functions include the mean absolute error for continuous network output and cross entropy loss for categorical network output.

4.3. Optimization Algorithms

Different optimization algorithms can be used to solve the minimization problem in Equation (7). The intuition on how to minimize $J(\boldsymbol{\theta})$ is to follow its gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ towards the minimum.

A key element in the success of neural networks is the *backpropagation* algorithm [89], which allows for simple and inexpensive computation of the gradients [57]. The backpropagation algorithms applies the chain rule[9], with a specific order of operations that is highly efficient [57]. The efficiency of the gradient computations is an integral part of the computational efficiency of the optimization procedure of a neural network. The details of the backpropagation algorithm is out of the scope of this thesis, and the inclined reader is referred to Chapter 6.5 in Goodfellow *et al.* [57] for a thorough derivation.

4.3.1. Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) [90, 91] and its variations are the most commonly used optimization algorithms for machine learning [57]. In SGD is a proportion ϵ of the gradient $\boldsymbol{g}$ subtracted from the parameters $\boldsymbol{\theta}$ that is to be optimized, over many iterations. The proportion $\epsilon \in \mathbb{R}_{>0}$ is referred to as the *learning rate*.

Depending on the size of the network and the training data in conjunction with the computational resources available, can the gradient $\boldsymbol{g} = \nabla_{\boldsymbol{x}} J(\boldsymbol{\theta})$ be computed for either a) several batches[10], or b) the entire training set. If the entire training set is used, there is no stochastic element in the

[9] Id est the chain rule of calculus.

[10] Subsets of the training set drawn at random.

algorithm, and it is truly just the gradient decent algorithm. Otherwise will the gradient be approximated for each batch.

The stop criterion in Algorithm 1 can be a limit on the number of epochs e or the convergence of some metric tracking the training.

The learning rate ϵ is reliably one of the most difficult hyperparameters to set because it significantly affects the model performance [57], and decay strategies are commonly employed to increase the convergence speed [92].

Algorithm 1: Stochastic Gradient Descent (SGD) minimization update [57]

Require *Learning rate schedule* $\epsilon_1, \epsilon_2, \ldots$*; initial parameter* $\boldsymbol{\theta}$.
e=1;
while *not stop criterion* **do**
  Take a batch of m samples from $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$;
  Compute gradient estimate $\hat{\boldsymbol{g}} = \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} \mathcal{E}(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{\theta}))$;
  Apply update $\boldsymbol{\theta} = \boldsymbol{\theta} - \epsilon_e \hat{\boldsymbol{g}}$;
  e++;
end

Several algorithms improving on SGD have been proposed [93], including methods with a momentum term on the gradient, such as Momentum SGD [94] and Nesterov Accelerated Gradient decent (NAG) [95], and methods with an adaptive learning rate scheme, such as Adagrad [96], Adadelta [97], RMSprop [98], Adam [99], AdaMax [99], and Nadam [100]. Luo *et al.* [101] recently proposed variations of some of these with a dynamic bound om the learning rate to achieve a gradual and smooth transition from the adaptive methods to SGD, and give a theoretical proof of convergence.

4.3.2. Momentum Stochastic Gradient Decent

Stochastic Gradient Decent has trouble navigating ravines [93]. As depicted in Figure 1 (a) can SGD end up alternating between the valley sides and thus move more across than along the valley. A momentum term on the gradient can be used to mend this issue. The intuition of momentum is that under the assumption that the last update went in the right direction, it is reasonable to keep moving in that direction. Thus is the momentum term a decaying memory of the previous gradients. An algorithmic description of MSGD is given in Algorithm 2.

Algorithm 2: Momentum Stochastic Gradient Descent (MSGD) minimization update [57]

Require *Learning rate schedule* $\epsilon_1, \epsilon_2, \ldots$; *momentum parameter* α; *initial parameter* $\boldsymbol{\theta}$; *initial velocity* $\boldsymbol{v}$.

e=1;

while *not stop criterion* **do**

  Take a batch of m samples from $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$;

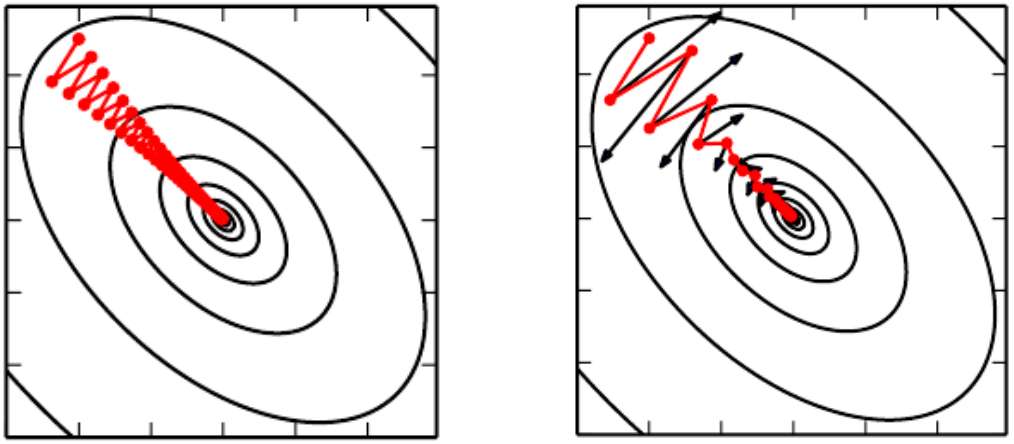
  Compute gradient estimate $\hat{\boldsymbol{g}} = \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} \mathcal{E}(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{\theta}))$;

  Compute velocity update $\boldsymbol{v} = \alpha \boldsymbol{v} - \epsilon \hat{\boldsymbol{g}}$;

  Apply update $\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{v}$;

  e++;

end



(a) *Without momentum: The SGD path is bouncing between the walls of the ravine.*

(b) *With momentum: The SGD path is straighter towards the optimum. The black arrows are the gradient at each step.*

Figure 1: *Traversing a ravine with and without momentum. Figures from [57].*

4.3.3. Adam

Adaptive moment estimation (Adam) [99] extends on the momentum idea, and incorporates in its parameter update a bias-corrected first- and second-order moment. This can be understood as introducing an individual adaptive learning rate for each parameter, based on estimates of the first and second moments of the gradients [99]. See Algorithm 3 for a description of the update scheme.

The learning rate ϵ here acts as an step size, i.e. as an upper bound on the update $\Delta\boldsymbol{\theta}$ [99]. Kingma and Ba [99] suggest default values for the hyperparameters $\epsilon = 1e-3$, $\rho_1 = 0.9$, $\rho_2 = 0.999$, and $\delta = 10^{-8}$.

Algorithm 3: Adaptive moment estimation (Adam) minimization [57]

Require *Step size* ϵ*; numerical stability constant* δ*; initial parameters* $\boldsymbol{\theta}$.
Require *Exponential decay rates for moment estimates* ρ_1 *and* ρ_2 *in* $[0, 1)$.
Initialize 1st and 2nd moment variables $\boldsymbol{s} = 0$, $\boldsymbol{r} = 0$;
t=0;
while *not stop criterion* **do**
  Take a batch of m samples from $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$;
  Compute gradient estimate $\hat{\boldsymbol{g}} = \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} \mathcal{E}(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{\theta}))$;
  t++;
  Update biased 1st moment estimate $\boldsymbol{s} = \rho_1 \boldsymbol{s} + (1 - \rho_1)\hat{\boldsymbol{g}}$;
  Correct bias $\hat{\boldsymbol{s}} = \boldsymbol{s} \cdot (1 - \rho_1^t)^{-1}$;
  Update biased 2nd moment estimate $\boldsymbol{r} = \rho_2 \boldsymbol{r} + (1 - \rho_2)\hat{\boldsymbol{g}} \odot \hat{\boldsymbol{g}}$;
  Correct bias $\hat{\boldsymbol{r}} = \boldsymbol{r} \cdot (1 - \rho_2^t)^{-1}$;
  Compute update element-wise $\Delta\boldsymbol{\theta} = -\epsilon \hat{\boldsymbol{s}} \cdot \left(\sqrt{\hat{\boldsymbol{r}}} + \delta\right)^{-1}$;
  Apply update $\boldsymbol{\theta} = \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end

4.3.4. Generalizability

A central problem in machine learning is to make algorithms that will perform well on new inputs, not just the training data [57]. The optimal architecture can be defined as the one which minimizes the generalization error [102]. Many actions can be taken during the training phase to increase generalizability.

4.3.4.1. Weight regularization

Many regularization techniques are based on limiting the expressive power of the model [57]. This can be done by penalizing the norm of the parameters $\boldsymbol{\theta}$ by adding a norm penalty to $J(\boldsymbol{\theta})$, typically the L_1 or the L_2 norm. The L_1 norm will penalize non-zero weights, encouraging sparse networks. The L_2 norm will favor small weights over large weights, forcing the solution to be closer to the origin of the parameter space, and thus be a 'simpler model'. The assumption is that a simpler model will have better generalization properties.

4.3.4.2. Dropout

Dropout [103] is a strategy to prevent overfitting when training neural networks. By leaving out a portion of randomly selected neurons during training, complex co-adaptations in which a neuron is only useful in the context of several other specific neurons can be avoided [103].

4.3.4.3. Data Augmentation

It is common to artificially increase the size of the training set by augmenting the data. For images is it common to flip, rotate, or add noise [104, 105]. This will add variation to the training data, presumably increasing the generalization properties of the network [106].

4.3.4.4. Batch Normalization

Batch Normalization [107] is a strategy to deal with the fact the distribution of each layer's input changes during training as the parameters of the previous layers change. This effect is called *internal covariate shift*, and can be mended by normalizing the input to each layer as

$$\boldsymbol{x}_i' = \frac{\boldsymbol{x}_i - \mathbb{E}_{\mathcal{B}}\left[\boldsymbol{x}_i\right]}{\sqrt{\mathrm{Var}_{\mathcal{B}}[\boldsymbol{x}_i]}},$$

where $\boldsymbol{x}_i'$ denotes the batch normalized $\boldsymbol{x}_i$, and $\mathbb{E}_{\mathcal{B}}\left[\cdot\right]$ and $\mathrm{Var}_{\mathcal{B}}[\cdot]$ denote the expectation and variance under the distribution of the batch $\mathcal{B}$.

4.4. Convolutional Neural Networks

Convolutional Neural Networks is a class of networks for processing data with a grid-like topology, such as regular interval time series or image data [57]. A CNN is a neural network with some of the layers on the form

$$f^{(l)}(\boldsymbol{x}^{(l-1)}; \boldsymbol{\theta}^{(l)}) = \phi\left(\boldsymbol{K} * \boldsymbol{x}^{(l-1)} + \boldsymbol{b}\right), \tag{8}$$

where $*$ denotes convolution, $\boldsymbol{K}$ is a convolution kernel, $\boldsymbol{b} \in \mathbb{R}^{k_l}$ is a bias vector, k_l denotes the number of filters in the convolution layer, $\boldsymbol{\theta}^{(l)} = \{\boldsymbol{K}, \boldsymbol{b}\}$, and the other symbols are as in Equation (5). The number of dimensions of $\boldsymbol{K}$, and thus also the convolution operator, depends on the dimensions of the input data.

For image data organized as height, width, and channels, that is $\boldsymbol{x}^{(0)} \in \mathbb{R}^{h \times w \times c}$, the kernel will have the shape $\boldsymbol{K}^{(l)} \in \mathbb{R}^{h_l \times w_l \times k_{l-1} \times k_l}$. The convolution operator is broadcasted across the last dimension of $\boldsymbol{K}^{(l)}$, which corresponds to the number of filters in the convolutional layer. That is for $\boldsymbol{x}^{(1)} = \boldsymbol{K}^{(1)} * \boldsymbol{x}^{(0)}$, $\boldsymbol{x}^{(1)} \in \mathbb{R}^{h \times w \times k_l}$. The convolution will in this case be a three-dimensional convolution, and for pixel (i, j) in $\boldsymbol{x}^{(0)}$ will

$$\boldsymbol{x}^{(1)}(i, j, k) = (\boldsymbol{K}^{(1)} * \boldsymbol{x}^{(0)})(i, j) = \sum_{m,n,o} \boldsymbol{x}^{(0)}(i + m, j + n, o) \cdot \boldsymbol{K}^{(1)}(m, n, o, k),$$

where $m = \left\lfloor \frac{-h_1}{2} \right\rfloor, \ldots, \left\lfloor \frac{h_1}{2} \right\rfloor$, $n = \left\lfloor \frac{-w_1}{2} \right\rfloor, \ldots, \left\lfloor \frac{w_1}{2} \right\rfloor$, and $o = 1, \ldots, c$. For brevity are h_1 and w_1 here assumed to be odd numbers, but the operation can be defined for an even kernel size. The broadcasting is done for $k = 1, \ldots, k_1$, i.e. the convolution operator is applied for each kernel k to produce k_l output channels. The kernel size (h_l, w_l) and the number of filters k_l must in the design of the network be specified for each layer, $l = 1, \ldots, L$.

An important construct in many CNN designs is pooling. A pooling function is a downsampling operation which outputs some statistic of the input values [57]. An example is max pooling [108], which takes a neighborhood, say an array of size 2×2, as input and outputs e.g. an 1×1 array with the value corresponding to the maximum of the inputs. This operation is computed in strides across the image, which downsamples and condenses the information. Described in mathematical terms;

$$\text{MaxPool}\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) \coloneqq [\max(a, b, c, d)]\,.$$

4.5. Autoencoders

An AutoEncoder (AE) [109] is an encoder–decoder pair (U, V),

$$U : \mathcal{X} \to \mathcal{Z}$$
$$V : \mathcal{Z} \to \mathcal{X},$$

trained with the target that $(V \circ U)(\boldsymbol{X}) = \boldsymbol{X}$. Training an autoencoder to minimize this reconstruction error is equivalent to maximizing a lower bound on mutual information between the input space $\mathcal{X}$ and the *code space* $\mathcal{Z}$ [109].

Autoencoders have proven capable of solving problems like feature extraction, dimensionality reduction, and clustering [1]. For an AE to solve such problems must the expressive power of the encoder–decoder pair be sufficiently low to not allow an identity mapping. This is commonly done by placing a bottleneck on the code space, i.e. limiting the dimensionality of the code space, or L_1 regularize the weights to enforce a sparse representation.

If the networks used for the encoder and decoder are deep and dropout is used during training, it is an Stacked Denoising Auto Encoder (SDAE) as proposed by Vincent *et al.* [109]. This is a really common case, and the distinction is not always made.

4.6. Adversarial Discriminative Training

Adversarial discriminative training is inspired by game theory, with two networks competing towards conflicting goals. One architecture, the *generator*, is generating the desired output, while the other, the *discriminator*, attempts to identify whether its input is real data or generator output.

This scheme was introduced as Generative Adverserial Networks (GANs) by Goodfellow *et al.* [54]. The goal is to train the generator to act as a multidimentional sampler from a distribution p_{real} given representative samples from said distribution.

This can be formulated as a minimax game between the generator G and the discriminator D:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{\text{real}}(x)} \left[\log(D(x))\right] + \mathbb{E}_{u \sim p_{\text{fake}}(u)} \left[\log(1 - D(G(u)))\right],$$

where $u \sim p_{\text{fake}}(u)$ denotes the output distribution of the generator G. Variations of this algorithm, such as Least Square GAN [110] and Wasserstein GAN [111], replace the negative log-likelyhood term, as it is unstable in training [72].

The advantage of adversarial training is that the loss function for the generator is learned [112]. One only needs to provide samples of the desired output distribution. For generative problems, this is desirable compared to conventional loss functions, which generally produce a blurry output [112].

4.7. Image–to–Image Translation

The idea of image–to–image translation is to translate one possible representation of a scene into another [112]. The concept dates back to the image analogies of Hertzmann *et al.* [113]. Examples of such translations are from satellite images into the corresponding map, from grayscale images to color images, or from photos taken during the day to photos taken at night [112]. Other examples are translation between photographs and paintings, translating photographs between seasons, and object transfiguration such as making zebras appear like horses [114, 115]. The goal is to retain the contents of the image, but transfer the style.

Herein, image–to–image translation is used to map remote sensing images from one image domain to another. Two main training strategies are specifically used in the training of neural networks performing the image–to–image translation.

4.7.1. Cyclic Consistency

Using transitivity to regularize structured data has a long history [115]. The idea is that a pair of injective maps $T_{\mathcal{X}} : \mathcal{X} \to \mathcal{Y}$ and $T_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{X}$ should have the property $\boldsymbol{X} = (T_{\mathcal{Y}} \circ T_{\mathcal{X}})(\boldsymbol{X})$. The maps can thus be trained in a similar fashion to autoencoders, with the goal of reproducing the input. This is a common part of the training strategy for many image–to–image translation systems.

