From satellite images to vector maps

Onur Tasar

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Des Images Satellites aux Cartes Vectorielles

From Satellite Images to Vector Maps

Onur Tasar

Inria Sophia Antipolis – Méditerranée

Présentée en vue de l'obtention du grade de docteur en
Automatique, Traitement du Signal et des Images
d'Université Côte d'Azur

Dirigée par : Pierre Alliez
Co-encadrée par : Yuliya Tarabalka

Soutenue le : 22 Septembre 2020

Devant le jury, composé de :
Rapporteurs
Xiaoxiang Zhu, Professeur, TUM / DLR
Bertrand Le Saux, Scientifique Senior, ESA
Examinateurs
Devis Tuia, Professeur Agrégé, EPFL
Xavier Descombes, Directeur de Recherche, INRIA
Invités
Alain Giros, Chercheur, CNES
Sébastien Clerc, Directeur de Département, ACRI-ST

Inria

cnes CENTRE NATIONAL D'ÉTUDES SPATIALES

ACRI ST

Des Images Satellites aux Cartes Vectorielles

From Satellite Images to Vector Maps

Jury

Président du jury

- Xavier Descombes, Directeur de Recherche, Institut National de Recherche en Sciences et Technologies du Numérique (INRIA), Sophia Antipolis, France

Rapporteurs

- Xiaoxiang Zhu, Professeur, Technical University of Munich (TUM) / Deutsches Zentrum für Luft- und Raumfahrt (DLR), Munich, Allemagne
- Bertrand Le Saux, Scientifique Senior, European Space Agency (ESA), Rome, Italie

Examinateurs

- Devis Tuia, Professeur Agrégé, École polytechnique fédérale de Lausanne (EPFL), Lausanne, Suisse
- Xavier Descombes, Directeur de Recherche, Institut National de Recherche en Sciences et Technologies du Numérique (INRIA), Sophia Antipolis, France

Invités

- Alain Giros, Chercheur, Centre National d'Études Spatiales (CNES), Toulouse, France
- Sébastien Clerc, Directeur de Département, ACRI-ST, Sophia Antipolis, France

Résumé

Grâce à d'importants développements technologiques au fil des ans, il a été possible de collecter des quantités massives de données de télédétection. Par exemple, les constellations de divers satellites sont capables de capturer de grandes quantités d'images de télédétection à haute résolution spatiale ainsi que de riches informations spectrales sur tout le globe. La disponibilité de données aussi gigantesques a ouvert la porte à de nombreuses applications et a soulevé de nombreux défis scientifiques. Parmi ces défis, la génération automatique de cartes précises est devenue l'un des problèmes les plus intéressants et les plus anciens, car il s'agit d'un processus crucial pour un large éventail d'applications dans des domaines tels que la surveillance et l'aménagement urbains, l'agriculture de précision, la conduite autonome et la navigation.

Cette thèse vise à développer de nouvelles approches pour générer des cartes vectorielles à partir d'images de télédétection. À cette fin, nous avons divisé la tâche en deux sous-étapes. La première étape consiste à générer des cartes matricielles à partir d'images de télédétection en effectuant une classification au niveau des pixels grâce à des techniques avancées d'apprentissage profond. La seconde étape vise à convertir les cartes matricielles en cartes vectorielles en utilisant des structures de données et des algorithmes de géométrie algorithmique. Cette thèse aborde les défis qui sont couramment rencontrés au cours de ces deux étapes.

Bien que des recherches antérieures aient montré que les réseaux neuronaux convolutifs (CNN) sont capables de générer d'excellentes cartes lorsque les données d'entraînement sont représentatives des données d'essai, leurs performances diminuent considérablement lorsqu'il existe une grande différence de distribution entre les images d'entraînement et d'essai. Dans la première étape de notre traitement, nous visons principalement à surmonter les capacités de généralisation limitées des CNN pour effectuer une classification à grande échelle. Nous explorons également un moyen d'exploiter de multiples ensembles de données collectées à différentes époques avec des annotations pour des classes distinctes afin de former des CNN capables de générer des cartes pour toutes les classes.

Dans la deuxième partie, nous décrivons une méthode qui vectorise les cartes matricielles pour les intégrer dans des applications de systèmes d'information géographique, ce qui complète notre chaîne de traitement. Tout au long de cette thèse, nous expérimentons sur un grand nombre d'images satellitaires et aériennes de très haute résolution. Nos expériences démontrent la robustesse et la capacité à généraliser des méthodes proposées.

Mots Clés: Apprentissage profond, Segmentation sémantique, Images satellites, Adaptation de domaine, Apprentissage incrémental, Vectorisation d'images

Abstract

With the help of significant technological developments over the years, it has been possible to collect massive amounts of remote sensing data. For example, the constellations of various satellites are able to capture large amounts of remote sensing images with high spatial resolution as well as rich spectral information over the globe. The availability of such huge volume of data has opened the door to numerous applications and raised many challenges. Among these challenges, automatically generating accurate maps has become one of the most interesting and long-standing problems, since it is a crucial process for a wide range of applications in domains such as urban monitoring and management, precise agriculture, autonomous driving, and navigation.

This thesis seeks for developing novel approaches to generate vector maps from remote sensing images. To this end, we split the task into two sub-stages. The former stage consists in generating raster maps from remote sensing images by performing pixel-wise classification using advanced deep learning techniques. The latter stage aims at converting raster maps to vector ones by leveraging computational geometry approaches. This thesis addresses the challenges that are commonly encountered within both stages.

Although previous research has shown that convolutional neural networks (CNNs) are able to generate excellent maps when training data are representative for test data, their performance significantly drops when there exists a large distribution difference between training and test images. In the first stage of our pipeline, we mainly aim at