4.7.2. Adversarial Methods

Adversarial methods have been popularized with applications in image–to–image translation, and are achieving state of the art performance.

One of the most popular models is the pix2pix model [112], which employs a U-net [116] like architecture combined with a conditional GAN (cGAN) training regime. In a cGAN the generator is trained to provide an output conditioned on some input, rather than noise, e.g., an input image that should be translated into another image domain.

Zhu *et al.* [114] achieved impressive results on unpaired image–to–image translation problems, using cGANs in conjunction with a cyclic consistency loss term [115].

Concerning change detection in remote sensing images, the works of Luppino *et al.* [1] and Niu *et al.* [7] are using adversarial training to perform image–to–image translation.

4.8. Metrics

Different metrics can be used to evaluate the performance of a neural network. Change detection can be considered a binary classification problem, and a selection of metrics for binary classification problems is presented here. Some of these metrics are used to evaluate the change detection model proposed in this thesis.

For the definition of the metrics, it is useful to consider a confusion matrix. A confusion matrix is found in Table 1, where the rows encode the predicted label while the columns encode the true label. The last row and column contains the sum of the corresponding column and row respectively. All elements contains the count of the elements of that type.

The first row of the table contains the true positives `TP`, the false positives `FP`, and the predicted positives `PP` = `TP` + `FP`. The second row contains the false negatives `FN`, the true negatives `TN` and the predicted negatives `PN` = `FN` + `TN`. The third row contains the real positives `RP` = `TP`+`FN` and the real negatives `RN` = `FP`+`TN`.

	Real positive (+R)	Real negative (–R)	
Predicted positive (+P)	`TP`	`FP`	`PP`
Predicted negative (–P)	`FN`	`TN`	`PN`
	`RP`	`RN`	N

Table 1: *Binary confusion matrix [117]. The color scheme of the cells is consistent with the confusion maps presented in Section 9*

In the following, let $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ denote a vector of true and predicted binary labels respectively.

Let `tpr` = `TP`/`RP` denote the True Positive Rate[11], and `tnr` = `TN`/`RN` the True Negative Rate[12]. Further let `tpa` = `TP`/`PP` denote the True Positive Accuracy[13].

4.8.1. Accuracy

Accuracy (ACC), also referred to as Rand Accuracy, is the percentage of correctly classified elements [117]

$$\mathrm{ACC}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{\texttt{TP} + \texttt{TN}}{N}$$

Accuracy is an intuitive metric, and is widely used for classification, although it is not appropriate when considering unbalanced cases [118].

[11] Also referred to as Recall
[12] Also referred to as Inverse Recall
[13] Also referred to as Precision

4.8.2. F–measure

The F–measure, or F1 score, is the harmonic mean of the True Positive Rate and the True Positive Accuracy:

$$F_1(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{2 \cdot \texttt{tpa} \cdot \texttt{tpr}}{\texttt{tpa} + \texttt{tpr}} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.8.3. Cohen's Kappa

Cohen's Kappa Coefficient (κ) [119] was introduced as a measure of agreement between two judges in the field of psychology. It is a standardized value in $[-1, 1]$, where 0 represents the amount of agreement that can be expected at chance [120]. It can be used in classification to measure the agreement between observed and predicted classes [118], and is defined as

$$\kappa(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{\text{ACC}(\boldsymbol{y}, \hat{\boldsymbol{y}}) - P_e}{1 - P_e},$$

where P_e is the hypothetical probability of chance agreement [118]. For a binary classification problem [1]:

$$P_e = \frac{\texttt{PP} \cdot \texttt{PN}}{N^2} + \frac{\texttt{RP} \cdot \texttt{RN}}{N^2}.$$

4.8.4. Matthews Correlation Coefficient

Matthews Correlation Coefficient (MCC) [121] is a metric similar to Cohen's Kappa, defined as [118]:

$$\text{MMC}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{\texttt{TP} \cdot \texttt{TN} - \texttt{FP} \cdot \texttt{FN}}{\sqrt{(\texttt{TP} + \texttt{FP})(\texttt{FP} + \texttt{TN})(\texttt{TP} + \texttt{FN})(\texttt{FN} + \texttt{TN})}} \tag{9}$$

in the binary case.

Part II – Proposed Methodology

5. Affinity–Guided Image-to-Image Translation

The work presented in this thesis have strong ties to the work of Luppino *et al.* [1]. The paired image–to–image translation approach to change detection seems promising, especially with the explainability of the approach. Efforts to explain the decisions of CNNs have come a long way [122–124], and it is reasonable to assume that errors made in image translation networks can be explained. Also, when the images are mapped between the input domains it is easier[14] to understand why something failed, as these representations are more or less human readable.

A core contribution of this thesis is the *Affinity Guided Image–to–Image Translation* (AGIT) loss term. The observation that inspired this approach is that the affinity matrices of one image in different domains should be consistent under certain assumptions on the affinity computation and the physical properties captured in the domains. This is true because affinities are normalized to the interval $[0, 1]$, hence the affinity matrix of an image is more tied to the depicted object than to the domain-specific image representation.

This realization lead to the idea of minimizing the difference between the affinity matrices from each representation domain to guide the training of the image–to–image translation maps.

Let $T_{\mathcal{X}} : \mathcal{X} \to \mathcal{Y}$ be a map from image domain $\mathcal{X}$ to $\mathcal{Y}$, and $g_{\mathcal{X}} : \mathcal{X} \to \mathcal{A}_{\mathcal{X}}$ and $g_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{A}_{\mathcal{Y}}$ be functions computing the affinity matrices of $\boldsymbol{X} \in \mathcal{X}$ and $\boldsymbol{Y} \in \mathcal{Y}$ respectively. $\mathcal{A}_{(\cdot)}$ denotes the affinity space of $\mathcal{X}$ and $\mathcal{Y}$. Further let $\boldsymbol{A}_{\boldsymbol{X}} = g_{\mathcal{X}}(\boldsymbol{X})$ and $\boldsymbol{A}_{\hat{\boldsymbol{Y}}} = (g_{\mathcal{Y}} \circ T_{\mathcal{X}})(\boldsymbol{X})$ be the affinity matrices of $\boldsymbol{X}$ and $\hat{\boldsymbol{Y}}$ respectively. In mathematical terms can the affinity consistency through the image translation be described as $\boldsymbol{A}_{\boldsymbol{X}} \approx \boldsymbol{A}_{\hat{\boldsymbol{Y}}}$.

The unsupervised affinity guiding loss term is defined as

$$\mathcal{E}_{\mathtt{A}}(\boldsymbol{X}) := \frac{1}{(hw)^2} \sum_{i=1}^{hw} \sum_{j=1}^{hw} |a_{ij}^{(\boldsymbol{X})} - a_{ij}^{(\hat{\boldsymbol{Y}})}|^2, \tag{10}$$

where $a_{ij}^{(\cdot)}$ are the elements of $\boldsymbol{A}_{\boldsymbol{X}}$ and $\boldsymbol{A}_{\hat{\boldsymbol{Y}}}$ respectively, whereas h and w denote the height and width of $\boldsymbol{X}$. A system diagram for the loss term is depicted in Figure 2. The loss term can be employed similarly for a map $T_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{X}$.

The structure of the affinity matrices $\boldsymbol{A}_{\boldsymbol{X}}$ and $\boldsymbol{A}_{\hat{\boldsymbol{Y}}}$ should be the same, as they are computed from an observation of the same truth. Nevertheless, the

[14] Compared to code space approaches.

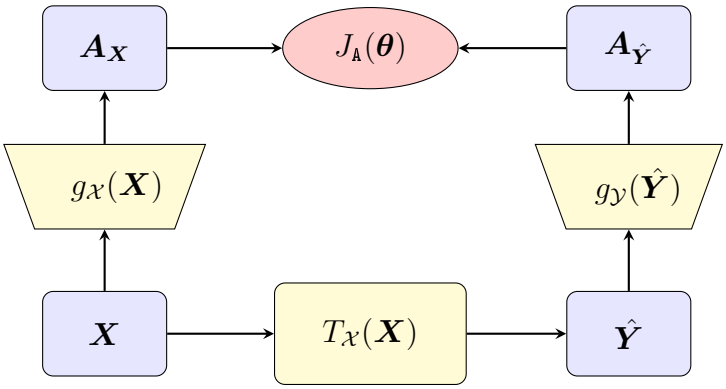


Figure 2: *System diagram for the $J_{\mathtt{A}}$ loss term. This model is referred to as* `A--`*. See Equation* (14) *for the definition of $J_{\mathtt{A}}$.*

profile of the monotonous distance-to-affinity mapping with e.g. a Gaussian kernel will depend on the kernel bandwidth. In order to align the affinities of co-located pixels, it is desired to match these mappings in the respective domains. Exact matching can be difficult to accomplish, since the mapping profiles will also depend on the domain-specific noise model and statistical characteristics.

The kernel bandwidth can be seen as a reference value for a characteristic distance between data points that are relatively similar. This parameter scales or normalizes the distance and determines the profile of the monotonous mapping from distance to affinity. It should be used to enforce affinity matrices that are comparable across the domains, i.e. the bandwidths must be adjusted such that the affinity matrices are reasonably aligned. Herein, it is assumed that using the same heuristic to choose the bandwidth in the two domains is sufficient to achieve directly comparable affinity matrices $\boldsymbol{A}_{\boldsymbol{X}}$ and $\boldsymbol{A}_{\hat{\boldsymbol{Y}}}$.

In a change detection system the main advantage of $\mathcal{E}_{\mathtt{A}}$ is that it incorporates a cross domain distance, but is not limited to learn from the unchanged pixels. Preferably should the map learn to translate each pixel to the corresponding pixel class representation in the other domain. If the map learns to translate all pixels belonging to class c_i in $\boldsymbol{X} \in \mathcal{X}$ to the corresponding class c_i in $\hat{\boldsymbol{Y}} \in \mathcal{Y}$, changes between $\boldsymbol{X}$ and $\boldsymbol{Y}$ can be found by comparing $\hat{\boldsymbol{Y}}$ and $\boldsymbol{Y}$. The affinity guidance should enforce this class consistency, as the affinity matrices would change if pixel memberships are changed in the image transformation.

Another advantage of $\mathcal{E}_{\mathtt{A}}$ is that it is unsupervised, and thus does it not require paired images as training data. The AGIT loss term might be useful for general image–to–image translation systems, but herein is it only tested in a change detection system.

6. Affinity-Guided X-net

The application that inspired the affinity-guiding loss term is change detection. Training neural networks for this application can be hard, as there are changes in the paired images used for training, i.e. the training targets are noisy. The affinity loss term is assumed beneficial in this setting, as the affinity matrices of an image $\boldsymbol{X}$ and its translated version $\hat{\boldsymbol{Y}}$ should be similar, irrespective of the changes that are present between $\boldsymbol{X}$ and $\boldsymbol{Y}$. The change detection model proposed herein is an extension of the X-net change detection model proposed by Luppino *et al.* [1].

6.1. X-net

X-net [1] consists of two tandem image–to–image translation networks $T_{\mathcal{X}} : \mathcal{X} \to \mathcal{Y}$ and $T_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{X}$. The core idea is that the image translations produce a representation in $\mathcal{Y}$ of what is observed in $\mathcal{X}$, and vice versa, such that changes can be extracted after the image translation. The details of the image translation CNNs are presented in Section 6.3.1.

6.1.1. Evaluation Flow

The image translation networks are used to translate the images $\boldsymbol{X}$, $\boldsymbol{Y}$ to their respective other domain $\hat{\boldsymbol{Y}} = T_{\mathcal{X}}(\boldsymbol{X}), \hat{\boldsymbol{X}} = T_{\mathcal{Y}}(\boldsymbol{Y})$. Thus can the image pairs $(\boldsymbol{X}, \hat{\boldsymbol{X}})$ and $(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$ be compared directly. As depicted in Figure 3, two difference images are computed, one for each domain, and these are averaged to produce the final difference image. This image is filtered and thresholded to produce the final change map. The details of these operations are found in Section 6.3.3.

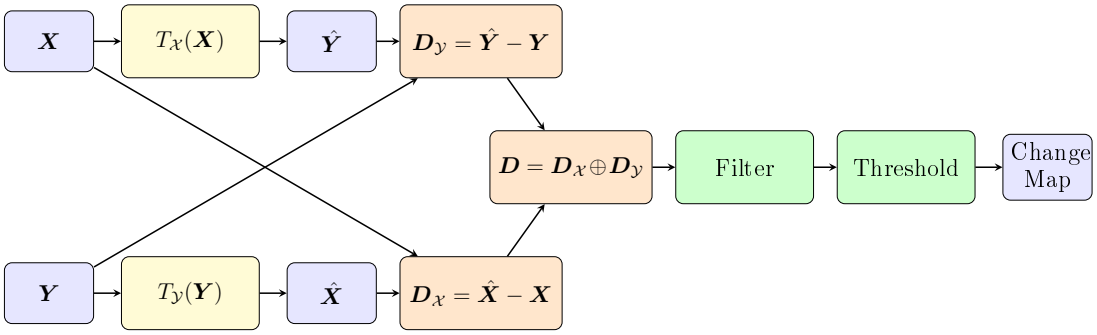


Figure 3: *Evaluation flowchart for X-net [1]. The details of these abstract operations are found in Section 6.3.3.*

6.1.2. Objective Function

A twofold objective function, depicted in Figure 4, with a weighted cross-domain term and a cyclic term is used to optimize the image translation CNNs.

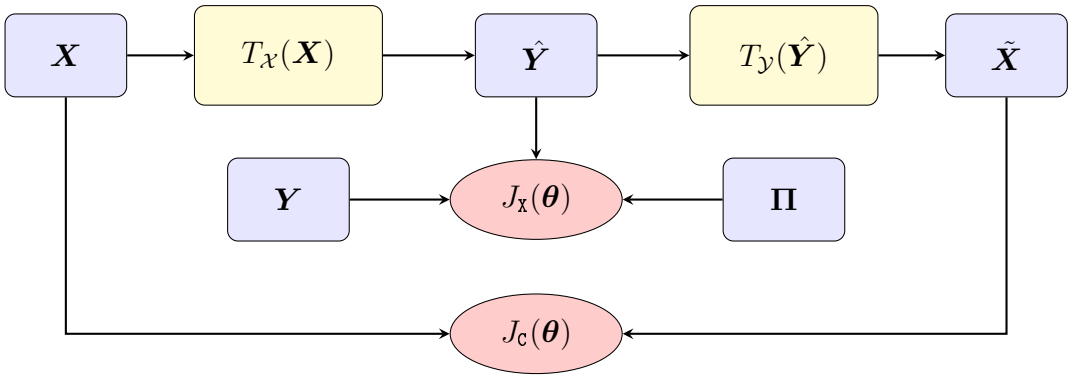


Figure 4: *One half of X-nets training scheme [1]. The definitions of $J_{\mathtt{X}}$ and $J_{\mathtt{C}}$ are found in Equations* (11) *and* (12)*. This model is also referred to as* `-CX` *herein. A similar flow can be created starting from $\boldsymbol{Y}$ and going to $\tilde{\boldsymbol{Y}}$ by swapping $(\boldsymbol{X}, \boldsymbol{Y})$ and $(\mathcal{X}, \mathcal{Y})$ in the chart.*

6.1.2.1. Weighted Cross-Loss Objective

Let $\mathcal{E}_{\mathtt{X}}$ denote the weighted cross-loss term, which compares $(\boldsymbol{X}, \hat{\boldsymbol{X}})$ or $(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$. Weighted mean square error is used for the comparison, and for $T_{\mathcal{X}}$, is the objective function

$$J_{\mathtt{X}}(\boldsymbol{\theta}) = \mathbb{E}_X\left[\mathcal{E}_{\mathtt{X}}(\boldsymbol{X}, \boldsymbol{Y})\right] = \frac{1}{hw}\sum_{i=1}^{h}\sum_{j=1}^{w}\pi_{ij}\|\boldsymbol{y}_{ij} - \hat{\boldsymbol{y}}_{ij}\|_2^2, \tag{11}$$

where $\pi_{ij} \in \boldsymbol{\Pi}$ is a weight indicating a prior belief on whether pixel i, j is changed, and h, w denotes the dimensions of the images $\boldsymbol{X}$ and $\boldsymbol{Y}$.

The *cross-loss weights* $\boldsymbol{\Pi} = 1 - \boldsymbol{C}_{\text{icm}}$ are computed from the initial change map, and the same $\boldsymbol{\Pi}$ is used for both cross-terms in Equation (13). The affinity norm initial change map scheme [1], described in Section 3.6, is used to produce $\boldsymbol{C}_{\text{icm}}$.

6.1.2.2. Cyclic Loss Objective

Let $\mathcal{E}_{\mathtt{C}}$ denote the cyclic loss term, which compares $\tilde{\boldsymbol{X}} = (T_{\mathcal{Y}} \circ T_{\mathcal{X}})(\boldsymbol{X})$ with $\boldsymbol{X}$ and $\tilde{\boldsymbol{Y}} = (T_{\mathcal{X}} \circ T_{\mathcal{Y}})(\boldsymbol{Y})$ with $\boldsymbol{Y}$. Mean squared error is used for the comparison, and for the image in $\mathcal{Y}$ the objective function is

$$J_{\mathtt{C}}(\boldsymbol{\theta}) = \mathbb{E}_Y\left[\mathcal{E}_{\mathtt{C}}(\boldsymbol{Y})\right] = \frac{1}{hw}\sum_{i=1}^{h}\sum_{j=1}^{w}\|\boldsymbol{y}_{ij} - \tilde{\boldsymbol{y}}_{ij}\|_2^2,, \tag{12}$$

which contributes to the optimization of both $T_{\mathcal{X}}$ and $T_{\mathcal{Y}}$.

6.1.2.3. Complete Objective Function

The cross-domain term is applied twice to influence the parameters of the individual networks, and the cyclic term is applied twice, but influences

the parameters of both networks. The combination of these loss terms with the addition of L_2 weight regularization yields the entire objective function

$$\begin{aligned} J(\boldsymbol{\theta}) =& \lambda_{\mathtt{X}} \cdot \mathbb{E}_X \left[\mathcal{E}_{\mathtt{X}}(\boldsymbol{X}, \boldsymbol{Y})\right] + \lambda_{\mathtt{C}} \cdot \mathbb{E}_X \left[\mathcal{E}_{\mathtt{C}}(\boldsymbol{X})\right] \\ &+ \lambda_{\mathtt{X}} \cdot \mathbb{E}_Y \left[\mathcal{E}_{\mathtt{X}}(\boldsymbol{Y}, \boldsymbol{X})\right] + \lambda_{\mathtt{C}} \cdot \mathbb{E}_Y \left[\mathcal{E}_{\mathtt{C}}(\boldsymbol{Y})\right] \\ &+ \lambda_{\mathtt{R}} \cdot \|\boldsymbol{\theta}\|_2^2, \end{aligned} \tag{13}$$

where $\lambda_{(\cdot)} \in \mathbb{R}_{>0}$ are hyperparameters to balance the contribution of each term.

The data flow of the loss terms $\mathbb{E}_X \left[\mathcal{E}_{\mathtt{X}}(\boldsymbol{X}, \boldsymbol{Y})\right]$ and $\mathbb{E}_X \left[\mathcal{E}_{\mathtt{C}}(\boldsymbol{X})\right]$ is visualized in Figure 4. A similar flow can be created for the other pair of loss terms, $\mathbb{E}_Y \left[\mathcal{E}_{\mathtt{X}}(\boldsymbol{Y}, \boldsymbol{X})\right]$ and $\mathbb{E}_Y \left[\mathcal{E}_{\mathtt{C}}(\boldsymbol{Y})\right]$ by swapping the symbols $(\boldsymbol{X}, \boldsymbol{Y})$ and $(\mathcal{X}, \mathcal{Y})$ in the chart. This holds true for all similar loss flowcharts throughout this thesis.

6.2. Affinity-Guided X-net

In the X-net affinity matrices are used to provide an initial change map, which serves as a change prior on pixel level. Changes typically occur in a region, which affects the semi-local affinity structures. Thus, comparison of the affinity matrices of the input images can produce a decent change map, which can also be used as an initial change map (ICM) [1].

During optimization of the networks, the pixels that are perceived as changed in the ICM will contribute less to the cross-loss term. They will still contribute in the cyclic loss term, which gives an indication of how a changed pixel should be mapped into the other domain through the composite mappings $\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{Y}}$. The provision of more information on how a pixel should be mapped between the domains, irrespective of changes between $\boldsymbol{X}$ and $\boldsymbol{Y}$, is presumably useful in the training process.

The goal is to learn an image translation as if no change had occurred. Under such a map should the affinity structures be similar for an image $\boldsymbol{X}$ and its transformed counterpart $\hat{\boldsymbol{Y}}$. The suggested loss term $\mathcal{E}_{\mathtt{A}}$ does enforce this class consistency in the image translation.

This holds true even for pixels that are changed, as the affinity matrices are computed for the same image in different domains. Thus, changed pixels are mapped to the appropriate class in the cross-image, even though the pixels do not contribute to the cross loss-term due to the change prior. The affinity regularization could also amend errors made due to the fallibility of the change prior.

The addition of a cyclic affinity term comparing $\boldsymbol{A}_{\boldsymbol{X}}$ with $\boldsymbol{A}_{\tilde{\boldsymbol{X}}}$ would not add more information, as the image $\tilde{\boldsymbol{X}}$ is directly comparable with $\boldsymbol{X}$. The information encoded in the difference between the affinity of the input

image and the cycled image would not help during training, as the information is intrinsically the same as in the ordinary cyclic term. A cyclic affinity term would likely only add computational complexity, and is not considered further.

These X-net models are directed acyclic graphs (DAGs) with multiple inputs and outputs. Further are the training and evaluation phases quite different. This increases the perceived complexity of the model, and introduces some pitfalls in understanding and implementing it.

6.2.1. Objective Function

The objective function of the Affinity-guided X-net (AX-net) is the same as for X-net, with the addition of two affinity-guided loss terms.

6.2.1.1. Affinity Loss Objective

Assume that functions $g_{\mathcal{X}}$ and $g_{\mathcal{Y}}$ for computing the affinity matrix in the domains $\mathcal{X}$ and $\mathcal{Y}$ respectively are available. Let $\mathcal{E}_{\mathtt{A}}$, as defined in Equation (10), denote the affinity loss term, which compares $\boldsymbol{A}_{\boldsymbol{X}}$ with $\boldsymbol{A}_{\hat{\boldsymbol{Y}}}$ and $\boldsymbol{A}_{\boldsymbol{Y}}$ with $\boldsymbol{A}_{\hat{\boldsymbol{X}}}$.

For a dataset with a pair of $h \times w$ images of c_1 and c_2 channels each, the affinity-guiding objective function is

$$J_{\mathtt{A}}(\boldsymbol{\theta}) = \mathbb{E}_X\left[\mathcal{E}_{\mathtt{A}}\right] = \frac{1}{(hw)^2}\sum_{i=1}^{hw}\sum_{j=1}^{hw}|a_{ij}^{(\boldsymbol{X})} - a_{ij}^{(\hat{\boldsymbol{Y}})}|^2, \tag{14}$$

where the image channels c_1 and c_2 are absorbed in the affinity calculation, since the scalar distances and affinities are computed from multivariate data.

6.2.1.2. Total Loss

Combining the loss terms from Equation (13) with the two affinity terms yields the objective function

$$\begin{aligned} J(\boldsymbol{\theta}) =& \lambda_{\mathtt{A}} \cdot \mathbb{E}_X\left[\mathcal{E}_{\mathtt{A}}(\boldsymbol{X})\right] + \lambda_{\mathtt{A}} \cdot \mathbb{E}_Y\left[\mathcal{E}_{\mathtt{A}}(\boldsymbol{Y})\right] && \text{(15a)}\\ &+\lambda_{\mathtt{C}} \cdot \mathbb{E}_X\left[\mathcal{E}_{\mathtt{C}}(\boldsymbol{X})\right] + \lambda_{\mathtt{C}} \cdot \mathbb{E}_Y\left[\mathcal{E}_{\mathtt{C}}(\boldsymbol{Y})\right] && \text{(15b)}\\ &+\lambda_{\mathtt{X}} \cdot \mathbb{E}_X\left[\mathcal{E}_{\mathtt{X}}(\boldsymbol{X},\boldsymbol{Y})\right] + \lambda_{\mathtt{X}} \cdot \mathbb{E}_Y\left[\mathcal{E}_{\mathtt{X}}(\boldsymbol{Y},\boldsymbol{X})\right] && \text{(15c)}\\ &+\lambda_{\mathtt{R}} \cdot \|\boldsymbol{\theta}\|_2^2. && \text{(15d)} \end{aligned}$$

Half a system diagram for this objective function is shown in Figure 5. The other half is the symmetric counterpart with the image domains exchanged.

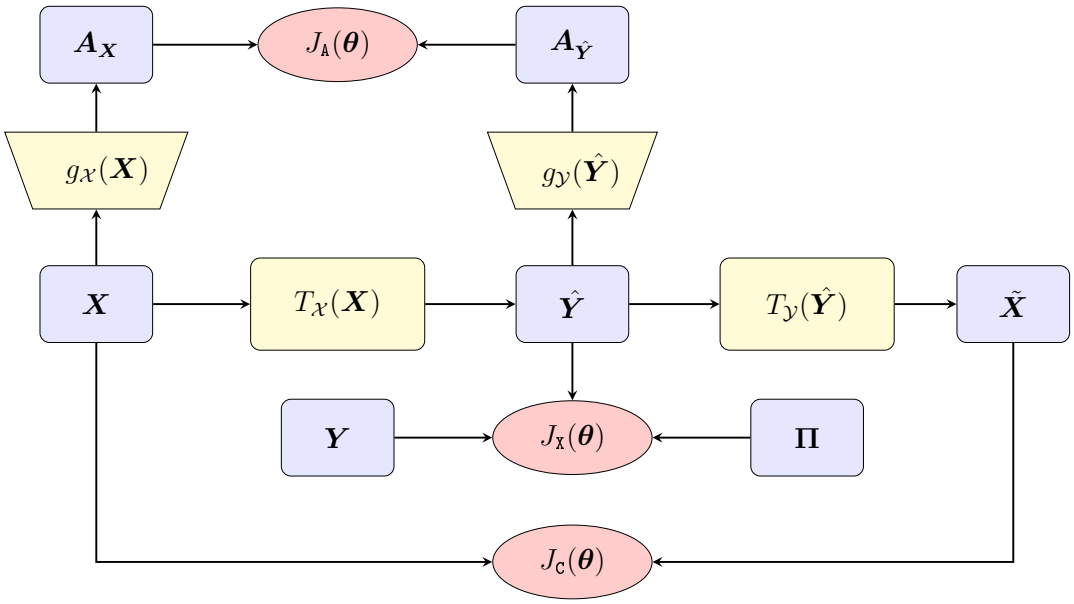


Figure 5: *System diagram for loss terms of the affinity-guided X-net. This model is referred to as* **ACX**. *See Equations* (11), (12) *and* (14) *for the definitions of* $J_{\mathtt{X}}$, $J_{\mathtt{C}}$ *and* $J_{\mathtt{A}}$.

6.3. Implementation Details

The models was implemented in Python 3.6.8 using TensorFlow 2.0.0 [125]. The full code is available at github.com/MadsAdrian/MastersThesis. The code was ported from the TensorFlow 1.4 implementation of Luppino *et al.* [1] by me, and further developed and generalized in collaboration with Luigi T. Luppino. The code that implement the models and run the experiments is between 1000 and 2000 lines of code, depending on what is included in the count. The code is object-oriented, and is currently used in several other projects in the research group.

Many of the choices made in designing and training the image translation networks build on standard choices. Most places, the default setting of the used library is applied, because we had no motivation to deviate from the standard.

The main contribution of this thesis is the Affinity-Guided X-net, with the objective function defined in Equation (15). Fine tuning and elaborate choices have been kept at a minimum, to focus on evaluating the overall performance of the model. It is likely that minor or major fine tuning could improve the model performance on some metric.

The particular implementation of e.g. the image translation networks presented here must be considered separate from the loss term of the Affinity-Guided X-net. Many network architectures and optimizing schemes can be chosen, and it is likely that the one used for the experiments presented herein is not the optimal configuration.

6.3.1. Image Translation Network

The image translation CNNs $T_{\mathcal{X}}$ and $T_{\mathcal{Y}}$ have the same structure as in Luppino *et al.* [1]. As depicted in Figure 6, the number of filters are $[100, 50, 20, c]$, where c denotes the the number of channels in the target domain. The images are padded such that the input and output has the same height and width.

The kernel size of the filters is 3×3, and the filters are initialized using the truncated normal scheme of Glorot and Bengio [126]. Zero initialized biases is used on all filters. Both the filter kernel and the biases are L_2 regularized.

Leaky ReLU with $\alpha = 0.3$ is used as activation function on all layers but for the last one, where $tanh(\cdot)$ is used, as the images are scaled to $[-1, 1]$. A dropout rate of 0.2 is used during training.

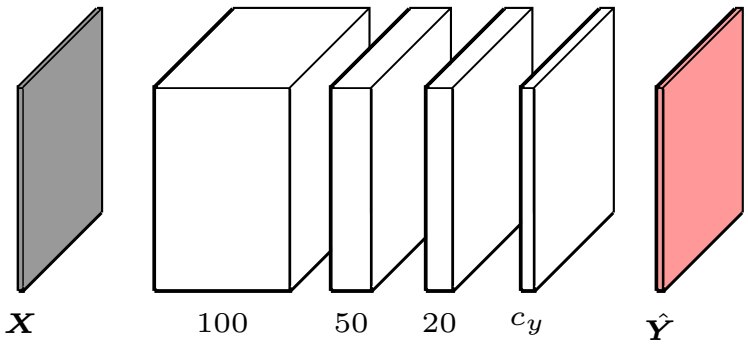


Figure 6: *Visualization of the filter bank used in the CNN that represents $T_{\mathcal{X}}$. The numbers indicates the number of filters in the layer, i.e. the number of output channels. The structure is likewise for $T_{\mathcal{X}}$, but with c_x rather than c_y.*

6.3.2. Training Details

6.3.2.1. Optimizer parameters

The Adam optimizer [99], described in Section 4.3.3, is used for the experiments. The TensorFlow default values of $\rho_1 = 0.9$, $\rho_2 = 0.999$ and $\delta = 1e-7$ was used, which are the same values suggested by Kingma and Ba [99], except for the numerical stability parameter $\delta = 1e-8$.

The learning rate ϵ was set to 10^{-4}. For the experiments with an exponential decay on the learning rate, a decay rate of 0.96 and a 10 000 step staircase scheme was used:

$$\epsilon_i = \epsilon \cdot 0.96^{\lfloor i/10\,000 \rfloor},$$

where i is increased for each batch.

In the optimization, all gradients were clipped to have a maximum norm of 1. This was mainly done because of the small batch size, to stabilize the training if e.g. two unrepresentative patches constitute one batch.

6.3.2.2. Computational Hardware

The experiments were run on a server cluster with different CUDA GPUs, each with more than 11GB VRAM. Each model was trained on single GPU within the cluster.

6.3.2.3. Initial Change Map

The cross loss weights $\pi_{ij} \in \mathbf{\Pi}$ in Equation (11) are computed from an initial change map $\boldsymbol{C}_{\text{icm}}$. The ICM $\boldsymbol{C}_{\text{icm}}$ is computed using the Improved Prior

Computaton scheme of Luppino *et al.* [1], which is described in Section 3.6. The resulting ICMs are depicted in Section 8.

For the ICM computation was a patch size $k = 20$ and a stride $\Delta = 5$ used. The computations was performed at the images original size, resampled at half the size, and interpolated at double the sizes [1].

6.3.2.4. Affinity Computation

The functions $g_{\mathcal{X}}$ and $g_{\mathcal{Y}}$ is used throughout to denote the affinity computation in the domains $\mathcal{X}$ and $\mathcal{Y}$ respectively. This is done to emphasize the fact that there exist no universally best affinity computation, and that it should be selected to describe the affinity structure of the problem at hand.

Herein are experiments conducted on multispectral images and log-transformed SAR images. Therefore is it appropiate to use a Gaussian kernel affinity computation. The bandwidth of the kernel is set using the kNN scheme described in Section 3.5. This scheme is adaptive on patch level, which have empirically proven robust [1].

The objective function $J_{\mathtt{A}}$ term requires that the affinity matrices are aligned, and this requirement is assumed fulfilled by using the same patch adaptive bandwidth selection scheme in the two domains.

6.3.2.5. Limitations of the Affinity Computations

Computation of the affinity matrices has a high memory requirement, $\mathcal{O}(c \cdot (hw)^2)$, where c, h and w denote the number of channels, height and width of the image, respectively.

Due to memory constraints is the training of all models performed on patches of the images. The the result of the memory requirement of the affinity computation is that the affinity-guided models cannot process image patches as large as the non-affinity models.

To evaluate the effect of the smaller patch size, are two variations of the affinity computation included in the experiments: one where the entire image patch is used to compute the affinity matrices, and one where several affinity matrices from non-overlapping subpatches of the image patches are computed and compared.

The latter variation consumes less memory, but the affinity information is also more local. For the affinity loss term, this is equivalent to working on larger batches of smaller images, which means that some longer-range relational information is not exploited. Also the other loss terms may be hypothesized to perform better with larger patches, which is why we would like to examine the effect of patched affinity computations and highlight the potential trade-off between computational speed and performance gain.

6.3.2.6. Training Data Sizes

The Patch Size (PS), Batch Size (BS), Number of Batches (NB) and Affinity Patch Size (APS) for the three types of models is reported in Table 2. The sizes are chosen within the bounds of the hardware (GPU memory) such that the number of pixels presented to the networks per epoch is the same for all the configurations. This patched affinity model configuration lends that the affinity matrices are computed on 16 subpatches of each batch.

Table 2: *Patch Size (PS), Batch Size (BS), Number of Batches (NB), Affinity Patch Size (APS) and Pixels per Epoch (PE) for the three types of models.*

Model	PS	BS	NB	APS	PE
Affinity Models	64×64	2	32	·	2^{18}
Non-Affinity Models	128×128	2	8	·	2^{18}
Patched Affinity Models	128×128	2	8	32	2^{18}

6.3.2.7. Data augmentation

For each batch are BS random patches of size PS×PS extracted. Each patch is rotated $0, 90, 180$ or 270 degrees with a $25\,\%$ probability, and horizontally flipped with a $50\,\%$ probability. These augmentations do not introduce any artifacts in the images.

6.3.3. Evaluation Details

The evaluation phase, e.g., the change map computation, follows the one of X-net [1] with minor adjustments. The flow is depicted in Figure 3. The specifics of the computation of the difference image, the filtering and the thresholding is as follows.

6.3.3.1. Difference Image

The computation of the difference image is twofold. First is a difference image computed in each image domain. These computations are performed pixel-wise like

$$\boldsymbol{D}_{\mathcal{X}} = \{d_{ij}^{(\mathcal{X})}\}_{i=1,j=1}^{h,w}, \tag{16}$$

where $d_{ij}^{(\mathcal{X})} = \|\boldsymbol{x}_{ij} - \hat{\boldsymbol{x}}_{ij}\|_2$. To handle outlier values is the mean μ and the standard deviation σ of the d_{ij}s computed, and any value larger than $\mu+3\sigma$ is set to this value. Then, the difference image is normalized to $[0, 1]$.

Luppino *et al.* [1] combined the two domain difference images by averaging them. We figured that the average should be weighted by the number of

channels in the domain the transformed image originated, to account for the amount of inferred information.

$$\boldsymbol{D} = \frac{c_x \cdot \boldsymbol{D}_{\mathcal{Y}} + c_y \cdot \boldsymbol{D}_{\mathcal{X}}}{c_x + c_y}. \tag{17}$$

The rationale is that the image translation should be easier when going from e.g. 11 to 3 channels, than the other way around, as the first is a compression problem and the latter an ill-posed inverse problem. Intuitively, it will be an advantage to use the changes extracted from the domain with the fewer channels, as this is an easier image translation problem.

6.3.3.2. Filtering

The filter used for on $\boldsymbol{D}$ before thresholding it is a based on the fully connected conditional random field model of Krähenbühl and Koltun [127]. It defines a pairwise potential between each pixel to filter the image using the spatial context. See Luppino *et al.* [1] for more details. The hyperparameters are set to 3 iterations and a kernel width of 0.1.

6.3.3.3. Thresholding

The final step of the evaluation phase is to threshold the difference image to produce the final binary change map. To not introduce undue complexity, and evaluate the performance of the image translation model as a mean to change detection, is Otsu's thresholding [28] used. It is possible that a more complex thresholding schemes, see Section 2.3.3, could yield some improvement.

Part III – Experiments

7. Experimental Setup

The Affinity-Guided X-net (AX-net) proposed in Section 6.2 has a quite complex loss function with three components. It seems inadequate to compare the model to other methods that aims to solve similar problems without understanding how the three loss terms works in conjunction. It is also of interest to study whether loss terms have an isolated effect, or if their effectiveness is synergetic. The main experiment conducted to evaluate the model is therefore an ablation study [128] on the different loss terms.

7.1. Ablation Study

Ablation is an experimental method to look into causality [3], and the aim of an ablation study in machine learning is to identify parts of a model that do not contribute to the inference the model performs. It is likely that bits and pieces can be removed from many deep learning setups, without substantial change in the performance [3]. The motivation for ablation studies can be tied to the principle of Occam's razor, which can be phrased as the assertion that *an explanation of the facts should be no more complicated than necessary* [130].

The initial question in the experiment design was: "What can be removed while the model still produces meaningful output?" This lends to explore the hypothesis that the affinity–guiding loss term can be useful for training paired image–to–image translation networks. The ablation study, which attempts to isolate the contribution of each individual loss term, is constructed to test this hypothesis.

Two experiments were conducted to better understand the affinity loss term. The first experiment is limited to ablate the loss terms, with all other circumstances being equal. After the first experiment, several models were excluded, as they did not produce meaningful output. In the second experiment the effect of computing the cross-domain affinities on smaller patches is explored. The experiments are designed to justly compare the different submodels.

7.1.1. Naming scheme

The loss terms are indexed with `A` for the affinity-guiding term, `C` for the cyclic term, and `X` for the weighted cross-loss term, and referred to with the respective three-character codes: `A--`, `-C-`, `--X`. When two or three loss

terms are combined to train a model, the corresponding combination of letters is used, e.g. `-CX` refers to X-net as visualized in Figure 4 and `ACX` to the Affinity-Guided X-net in Figure 5.

All models are trained with an L_2 regularization term on the weights. The regularization term is thus not included in the naming scheme.

7.1.2. Metrics

Change detection can be considered a binary classification problem, with the classes *changed* or *unchanged* given to each pixel. The datasets used for the experiments, presented in Section 8, contain $20 - 30\,\%$ changed pixels. This should be considered an unbalanced classification problem, and appropriate metrics should be used.

Cohen's Kappa (κ) is designed to account for unbalanced classes, and is used as the main metric. This allows for comparison with the performance of X-net reported in Luppino *et al.* [1].

The performance according to κ is reported in box plots based on several random initializations of the networks. The boxes are drawn from the 25th percentile to the 75th percentile, with the median indicated with an orange line. The whiskers are drawn at the 5th and 95th percentile.

Delgado and Tibau [118] argue that Cohens's Kappa should be used with care, based on experiments with some quite extreme cases. To verify that κ is a suitable metric, the F1 Score and Matthews Correlation Coefficient (MCC) [121] is reported for some of the models.

7.1.3. Loss Term Weights

The loss term weights $\lambda_{(\cdot)}$ were set manually, aiming to balance the influence of the main terms while keeping the impact of the regularization term lower. The λ values reported in Table 3 were used for all the experiments.

Table 3: *Loss term weights for the experiments. The symbol · indicates not applicable.*

	`A--`	`-C-`	`--X`	`AC-`	`A-X`	`-CX`	`ACX`
$\lambda_{\mathtt{A}}$	1.0	·	·	0.8	1.0	·	0.8
$\lambda_{\mathtt{C}}$	·	1.0	·	1.0	·	1.0	1.0
$\lambda_{\mathtt{X}}$	·	·	1.0	·	1.0	0.8	0.8
$\lambda_{\mathtt{R}}$	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5

7.1.4. Experiment One – All Subsets of Loss Terms

To assess how each loss term contribute to the parameter optimization, it is natural to remove the loss term to see how the optimization turns out without it.

In this experiment all one, two and three line combinations of Equation (15) are used to train the image translation networks. The regularization term in Equation (15d) was used in all models. The models are:

- `A--`, corresponding to Equation (15a) and Figure 2;
- `-C-`, corresponding to Equation (15b) and Figure 7 (a);
- `--X`, corresponding to Equation (15c) and Figure 7 (b);
- `AC-`, corresponding to Equations (15a) and (15b) and Figure 7 (c);
- `A-X`, corresponding to Equations (15a) and (15c) and Figure 7 (d);
- `-CX`, corresponding to Equations (15b) and (15c) and Figure 4; and
- `ACX`, corresponding to Equation (15) and Figure 5.

To get decent performance statistics each method is initialized 15 times and trained for 100 epochs. For each epoch are the networks presented with $2^{18} = 262\,144$ pixels, as presented in Table 2 for affinity models and non-affinity models. The initial learning rate is set to $\epsilon = 10^{-4}$, and the learning rate is decayed exponentially, as described in Section 6.3.2. The loss term weights $\lambda_{(\cdot)}$ are set as presented in Table 3.

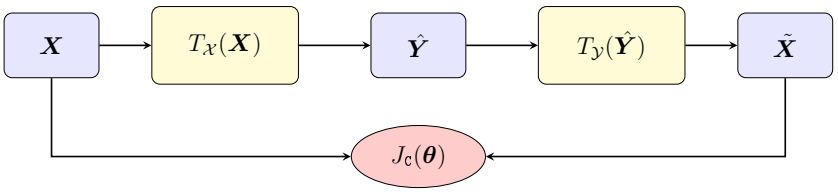
7.1.5. Experiment Two – Patched Affinity Computation

Based on the results of Experiment One are the models `-CX`, `A-X` and `ACX` selected for a more thorough analysis. The goal of this experiment is to gain a deeper understanding of the models that performed better than the deterministic baseline model described in Section 8.3.

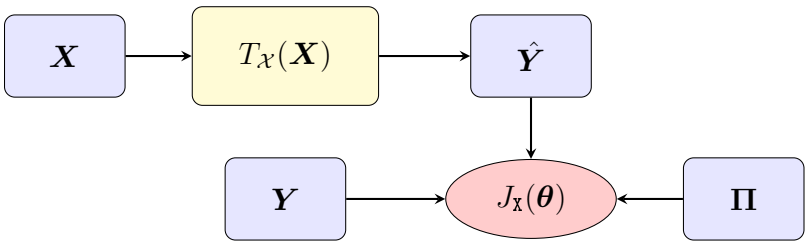
Due to the memory consumption of the affinity computation must the images presented to the affinity models be smaller than the ones presented to the non-affinity models. To understand the effect this has on the other loss terms, and to explore if smaller affinity patches yields similar performance, patched affinity computation is introduced.

The patched affinity computation divides an image patch into non-overlapping sub-patches, and computes the affinity of each sub-patch. This is equivalent to computing the affinity loss term for a larger batch of smaller patches, but the other loss terms still get to work with the larger patch.

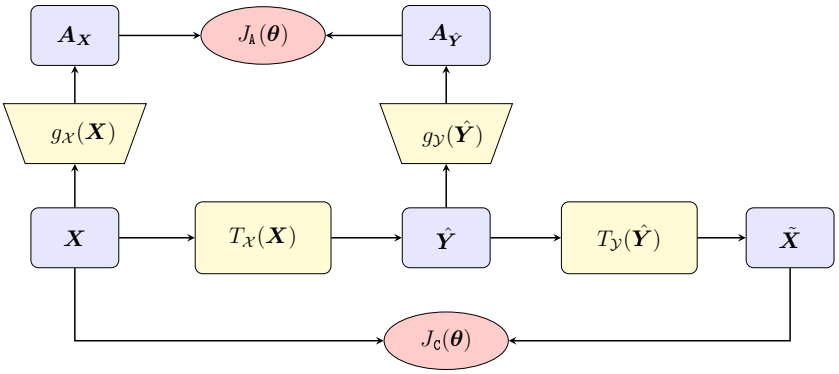
The dimensions of the training patches, affinity patches, and batches are as presented in Table 2. A constant learning rate $\epsilon = 10^{-4}$ is used. For better statistics than in experiment one is each model initialized 35 times and trained for 100 epochs. The loss term weights $\lambda_{(\cdot)}$ are set as presented in Table 3.



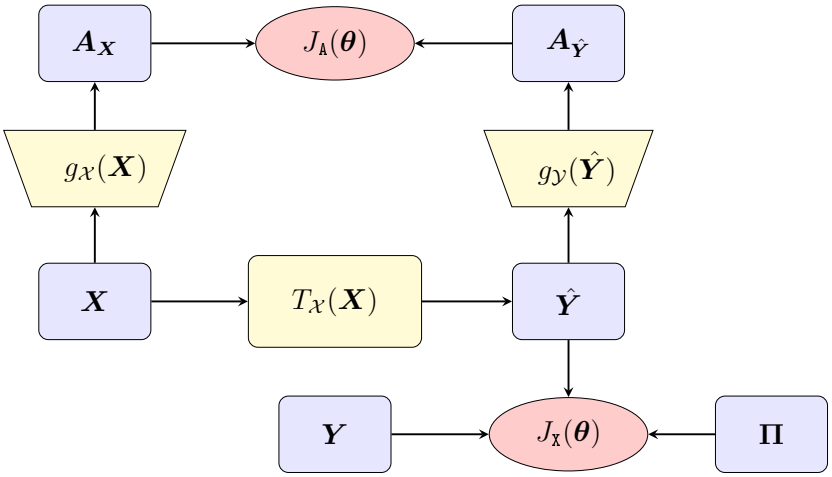
(a) `-C-`



(b) `--X`



(c) `AC-`



(d) `A-X`

Figure 7: *The submodels used in the ablation study that are not described in Sections 5 and 6.*

8. Datasets

The two datasets used for the experiments are presented in the following. The data is augmented and batched as described in Section 6.3.2.

8.1. Texas Dataset

The images in this dataset were captured before and after a forest fire in Bastrop County in Texas during the months of September and October 2011 [1].

The Landsat 5 Thematic Mapper (TM) instrument acquired a multispectral image with 7 image bands before the event, depicted in Figure 8 (a). The Earth Observing-1 Advanced Land Imager (EO-1 ALI) acquired a multispectral image with 10 bands after the event, depicted in Figure 8 (b). The ground truth, provided by Volpi *et al.* [48], is depicted in Figure 8 (c).

The images are registered and cropped to 1520×800 pixels, and are displayed in false color. Some the spectral bands coincide, i.e. the land cover signatures are partly similar [1].

8.2. California Dataset

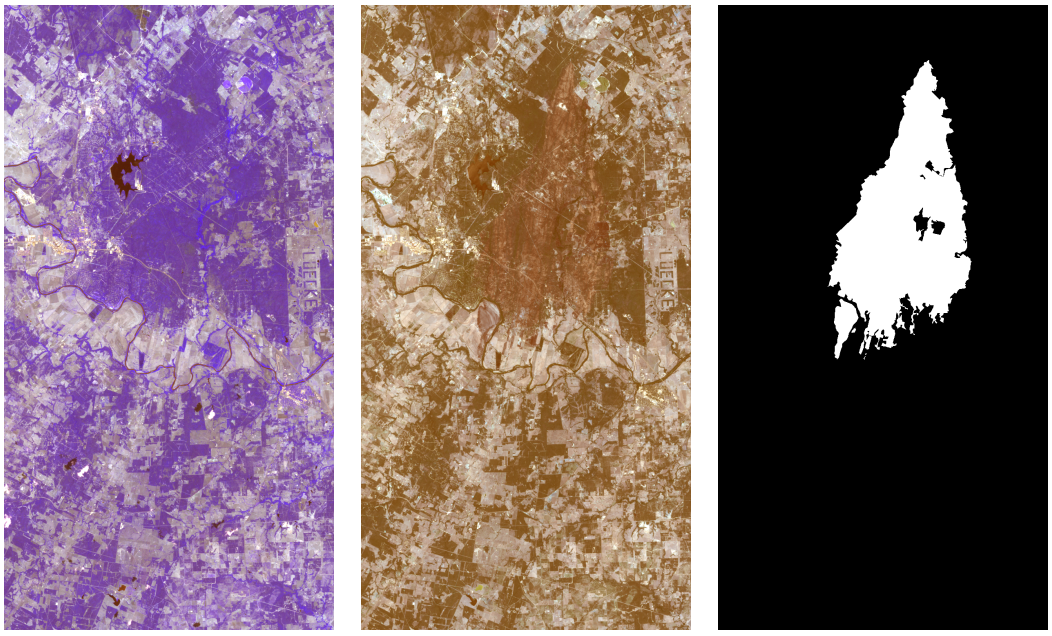
An area in Sacramento County, Yuba County and Sutter County, California was flooded during January and February 2017 [1].

Landsat 8's Operational Land Imager (OLI) and Thermal Infrared Sensor (TIRS) acquired a multispectral image with 11 image bands on January 5th 2017. The 11 bands cover the range from deep blue to long-wave infrared. The RGB channels are depicted in Figure 9 (a). Sentinel-1A acquired a Synthetic Aperture Radar (SAR) with polarizations VV and VH on February 18th 2017. The image is augmented with the ratio between the two intensities as the third channel [1], and is depicted in false color in Figure 9 (b).

The images are registered at a 3500×2000 resolution, but resampled to 850×500 to reduce the computation time. The ground truth, provided by [2], is depicted in Figure 9 (c).

8.3. Deterministic Baseline

The initial change map used as weights in the cross–loss term is also used as a deterministic baseline for the experiments. The ICM scheme [1] is described in Section 3.6 and the implementation details is described in Section 6.3.2. The ICM is filtered and thresholded as described in Section 6.3.3 produce a change map, which is evaluated alongside the other models. In Figures 10 and 11 are qualitative results for the baseline on the two datasets presented.

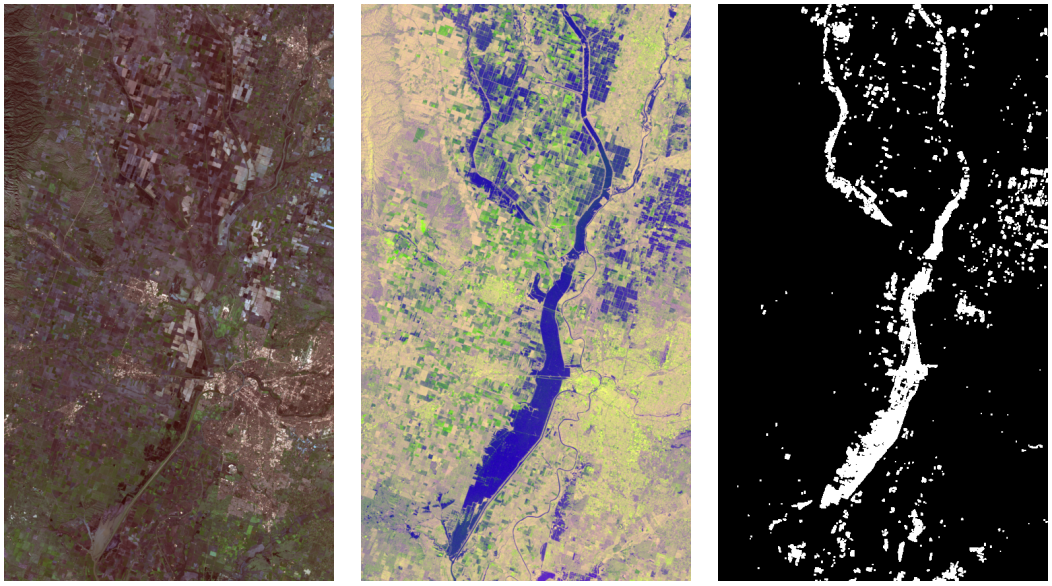


(a) $\boldsymbol{X}$, *Landsat 5 image displayed in false colors.*

(b) $\boldsymbol{Y}$, *EO1-ALI image displayed in false colors.*

(c) *Ground truth change map [48].*

Figure 8: *Forest fire in Texas. Referred to as the Texas dataset.*

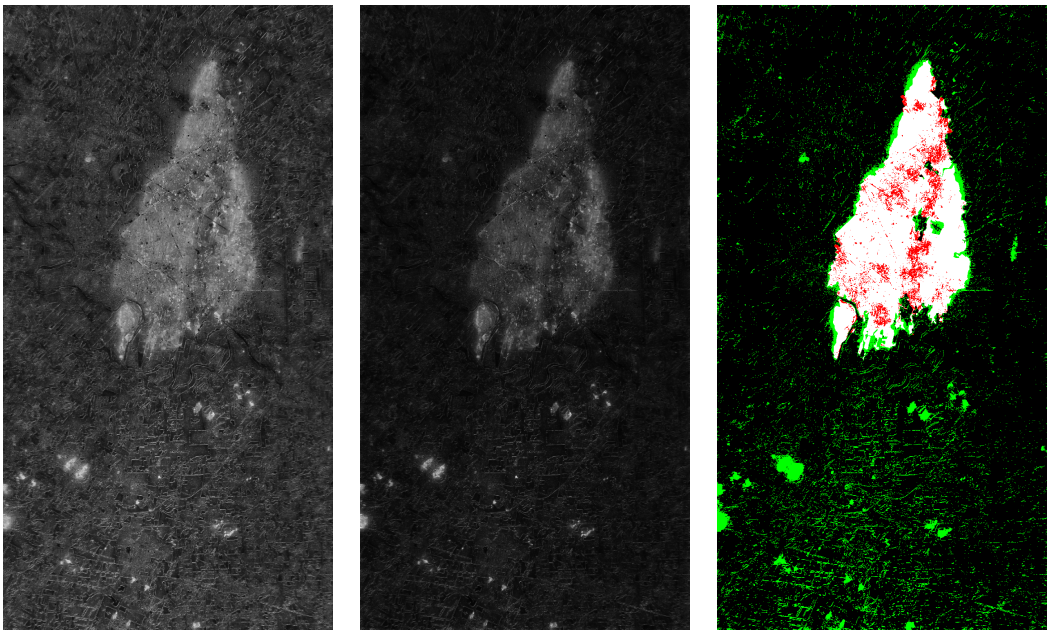


(a) $\boldsymbol{X}$, *Landsat 8 image displayed in RGB.*

(b) $\boldsymbol{Y}$, *Sentinel-1A image displayed in false colors.*

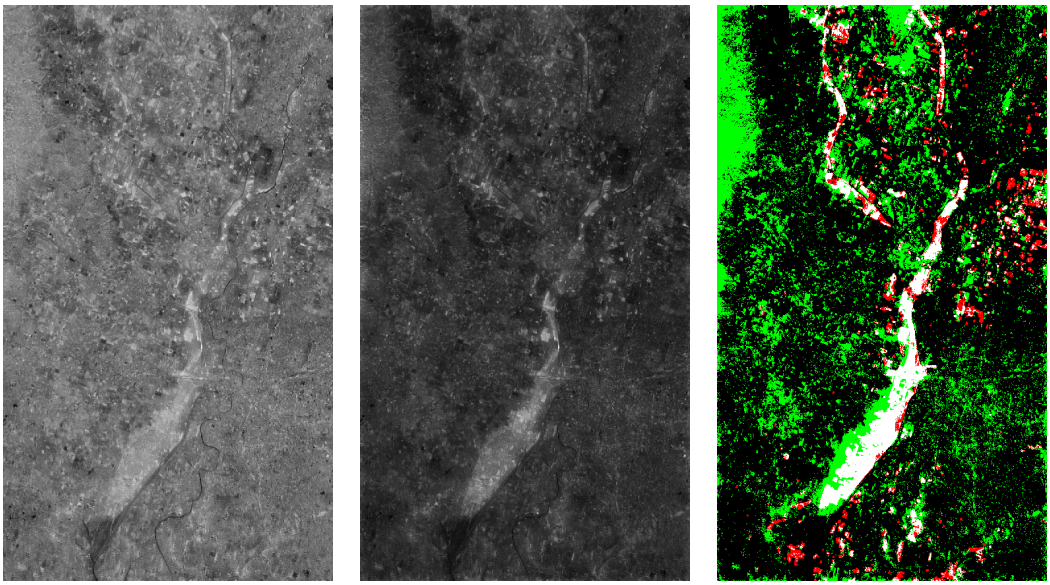
(c) *Ground truth change map [2].*

Figure 9: *Flood in California. Referred to as the California dataset.*



(a) *Initial change map.* (b) *Gaussian filtered ICM.* (c) *Confusion map.*

Figure 10: *Qualitative results for the deterministic baseline on the Texas dataset.*



(a) *Initial change map.* (b) *Gaussian filtered ICM.* (c) *Confusion map.*

Figure 11: *Qualitative results for the deterministic baseline on the California dataset.*

9. Results

9.1. Experiment One – All Subsets of Loss Terms

The first experiment was only performed on the Texas dataset, as this is the easiest of the two [2]. For this initial experiment only a quantitative analysis of the results is presented, as there are too many models whose performance is close to *a chance classification* for a qualitative study to be viable. A box plot of Cohen's κ is found in Figure 12.

9.1.1. Models Without Sample Selection

With their 25th to 75th percentile spanning the range $\kappa = 0$ to $\kappa = 0.2$, it is evident that the loss terms of the models `A--`, `AC-` and `-C-` are inadequate to solve the image–to–image translation problem. They hardly perform better than chance classification compared to the ground truth.

It is not surprising that the cyclic term alone, i.e. `-C-`, is not able to train the two translation networks to work together. The two halves of the model $(T_{\mathcal{Y}} \circ T_{\mathcal{X}})(\boldsymbol{X})$ and $(T_{\mathcal{X}} \circ T_{\mathcal{Y}})(\boldsymbol{Y})$ are optimized in parallel towards their respective targets $\min_{\boldsymbol{\theta}} \|\boldsymbol{X} - (T_{\mathcal{Y}} \circ T_{\mathcal{X}})(\boldsymbol{X})\|_2^2$ and $\min_{\boldsymbol{\theta}} \|\boldsymbol{Y} - (T_{\mathcal{X}} \circ T_{\mathcal{Y}})(\boldsymbol{Y})\|_2^2$, but there is no mechanism to pull the two optimization targets toward each other. As can be seen from e.g. `-CX` in Figure 12, a much better solution exists, but the loss term of `-C-` is not sufficient for the optimization scheme to find it.

It is more surprising that the models `A--` and `AC-` performs on par with `-C-`. The data flow is similar to `--X` and `-CX`, and one might expect performance on par with these two models respectively. The underlying assumption is that the affinity matrices capture the structure of the two image domains without changes, and that the guidance on how to translate this structure can act as a mechanism to pull the two halves toward a common solution during the optimization. This does not seem to happen.

The main explanation is probably the lack of a priori information. The three models `A--`, `AC-`, and `-C-` do not include the cross-loss term `--X`, which is weighted with the initial change map (ICM). Although noisy, this underlying sample selection seems to be of major importance. There is a clear difference between the methods with and without ICM-based cross-loss, and it is safe to conclude that the models without this sample selection are simply not able to solve the image–to–image translation problem.

It might seem peculiar that the affinity loss term alone is not able to describe the same information as the initial change map, but although the computation is based on similar principles, the data differs. For the ICM the affinity matrices of the two input images $\boldsymbol{X}$ and $\boldsymbol{Y}$ are computed and compared. In the affinity loss term the affinity matrices of the input images

$\boldsymbol{X}$ and $\boldsymbol{Y}$ are compared against the ones of the translated images $\hat{\boldsymbol{Y}}$ and $\hat{\boldsymbol{X}}$ respectively. This does not highlight change information to act as a sample selection, but rather enforces that the affinity structure of the transformed image is similar to the input image.

For `A--` is it likely that the image translation network learns something resembling an identity transformation. There is nothing in the loss term encouraging it to do otherwise. This might also be true for `AC-` and `-C-`. It is possible that the image translation networks have too large expressive power, and that there exists some parameter configuration that allows for an near-identity mapping through the composition of the two translation networks.

9.1.2. Models With Sample Selection

The horizontal, red line in Figure 12 indicates the deterministic baseline. The baseline is the ICM used to weight the cross-loss term `--X`, and it is therefore surprising that `--X` perform consistently worse than it.

The baseline does not attempt to solve the image translation problem, so a direct comparison of `--X` and the baseline is hard. However, they both attempt to solve the same change detection problem, for which purpose the baseline is superior. The confusion map of the baseline is found in Figure 10, while the confusion map of the best[15] `--X` model is found in Figure 13 (h).

It seems like part of the explanation for the `--X` architecture's inferior performance is that some local classes, e.g. the water bodies on the left of the changed area and to the far north, are not translated well. These are thus perceived as a change. This must be considered an artifact of the image translation process, but ends up affecting the final change detection performance.

There are also more false negatives (red) in the confusion map of `--X`, mostly found in areas that the ICM indicates as changed. Part of the explanation might be the class balance, as the changed area constitutes a smaller part of the image. This theory is supported by the fact that the confusion map of `--X` is much cleaner outside of the changed area.

The image pairs (a), (b) and (d), (e) in Figure 13, summarized by the respective difference images in (c) and (f), indicate that that `--X` has a decent performance on the image translation problem. It takes a skilled eye to distinguish the original and translated images. Its change detection performance is nevertheless inferior to the ICM, and the model is not included in the further experiments.

[15] As judged by the κ value.

The models `A-X`, `-CX` and `ACX` perform better than the baseline. This indicates that the weighted cross-loss is the most important loss term, but it needs to be combined with more information for the image translations to perform well in a change detection system. These three models will be explored in more detail in the other experiments.

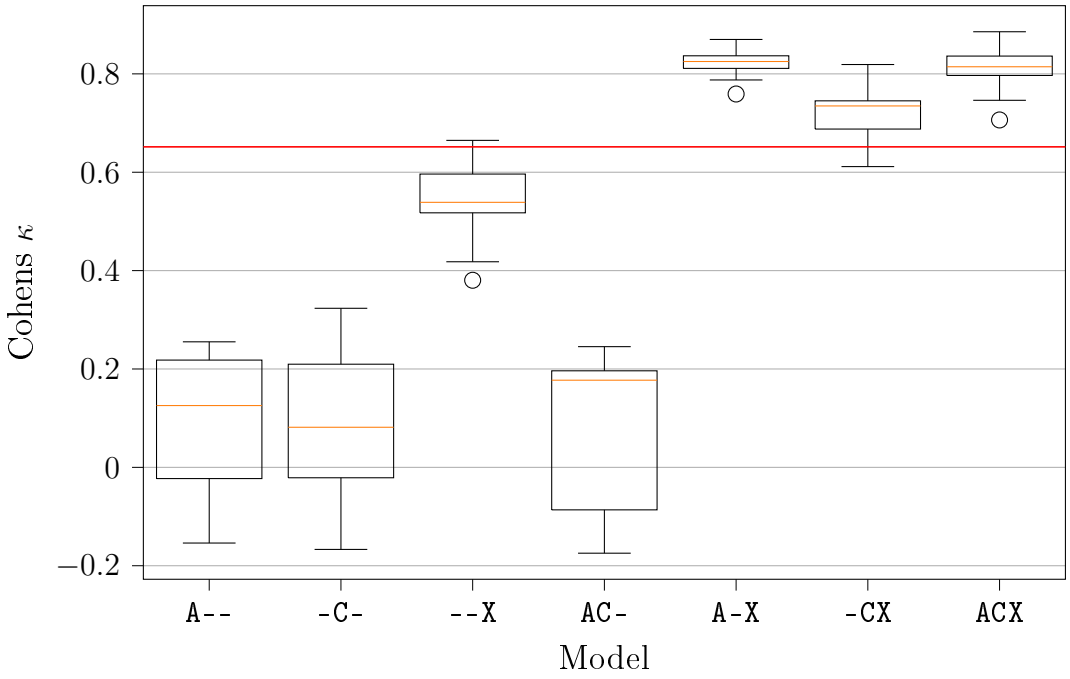
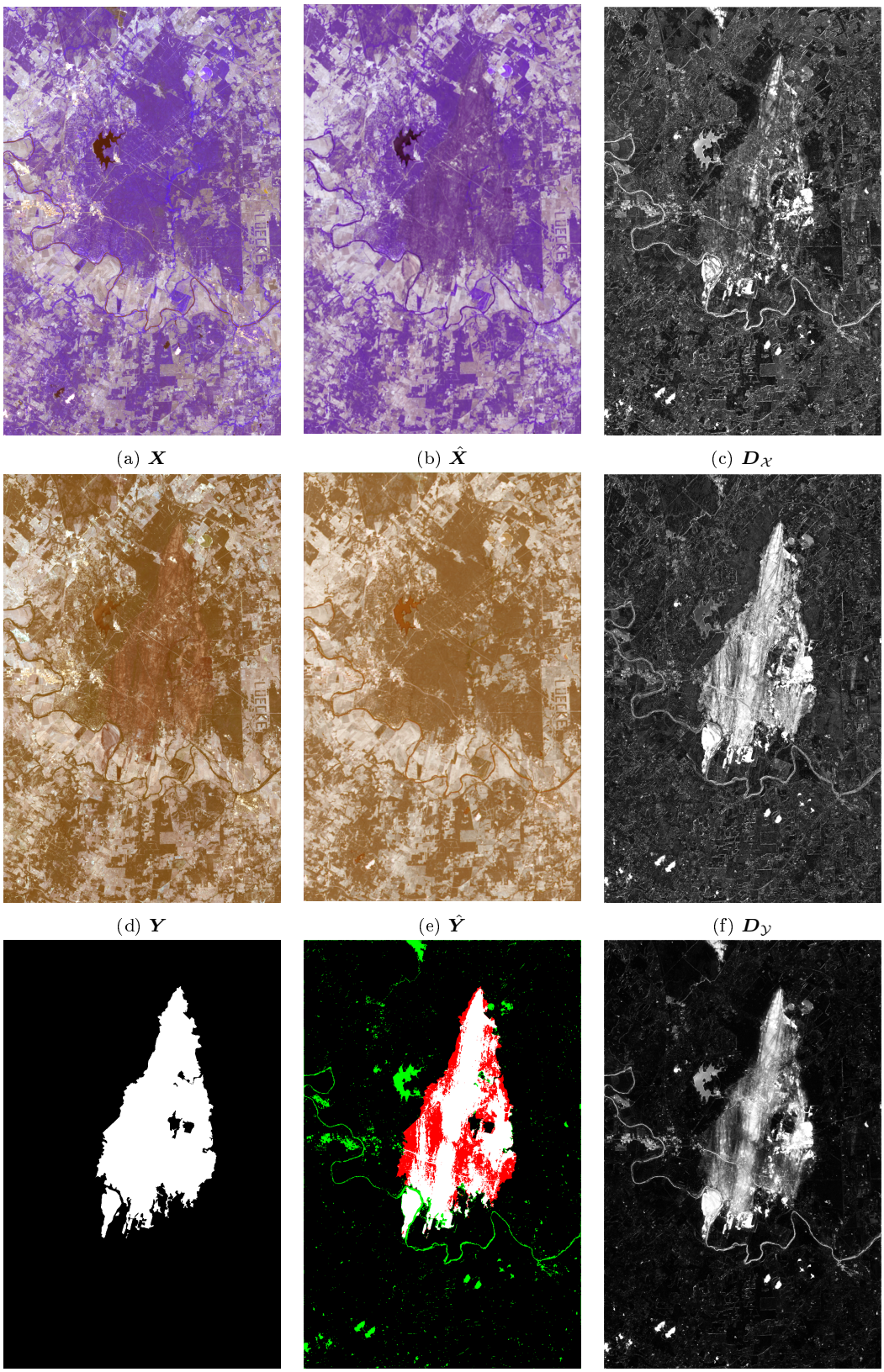


Figure 12: *Cohen's κ for the different models in the full ablation study on the Texas dataset. Each model was trained for 100 epochs from 15 different initializations. The red line indicates the deterministic baseline as described in Section 8.3.*



(a) $\boldsymbol{X}$ (b) $\hat{\boldsymbol{X}}$ (c) $\boldsymbol{D}_{\mathcal{X}}$

(d) $\boldsymbol{Y}$ (e) $\hat{\boldsymbol{Y}}$ (f) $\boldsymbol{D}_{\mathcal{Y}}$

(g) $\boldsymbol{C}_{gt}$ (h) *Confusion map* (i) $\boldsymbol{D}_f$

Figure 13: *The best result for* `--X` *according to* κ. *Cropped at the bottom to fit the page.*

9.2. Experiment Two – Patched Affinity Computation

The second experiment was performed on both datasets, to explore how the models perform also on the harder California dataset. Box plots of κ for the Texas and California dataset is found in Figure 14 (a) and (b). Note that the range on the y-axis is dramatically changed from Figure 12, and that the x-axis position of `-CX` and `A-X` is swapped as `-CX` is used as a reference point to argue whether affinity–guiding is useful in training paired image–to–image translation CNNs.

9.2.1. Non–Affinity Model

Even though the cyclic and cross–loss terms separately performed worse than the baseline, their performance is significantly better when combined. An explanation for their combined performance is that the cross–loss term indicates how unchanged pixels should be translated, while preventing the cyclic term from utilizing identity-like paths through the networks.

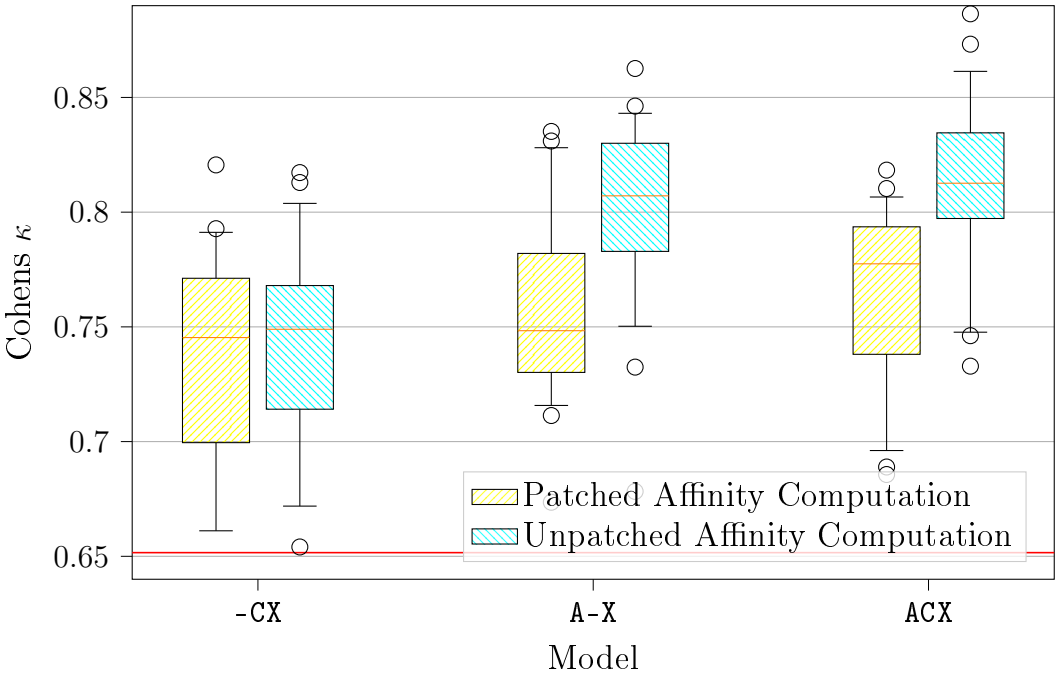
At the same time does the cyclic term give indications on how both changed and unchanged pixels should be translated through the composition of the networks, and adds two contributions to each gradient update. That is, a parameter $\theta \in \boldsymbol{\theta}$ associated with the CNN $T_{\mathcal{X}}$ will be updated based on its contribution to the difference between $\tilde{\boldsymbol{Y}} = (T_{\mathcal{X}} \circ T_{\mathcal{Y}})(\boldsymbol{Y})$ and $\boldsymbol{Y}$, as well as its contribution to the difference between $\tilde{\boldsymbol{X}} = (T_{\mathcal{Y}} \circ T_{\mathcal{X}})(\boldsymbol{X})$ and $\boldsymbol{X}$.

The combination of the two loss terms yields an synergistic effect. The two loss terms complement each other, and their combined contribution to the parameter updates become greater than the sum of its parts.

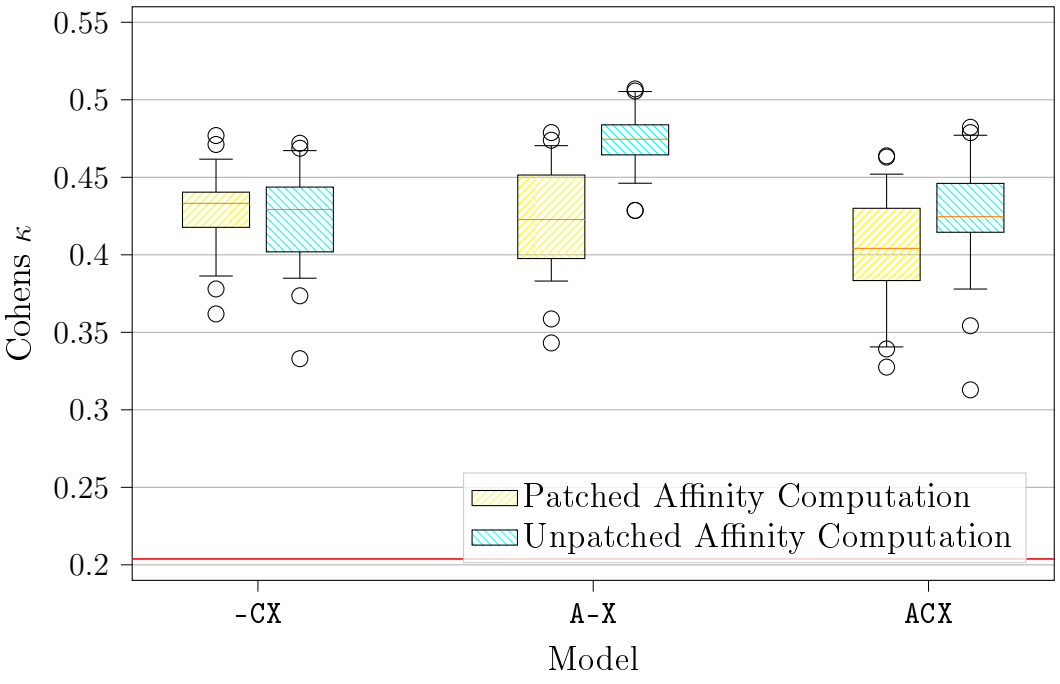
As `-CX` does not include the affinity loss term, the architecture is not affected by the affinity patching, as seen in the subplots in Figure 14.. That is, the two boxes for `-CX` in each subplot are visualizations of the distribution of 35 realizations of the same random variable. Some small variations can be observed, but these are likely to be statistical fluctuations due to the limited number of runs of each algorithm.

Indeed, the performances of the patched and the unpatched experiment runs are reasonably similar, save from not so relevant shifts of percentiles that could be attributed to statistical variations. Nevertheless, this indicates that all box plots should be interpreted with some caution. An analysis of statistical significance of differences between the algorithms and implementations is admittedly called for, but has not been conducted due to time limitations.

As reported by Luppino *et al.* [1], `-CX` (X-net) performs well on the two datasets, with a performance in terms of κ that is very consistent with the one reported here.

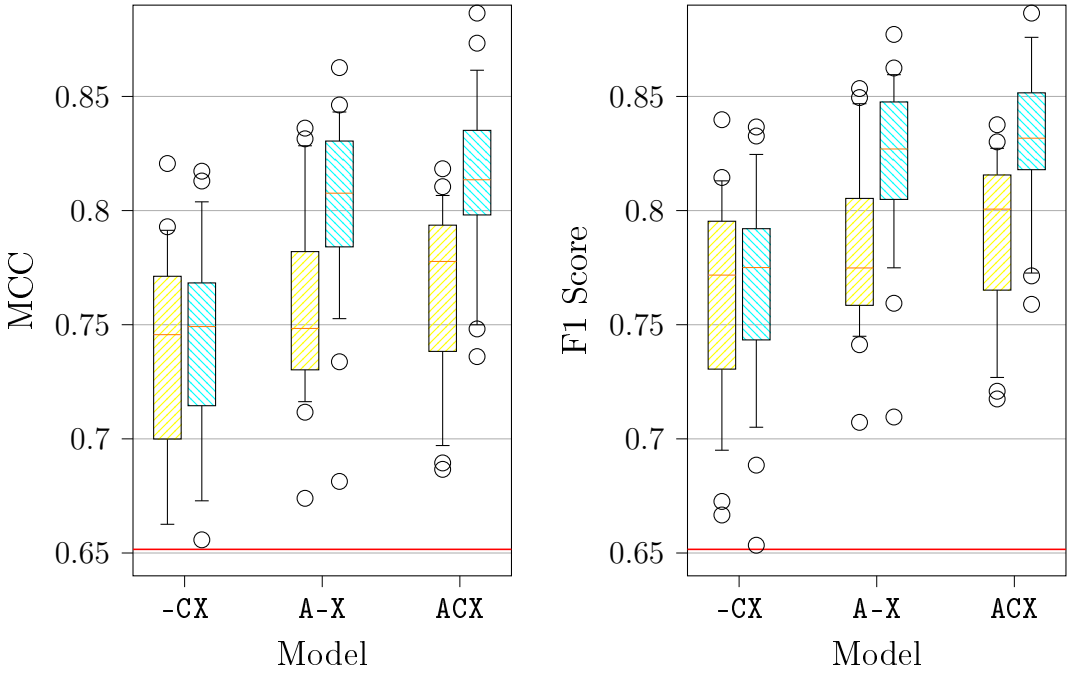


(a) *Texas dataset.*



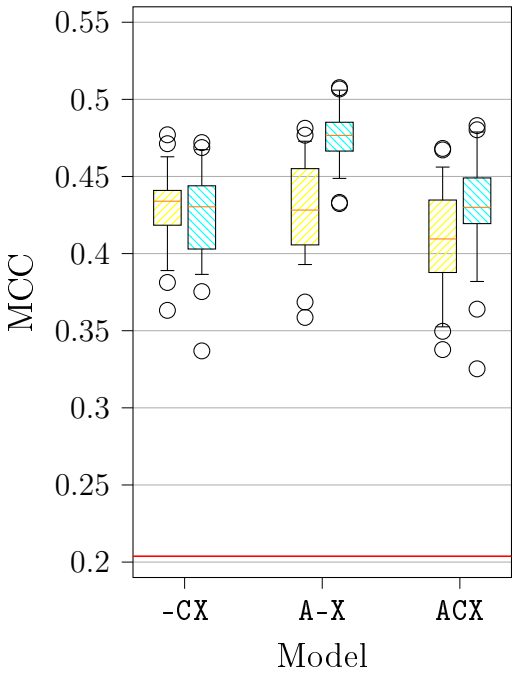
(b) *California dataset.*

Figure 14: *Effect of Patched Affinity Computation on Cohen's κ. Each model was trained for 100 epochs from 35 different initializations. The red line indicates the deterministic baseline as described in Section 8.3.*

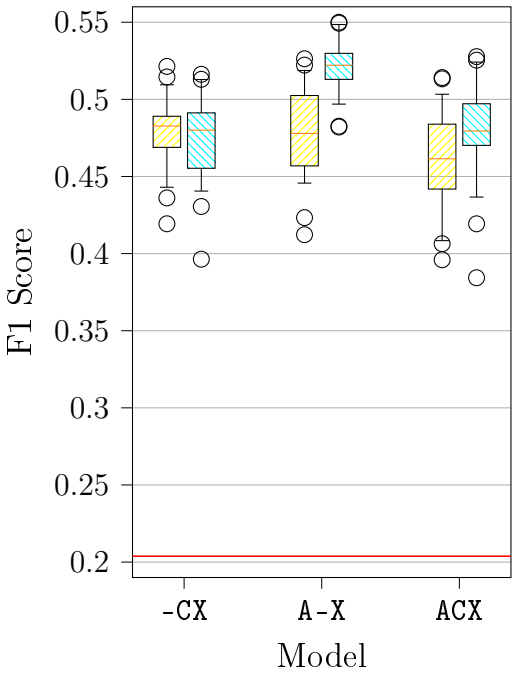


(a) *MCC on Texas dataset.*

(b) *F1 Score on Texas dataset.*



(c) *MCC on California dataset.*



(d) *F1 Score on California dataset.*

Figure 15: *Effect of Patched Affinity Computation on MMC (left) and F1 Score (right). The legend is the same as in Figure 14. Each model was trained for 100 epochs from 35 different initializations. The red line indicates the deterministic baseline as described in Section 8.3.*

9.2.2. Patched Affinity Models

For the patched models, indicated with yellow boxes in Figures 14 and 15, the affinity loss term is computed on 16 subpatches of size 32×32, while the other loss terms are computed on patches of size 128×128.

9.2.2.1. `A-X` Architecture

Neither the cross–loss nor the affinity loss term did perform well alone, but when combined in `A-X` they perform quite well, even for small affinity patches. This indicates that the intuition that lead us to propose the affinity loss term might be good.

The cross–loss term indicates how the unchanged pixels should be translated, while the affinity loss indicates how these translations shall be dispersed to the changed areas by describing the desired neighborhood structure after the transformation.

The distribution of κ for the patched version of `A-X` on the Texas dataset has a median on the same level as `-CX`, and a similar distance between the 25th and 75th percentile. Unlike for `-CX`, the median is shifted towards worse performance. Combined with the 95th percentile whisker being much longer than the 5th percentile whisker, this gives a heavy tail towards better performance. On the California dataset the performance of `A-X` is on par with `-CX`, with a fairly symmetric distribution and the same number of outliers.

Based on the performance in these experiments does `A-X` seem like an alternative that is on par with `-CX`. If the heavy tail on the Texas dataset is considered, it might even be preferred. One advantage with `A-X` is that it can be used to train only one translation CNN, as the tandem network setup is not required without the cyclic loss term. Such a single network setup can be trained even with small affinity patches.

9.2.2.2. `ACX` Architecture

With the good performance of `-CX` and `A-X`, it is expected that the joint model should be at least as good. The combined effect of the weighted, direct training targets of the cross–loss term, the dispersion effect of the affinity loss in the neighborhood around unchanged pixels, and the cyclic consistency should yield solid performance.

With small affinity patches on the Texas dataset, `ACX` has the highest median performance of the three models. It seems like substantial improvement at first glance, but the distribution of κ has a heavy tail towards worse performance. On the California dataset the `ACX` performs slightly worse than the two other models. An explanation might be that the cyclic and affinity terms together overrule the cross–loss term, which seems like

the most important term judging from experiment one. If this is the explanation, then the lower performance of `ACX` is due to bad hyperparameter settings.
From these experiments it seems like `A-X` should be favored over `ACX` for small affinity patches, especially if the worst case performance on the Texas dataset is considered.

9.2.3. Unpatched Affinity Models

All loss terms are computed on patches of size 64×64, with an increased number of batches to balance the number of pixels per epoch. These are the same dimensions as in experiment one, and the performance is consistent with what was observed there. These models are indicated with teal boxes in Figures 14 and 15.
The performance of the two affinity models with this experiment setting is comparable on the Texas dataset. The performance is clearly better than the `-CX` models.
On the California dataset the performance of `A-X` is clearly better, with the worst outlier on par with the median performance of the other models. The `ACX` model does not match our expectations, given a performance on par with the patched version of `A-X`. The explanation can be the hyperparameter configuration, which may cause the cyclic and affinity loss terms that did not work alone and together to overrule the cross–loss term.

9.2.4. Matthews Correlation Coefficient and F1 Score

To verify that Cohen's Kappa is a suitable metric to compare the models, the Matthews Correlation Coefficient (MCC) and F1 Score were also computed. These are reported in Figure 15. The range of the y-axis is static for each dataset across Figures 14 and 15, although the plots for MCC and F1 are compressed on the x-axis. Note that the F1 Score lies in $[0, 1]$, while κ and MCC lie in $[-1, 1]$. Thus is it to be expected that the F1 Score shall be a bit higher than the other two metrics.
MCC and κ are similar measures, and coincide for a symmetric confusion matrix [118]. Our general impression from conducting the experiments is that the confusion matrices are not far from symmetric, i.e. the number of false positives `FP` and false negatives `FN` are approximately the same. With this in mind, is it expected that κ and MCC have similar values. This is also the case, as can be seen from Figures 14 and 15 (a, b) and (a, c).
The F1 Score is a little higher than the two other metrics, but the shape of the distributions are the same. The three metrics also have a similar outlier structure. This indicates that κ is as good a choice as MCC or F1 Score to evaluate these experiments, and the performance descriptions in Sections 9.2.2 and 9.2.3 would be more or less the same if MCC or F1 Score was used as the main metric.

9.2.5. Training Times

The average training times of the three models are reported in Table 4. As expected, there is a significant difference in the training time, since the affinity loss term is a major addition to the computational complexity.

The performance gain of the unpatched variation of `A-X` could justify the increased training time. It should be noted that without the cyclic term there is a potential to optimize only $T_{\mathcal{X}}$ or $T_{\mathcal{Y}}$, and still optimize it with `A-X`. This would reduce the training time, as the number of affinity computations is halved. Furthermore, the affinity matrices of the input images can be precomputed, i.e., computed only once. This optimization would presumably decrease the training time.

When considering training time, the time spent to compute the initial change map also must be considered. For the Texas dataset it took 42 minutes, and for the California dataset it took 13 minutes as reported by Luppino *et al.* [1].

Table 4: *Average training times [mm:ss] for the three models, with unpatched (UP) and patched (P) times for the models with the affinity loss term.*

	`-CX`	`A-X`		`ACX`	
		P	UP	P	UP
Texas	07:04	13:37	28:03	13:43	27:43
California	03:22	09:28	23:38	09:35	23:31

Note on the training time of the affinity models: The affinity models produce an error in an optimization scheme in the TensorFlow graph compilation. The reports in the GitHub issue[16] is that the occurrence of this error impacts the performance. This may have impaired the training time for these models, but there is no doubt that the affinity models should be slower to train given the extra $\mathcal{O}(c \cdot (hw)^2)$ computations[17], where c, h, w denotes the channels, height and width of the image respectively.

9.2.6. General remarks

The affinity loss terms ability to disperse the information about unchanged pixels to the neighborhood around can be seen as the introduction of contextual information to the loss term. The experiments indicate that this might be beneficial, both with smaller and larger affinity patches. At the least, the unpatched `A-X` is the model that performs best if the performance on both datasets is considered.

The decreased performance from `A-X` to `ACX` on the California dataset might be tied to the loss term weights λ. They were set manually, at an early

[16] `https://github.com/tensorflow/tensorflow/issues/34499`

[17] The batch size is negligible in this case.

stage of the experiment timeline. It might be that the choice to set $\lambda_{\texttt{C}}$ higher than $\lambda_{\texttt{A}}$ and $\lambda_{\texttt{X}}$ was a bad decision, and that the cyclic and affinity term in conjunction overrules the cross–loss term in the training of `ACX`. Another explanation might be that this harder dataset requires a simpler loss function.

9.3. General Observations

The limited experiments conducted in the ablation study indicate that the affinity–guiding loss term might be useful in the training of image–to–image translation networks used for change detection. Although, the proposed model is only tested on two datasets, for which one has some overlapping spectral bands.

Experiments should be conducted on many more datasets of different complexity in an further effort to falsify the hypothesis herein. A general issue is the limited access to datasets with a ground truth, a well-known problem in remote sensing and earth observation research. The proposed change detection model is unsupervised. Nevertheless, there is a need for a ground truth to firmly evaluate the models performance.

In comparing `A-X` and `-CX`, the former offers one major benefit, namely that it can be used to train only one map, either $T_{\mathcal{X}}$ or $T_{\mathcal{Y}}$. This reduces the number of parameters to optimize, which is beneficial when working with a small training set. Such a setup would reduce the training time of the affinity model, as the number of trainable parameters is significantly reduced. Another mean to reduce the computation time is to precompute the affinity matrices of the input images, either for the full image or as a parallel process on the CPU during training on the GPU.

9.3.1. Critique of experimental setup

The experimental setup has several weaknesses and shortcomings, including the following:

- Due to time restrictions, no hypothesis tests for the significance of changes in the value of κ were performed. This would be a natural extension of this work.
- The values of the loss term weights $\lambda_{(\cdot)}$ were set manually for the experiments and remained fixed. This might have impacted the performance of especially `ACX`. This illuminates another important perspective within ablation studies, namely to keep the number of hyperparameters to a minimum.
- The ablation study only removed loss terms. It is likely that the network dimensions could have been trimmed without significant loss in performance.

- The gradient clip norm was adapted from [1] without reflection. This is an element of the training scheme that might have been removed, reducing the number of hyperparameters.
- No heuristic for early stopping was used. This would probably have been beneficial, as the models could conclude the training at will, rather than at a fixed number of epochs.
- The effect of weighting the average in the difference image computation (introduced in Equation (17)) was not evaluated.

Part IV – Conclusion

10. Concluding Remarks

In this thesis, we have proposed a new loss function which effectively contributes contextual information to the training process, which means that it has the ability to retain the information contained in a pixel's neighborhood. Whilst the other loss terms contain pixel-based differences, the affinity loss on the other hand explicitly uses contextual information encoded through the new graph-theoretic cross-domain distance. As such, it can be compared to well-known contextual methods like random Markov fields and conditional random fields, and notably offers a practical way of incorporating it in the training.

The experimental results indicate that the affinity loss term is useful when training image–to–image translation networks for earth observation remote sensing images. The extent of the experiments is not sufficient to claim that the affinity–guiding is beneficial in an absolute and general sense. Nonetheless, it is plausible that the affinity-guiding loss term is beneficial for paired image–to–image translation.

The approach designed and developed in this work is not easily transferable, as a new tandem of considered sensors requires a new training of the proposed architecture, either from scratch or as a fine-tuning of the set of parameters of previously trained image translation networks. On the contrary, latent-space approaches such as the ACE-Net [1] can be generalized by, for example, finding a common latent space $\mathcal{Z}$ for all the possible encoder–decoder pairs associated with each and every sensor, which allows for comparison of any number of representation domains.

However, the main approach to machine learning for change detection at the current is to use only one pair of images during training. That is, the model is trained to be dataset specific. In fact, when dealing with a sudden event it is more reasonable to train a model from scratch based on the first available images rather than to adapt a previous model to fit the problem at hand. For example, creating a model that can describe flooding in a city area given two sensors and also in a woodland area with other two sensors in another season is probably not tractable, or even a relevant problem.

Moreover, the adversarial training paradigms are known to be unstable and associated with difficult hyperparameter tuning [54]. These algorithms are also known to need a large amount of training data, which might not be sufficient in the case of bi-temporal remote sensing imagery. Instead, the `A-X` model has proven as simple as successful, and the scenario in which an even simper model can work, where image translation is done in only one direction, does not sound so absurd.

10.1. Future Work

Ideas for future work with the affinity loss term include:

- Test the model on more datasets.
- Explore how the affinity-guiding loss term perform with other initial change map schemes.
- Evaluate the performance of affinity-guiding in other image–to–image translation scenarios.
- Perform a fine grid search of the hyperparameters and loss weights to achieve the best performance across the different datasets.
- Find a suitable scheme for early stopping of the training.

Ideas for future work on the affinity computation include:

- Explore how small affinity patches can be used for `A-X` to still perform on par with `-CX`.
- Explore whether overlapping affinity patches can be a good trade–off between computation time and performance.
- Use convolutional spatial propagation networks (CSPN) [79] to estimate the affinity matrices. This will likely yield a significant decrease in the computation time, while it is likely that it does not significantly decrease the performance [75, 80]. Although, this must be considered an antithesis of the ablation study performed herein.

References

[1] L. T. Luppino, M. Kampffmeyer, F. M. Bianchi, R. Jenssen, S. B. Seprico, G. Moser, and S. N. Anfinsen, "Deep Image Translation with an Affinity-Based Change Prior for Unsupervised Multimodal Change Detection", *Unpublished*, 2019.

[2] L. T. Luppino, F. M. Bianchi, G. Moser, and S. N. Anfinsen, "Unsupervised Image Regression for Heterogeneous Change Detection", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 12, pp. 9960–9975, Dec. 2019. DOI: 10.1109/TGRS.2019.2930348.

[3] F. Chollet, *Ablation studies are crucial for deep learning research*, Tweet, Jun. 2018. [Online]. Available: https://twitter.com/fchollet/status/1012721582148550662 (visited on 12/10/2019).

[4] P. Coppin, I. Jonckheere, K. Nackaerts, B. Muys, and E. Lambin, "Review Article Digital change detection methods in ecosystem monitoring: A review", *International Journal of Remote Sensing*, vol. 25, no. 9, pp. 1565–1596, May 2004. DOI: 10.1080/0143116031000101675.

[5] D. Lu, P. Mausel, E. Brondízio, and E. Moran, "Change detection techniques", *International Journal of Remote Sensing*, vol. 25, no. 12, pp. 2365–2401, Jun. 2004. DOI: 10.1080/0143116031000139863.

[6] Y. Li, C. Peng, Y. Chen, L. Jiao, L. Zhou, and R. Shang, "A Deep Learning Method for Change Detection in Synthetic Aperture Radar Images", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 8, pp. 5751–5763, Aug. 2019. DOI: 10.1109/TGRS.2019.2901945.

[7] X. Niu, M. Gong, T. Zhan, and Y. Yang, "A Conditional Adversarial Network for Change Detection in Heterogeneous Images", *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 1, pp. 45–49, Jan. 2019. DOI: 10.1109/LGRS.2018.2868704.

[8] M. Gong, X. Niu, T. Zhan, and M. Zhang, "A coupling translation network for change detection in heterogeneous images", *International Journal of Remote Sensing*, vol. 40, no. 9, pp. 3647–3672, May 2019. DOI: 10.1080/01431161.2018.1547934.

[9] J. Liu, M. Gong, K. Qin, and P. Zhang, "A Deep Convolutional Coupling Network for Change Detection Based on Heterogeneous Optical and Radar Images", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 3, pp. 545–559, Mar. 2018. DOI: 10.1109/TNNLS.2016.2636227.

[10] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithms: A systematic survey", *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 294–307, Mar. 2005. DOI: 10.1109/TIP.2004.838698.

[11] A. Singh, "Review Article Digital change detection techniques using remotely-sensed data", *International Journal of Remote Sensing*, vol. 10, no. 6, pp. 989–1003, Jun. 1989. DOI: 10.1080/01431168908903939.

[12] S. Theodoridis and K. Koutroumbas, *Pattern recognition*, 4. ed. Amsterdam: Elsevier Acad. Press, 2009, ISBN: 978-1-59749-272-0.

[13] R. Touati, M. Mignotte, and M. Dahmane, "A Reliable Mixed-Norm-Based Multiresolution Change Detector in Heterogeneous Remote Sensing Images", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 9, pp. 3588–3601, Sep. 2019. DOI: 10.1109/JSTARS.2019.2934602.

[14] G. Mercier, G. Moser, and S. B. Serpico, "Conditional Copulas for Change Detection in Heterogeneous Remote Sensing Images", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 5, pp. 1428–1441, May 2008. DOI: 10.1109/TGRS.2008.916476.

[15] L. T. Luppino, S. N. Anfinsen, G. Moser, R. Jenssen, F. M. Bianchi, S. Serpico, and G. Mercier, "A Clustering Approach to Heterogeneous Change Detection", in *Image Analysis*, P. Sharma and F. M. Bianchi, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2017, pp. 181–192. DOI: 10.1007/978-3-319-59129-2_16.

[16] Z.-g. Liu, G. Mercier, J. Dezert, and Q. Pan, "Change Detection in Heterogeneous Remote Sensing Images Based on Multidimensional Evidential Reasoning", *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 1, pp. 168–172, Jan. 2014. DOI: 10.1109/LGRS.2013.2250908.

[17] J. Prendes, M. Chabert, F. Pascal, A. Giros, and J.-Y. Tourneret, "A New Multivariate Statistical Model for Change Detection in Images Acquired by Homogeneous and Heterogeneous Sensors", *IEEE Transactions on Image Processing*, vol. 24, no. 3, pp. 799–812, Mar. 2015. DOI: 10.1109/TIP.2014.2387013.

[18] M. Gong, P. Zhang, L. Su, and J. Liu, "Coupled Dictionary Learning for Change Detection From Multisource Data", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 12, pp. 7077–7091, Dec. 2016. DOI: 10.1109/TGRS.2016.2594952.

[19] B. Zitová and J. Flusser, "Image registration methods: A survey", *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, Oct. 2003. DOI: 10.1016/S0262-8856(03)00137-9.

[20] S. Wang, D. Quan, X. Liang, M. Ning, Y. Guo, and L. Jiao, "A deep learning framework for remote sensing image registration", *ISPRS Journal of Photogrammetry and Remote Sensing*, Deep Learning RS Data, vol. 145, pp. 148–164, Nov. 2018. DOI: 10.1016/j.isprsjprs.2017.12.012.

[21] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a Deep Convolutional Network for Image Super-Resolution", in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2014, pp. 184–199. DOI: 10.1007/978-3-319-10593-2_13.

[22] P. Zhang, M. Gong, L. Su, J. Liu, and Z. Li, "Change detection based on deep feature representation and mapping transformation for multi-spatial-resolution remote sensing images", *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 116, pp. 24–41, Jun. 2016. DOI: 10.1016/j.isprsjprs.2016.02.013.

[23] L. Bruzzone and D. Prieto, "Automatic analysis of the difference image for unsupervised change detection", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 3, pp. 1171–1182, May 2000. DOI: 10.1109/36.843009.

[24] B. Storvik, G. Storvik, and R. Fjortoft, "On the Combination of Multisensor Data Using Meta-Gaussian Distributions", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 7, pp. 2372–2379, Jul. 2009. DOI: 10.1109/TGRS.2009.2012699.

[25] Z.-g. Liu, J. Dezert, G. Mercier, Q. Pan, and Y.-m. Cheng, "Change detection from remote sensing images based on evidential reasoning", in *14th International Conference on Information Fusion*, Jul. 2011, pp. 1–8.

[26] Z.-g. Liu, J. Dezert, G. Mercier, and Q. Pan, "Dynamic evidential reasoning for change detection in remote sensing images", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 5, pp. 1955–1967, May 2012. DOI: 10.1109/TGRS.2011.2169075.

[27] G. Shafer, *A mathematical theory of evidence*. Princeton, NJ: Princeton Univ. Press, 1976, OCLC: 1859710, ISBN: 978-0-691-10042-5.

[28] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan. 1979. DOI: 10.1109/TSMC.1979.4310076.

[29] J. Kittler and J. Illingworth, "Minimum error thresholding", *Pattern Recognition*, vol. 19, no. 1, pp. 41–47, Jan. 1986. DOI: 10.1016/0031-3203(86)90030-0.

[30] G. Moser and S. Serpico, "Generalized minimum-error thresholding for unsupervised change detection from SAR amplitude imagery", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 10, pp. 2972–2982, Oct. 2006. DOI: 10.1109/TGRS.2006.876288.

[31] J. M. Prewitt and M. L. Mendelsohn, "The analysis of cell images", *Annals of the New York Academy of Sciences*, vol. 128, no. 3, pp. 1035–1053, Jan. 1966. DOI: 10.1111/j.1749-6632.1965.tb11715.x.

[32] G. W. Zack, W. E. Rogers, and S. A. Latt, "Automatic measurement of sister chromatid exchange frequency", *The Journal of Histochemistry and Cytochemistry: Official Journal of the Histochemistry Society*, vol. 25, no. 7, pp. 741–753, Jul. 1977. DOI: 10.1177/25.7.70454.

[33] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram", *Computer Vision, Graphics, and Image Processing*, vol. 29, no. 3, pp. 273–285, Mar. 1985. DOI: 10.1016/0734-189X(85)90125-2.

[34] A. G. Shanbhag, "Utilization of Information Measure as a Means of Image Thresholding", *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 5, pp. 414–419, Sep. 1994. DOI: 10.1006/cgip.1994.1037.

[35] J. C. Yen, F. J. Chang, and S. Chang, "A new criterion for automatic multilevel thresholding", *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, vol. 4, no. 3, pp. 370–378, 1995. DOI: 10.1109/83.366472.

[36] F. Melgani and Y. Bazi, "Robust Unsupervised Change Detection with Markov Random Fields", in *2006 IEEE International Symposium on Geoscience and Remote Sensing*, Jul. 2006, pp. 208–211. DOI: 10.1109/IGARSS.2006.58.

[37] S. Kullback and R. A. Leibler, "On Information and Sufficiency", *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, Mar. 1951. DOI: 10.1214/aoms/1177729694.

[38] J. Inglada and G. Mercier, "A New Statistical Similarity Measure for Change Detection in Multitemporal SAR Images and Its Extension to Multiscale Change Analysis", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, pp. 1432–1445, May 2007. DOI: 10.1109/TGRS.2007.893568.

[39] Z. Liu, G. Li, G. Mercier, Y. He, and Q. Pan, "Change Detection in Heterogenous Remote Sensing Images via Homogeneous Pixel Transformation", *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1822–1834, Apr. 2018. DOI: 10.1109/TIP.2017.2784560.

[40] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm", *Computers & Geosciences*, vol. 10, no. 2, pp. 191–203, Jan. 1984. DOI: 10.1016/0098-3004(84)90020-7.

[41] F. Smarandache and J. Dezert, *Advances and Applications of DSmT for Information Fusion (Collected works)*. Rehoboth: American Research Press, 2006, vol. 3, ISBN: 978-1-59973-073-8.

[42] G. Camps-Valls, L. Gomez-Chova, J. Munoz-Mari, J. L. Rojo-Alvarez, and M. Martinez-Ramon, "Kernel-Based Framework for Multitemporal and Multisource Remote Sensing Data Classification and Change Detection", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 6, pp. 1822–1835, Jun. 2008. DOI: 10.1109/TGRS.2008.916201.

[43] S. Krinidis and V. Chatzis, "A Robust Fuzzy Local Information C-Means Clustering Algorithm", *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1328–1337, May 2010. DOI: 10.1109/TIP.2010.2040763.

[44] Z.-g. Liu, J. Dezert, Q. Pan, and Y.-m. Cheng, "A new evidential c-means clustering method", in *2012 15th International Conference on Information Fusion*, Jul. 2012, pp. 239–246.

[45] K.-S. Chuang, H.-L. Tzeng, S. Chen, J. Wu, and T.-J. Chen, "Fuzzy c-means clustering with spatial information for image segmentation", *Computerized Medical Imaging and Graphics*, vol. 30, no. 1, pp. 9–15, Jan. 2006. DOI: 10.1016/j.compmedimag.2005.10.001.

[46] J. Prendes, M. Chabert, F. Pascal, A. Giros, and J.-Y. Tourneret, "A Bayesian Nonparametric Model Coupled with a Markov Random Field for Change Detection in Heterogeneous Remote Sensing Images", *SIAM Journal on Imaging Sciences*, vol. 9, no. 4, pp. 1889–1921, Jan. 2016. DOI: 10.1137/15M1047908.

[47] D. Tuia, J. Munoz-Mari, L. Gomez-Chova, and J. Malo, "Graph Matching for Adaptation in Remote Sensing", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 1, pp. 329–341, Jan. 2013. DOI: 10.1109/TGRS.2012.2200045.

[48] M. Volpi, G. Camps-Valls, and D. Tuia, "Spectral alignment of multitemporal cross-sensor images with automated kernel canonical correlation analysis", *ISPRS Journal of Photogrammetry and Remote Sensing*, Multitemporal remote sensing data analysis, vol. 107, pp. 50–63, Sep. 2015. DOI: 10.1016/j.isprsjprs.2015.02.005.

[49] R. Touati and M. Mignotte, "An Energy-Based Model Encoding Nonlocal Pairwise Pixel Interactions for Multisensor Change Detection", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 1046–1058, Feb. 2018. DOI: 10.1109/TGRS.2017.2758359.

[50] J. B. Kruskal and M. Wish, *Multidimensional scaling*, Nachdr., ser. Sage university papers Quantitative applications in the social sciences 11. Newbury Park, Calif.: Sage Publ, 2009, ISBN: 978-0-8039-0940-3.

[51] R. Touati, M. Mignotte, and M. Dahmane, "Change Detection in Heterogeneous Remote Sensing Images Based on an Imaging Modality-Invariant MDS Representation", in *2018 25th IEEE International Conference on Image Processing (ICIP)*, Oct. 2018, pp. 3998–4002. DOI: 10.1109/ICIP.2018.8451184.

[52] D. Shapira, S. Avidan, and Y. Hel-Or, "Multiple histogram matching", in *2013 IEEE International Conference on Image Processing*, Sep. 2013, pp. 2269–2273. DOI: 10.1109/ICIP.2013.6738468.

[53] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks", in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6645–6649. DOI: 10.1109/ICASSP.2013.6638947.

[54] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets", in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014, pp. 2672–2680.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification", in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15, Washington, DC, USA: IEEE Computer Society, 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123.

[56] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. DOI: 10.1038/nature14539.

[57] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, ser. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016, ISBN: 978-0-262-03561-3.

[58] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks", *Nature*, vol. 542, no. 7639, pp. 115–118, Feb. 2017. DOI: 10.1038/nature21056.

[59] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge", *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017. DOI: 10.1038/nature24270.

[60] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners", *OpenAI Blog*, vol. 1, no. 8, pp. 1–24, Feb. 2019.

[61] G. Chen, G. J. Hay, L. M. T. Carvalho, and M. A. Wulder, "Object-based change detection", *International Journal of Remote Sensing*, vol. 33, no. 14, pp. 4434–4457, Jul. 2012. DOI: 10.1080/01431161.2011.648285.

[62] T. Zhan, M. Gong, J. Liu, and P. Zhang, "Iterative feature mapping network for detecting multiple changes in multi-source remote sensing images", *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 146, pp. 38–51, Dec. 2018. DOI: 10.1016/j.isprsjprs.2018.09.002.

[63] F. Bovolo, S. Marchesi, and L. Bruzzone, "A Framework for Automatic and Unsupervised Detection of Multiple Changes in Multitemporal Images", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 6, pp. 2196–2212, Jun. 2012. DOI: 10.1109/TGRS.2011.2171493.

[64] L. Su, M. Gong, P. Zhang, M. Zhang, J. Liu, and H. Yang, "Deep learning and mapping based ternary change detection for information unbalanced images", *Pattern Recognition*, vol. 66, pp. 213–228, Jun. 2017. DOI: 10.1016/j.patcog.2017.01.002.

[65] Y. Bazi, L. Bruzzone, and F. Melgani, "Automatic identification of the number and values of decision thresholds in the log-ratio image for change detection in SAR images", *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 3, pp. 349–353, Jul. 2006. DOI: 10.1109/LGRS.2006.869973.

[66] F. Bovolo and L. Bruzzone, "A Theoretical Framework for Unsupervised Change Detection Based on Change Vector Analysis in the Polar Domain", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 1, pp. 218–236, Jan. 2007. DOI: 10.1109/TGRS.2006.885408.

[67] T. Zhan, M. Gong, X. Jiang, and S. Li, "Log-Based Transformation Feature Learning for Change Detection in Heterogeneous Images", *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 9, pp. 1352–1356, Sep. 2018. DOI: 10.1109/LGRS.2018.2843385.

[68] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96, AAAI Press, 1996, pp. 226–231.

[69] W. Zhao, Z. Wang, M. Gong, and J. Liu, "Discriminative Feature Learning for Unsupervised Change Detection in Heterogeneous Images Based on a Coupled Neural Network", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 12, pp. 7066–7080, Dec. 2017. DOI: 10.1109/TGRS.2017.2739800.

[70] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes", *arXiv:1312.6114 [cs, stat]*, May 2014.

[71] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models", *arXiv:1401.4082 [cs, stat]*, May 2014.

[72] N. Merkle, S. Auer, R. Müller, and P. Reinartz, "Exploring the Potential of Conditional Adversarial Networks for Optical and SAR Image Matching", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 6, pp. 1811–1820, Jun. 2018. DOI: 10.1109/JSTARS.2018.2803212.

[73] J. Fagir, M. Frioud, and D. Henke, "Change Detection between High-Resolution Airborne SAR and Multispectral Data with Dempster-Shafer Theory", in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, Jul. 2019, pp. 1526–1529. DOI: 10.1109/IGARSS.2019.8900637.

[74] X. X. Zhu, D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources", *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, Dec. 2017. DOI: 10.1109/MGRS.2017.2762307.

[75] S. Liu, S. De Mello, J. Gu, G. Zhong, M.-H. Yang, and J. Kautz, "Learning Affinity via Spatial Propagation Networks", in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 1520–1530.

[76] F. R. K. Chung, *Spectral graph theory*, ser. Regional conference series in mathematics no. 92. Providence, R.I: Published for the Conference Board of the mathematical sciences by the American Mathematical Society, 1997, ISBN: 978-0-8218-0315-8.

[77] U. von Luxburg, "A Tutorial on Spectral Clustering", *arXiv:0711.0189 [cs]*, Nov. 2007.

[78] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*, ser. Adaptive computation and machine learning. Cambridge, MA: MIT Press, 2009, ISBN: 978-0-262-01319-2.

[79] X. Cheng, P. Wang, and R. Yang, "Depth Estimation via Affinity Learned with Convolutional Spatial Propagation Network", *arXiv:1808.00150 [cs]*, Jul. 2018.

[80] M. Maire, T. Narihira, and S. X. Yu, "Affinity CNN: Learning Pixel-Centric Pairwise Relations for Figure/Ground Embedding", *arXiv:1512.02767 [cs]*, Dec. 2015.

[81] H. Rue and L. Held, *Gaussian Markov random fields: theory and applications*, ser. Monographs on statistics and applied probability 104. Boca Raton: Chapman & Hall/CRC, 2005, ISBN: 978-1-58488-432-3.

[82] R. J. Trudeau and R. J. Trudeau, *Introduction to graph theory*, ser. Dover books on advanced mathematics. New York: Dover Pub, 1993, ISBN: 978-0-486-67870-2.

[83] J. Shi and J. Malik, "Normalized cuts and image segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug. 2000. DOI: 10.1109/34.868688.

[84] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989. DOI: 10.1016/0893-6080(89)90020-8.

[85] G. Cybenko, "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989. DOI: 10.1007/BF02551274.

[86] A. Kratsios, "Universal Approximation Theorems", *arXiv:1910.03344 [cs, math, stat]*, Oct. 2019.

[87] A. S. Strauman, "Segmentation and Unsupervised Adversarial Domain Adaptation Between Medical Imaging Modalities", Master's thesis, Jul. 2019.

[88] D. J. Trosten, "Deep Image Clustering with Tensor Kernels and Unsupervised Companion Objectives", Master's thesis, May 2019.

[89] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986. DOI: 10.1038/323533a0.

[90] H. Robbins and S. Monro, "A Stochastic Approximation Method", *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, Sep. 1951. DOI: 10.1214/aoms/1177729586.

[91] J. Kiefer and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function", *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, Sep. 1952. DOI: 10.1214/aoms/1177729392.

[92] C. Darken and J. E. Moody, "Note on Learning Rate Schedules for Stochastic Optimization", in *Advances in Neural Information Processing Systems 3*, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds., Morgan-Kaufmann, 1991, pp. 832–838.

[93] S. Ruder, "An overview of gradient descent optimization algorithms", *arXiv:1609.04747 [cs]*, Jun. 2017.

[94] N. Qian, "On the momentum term in gradient descent learning algorithms", *Neural Networks*, vol. 12, no. 1, pp. 145–151, Jan. 1999. DOI: 10.1016/S0893-6080(98)00116-6.

[95] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate O(1/k2)", *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543–547, 1983.

[96] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization", *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[97] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method", *arXiv:1212.5701 [cs]*, Dec. 2012.

[98] G. Hinton, N. Srivastava, and K. Swersky, *Neural networks for machine learning lecture 6a overview of mini-batch gradient descent*, 2012.

[99] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", *arXiv:1412.6980 [cs]*, Jan. 2017.

[100] T. Dozat, "Incorporating Nesterov Momentum into Adam", 2016.

[101] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive Gradient Methods with Dynamic Bound of Learning Rate", *arXiv:1902.09843 [cs, stat]*, Feb. 2019.

[102] J. Larsen and L. Hansen, "Generalization performance of regularized neural network models", in *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, Sep. 1994, pp. 42–51. DOI: 10.1109/NNSP.1994.366065.

[103] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", *arXiv:1207.0580 [cs]*, Jul. 2012.

[104] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: When to warp?", *arXiv:1609.08764 [cs]*, Nov. 2016.

[105] J. Ding, B. Chen, H. Liu, and M. Huang, "Convolutional Neural Network With Data Augmentation for SAR Target Recognition", *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 3, pp. 364–368, Mar. 2016. DOI: 10.1109/LGRS.2015.2513754.

[106] L. Perez and J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning", *arXiv:1712.04621 [cs]*, Dec. 2017.

[107] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", *arXiv:1502.03167 [cs]*, Mar. 2015.

[108] Zhou and Chellappa, "Computation of optical flow using a neural network", in *IEEE 1988 International Conference on Neural Networks*, Jul. 1988, 71–78 vol.2. DOI: 10.1109/ICNN.1988.23914.

[109] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion", *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.

[110] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least Squares Generative Adversarial Networks", in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2813–2821. DOI: 10.1109/ICCV.2017.304.

[111] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Networks", in *International Conference on Machine Learning*, Jul. 2017, pp. 214–223.

[112] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 5967–5976. DOI: 10.1109/CVPR.2017.632.

[113] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image Analogies", in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '01, New York, NY, USA: ACM, 2001, pp. 327–340. DOI: 10.1145/383259.383295.

[114] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks", in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244.

[115] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", *arXiv:1703.10593 [cs]*, Nov. 2018.

[116] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[117] D. Powers, "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation", *Mach. Learn. Technol.*, vol. 2, Jan. 2008.

[118] R. Delgado and X.-A. Tibau, "Why Cohen's Kappa should be avoided as performance measure in classification", *PLOS ONE*, vol. 14, no. 9, e0222916, Sep. 2019. DOI: 10.1371/journal.pone.0222916.

[119] J. Cohen, "A Coefficient of Agreement for Nominal Scales", *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, Apr. 1960. DOI: 10.1177/001316446002000104.

[120] M. L. McHugh, "Interrater reliability: The kappa statistic", *Biochemia medica : Biochemia medica*, vol. 22, no. 3, pp. 276–282, Oct. 2012.

[121] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme", *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, Oct. 1975. DOI: 10.1016/0005-2795(75)90109-9.

[122] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks", *arXiv:1311.2901 [cs]*, Nov. 2013.

[123] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, "Unmasking Clever Hans predictors and assessing what machines really learn", *Nature Communications*, vol. 10, no. 1, pp. 1–8, Mar. 2019. DOI: 10.1038/s41467-019-08987-4.

[124] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, Eds., *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, vol. 11700, ISBN: 978-3-030-28953-9. DOI: 10.1007/978-3-030-28954-6.

[125] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems", 2015.

[126] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Mar. 2010, pp. 249–256.

[127] P. Krähenbühl and V. Koltun, "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials", in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2011, pp. 109–117.

[128] R. Meyes, M. Lu, C. W. de Puiseau, and T. Meisen, "Ablation Studies in Artificial Neural Networks", *arXiv:1901.08644 [cs, q-bio]*, Feb. 2019.

[129] *What is an ablation study?* [Online]. Available: https://stats.stackexchange.com/questions/380040/what-is-an-ablation-study-and-is-there-a-systematic-way-to-perform-it (visited on 12/14/2019).

[130] W. H. Jefferys and J. O. Berger, "Ockham's Razor and Bayesian Analysis", *American Scientist*, vol. 80, no. 1, pp. 64–72, 1992.

[131] M. Schmitt, L. H. Hughes, and X. X. Zhu, "The SEN1-2 Dataset for Deep Learning in SAR-Optical Data Fusion", *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1, pp. 141–146, Sep. 2018. DOI: 10.5194/isprs-annals-IV-1-141-2018.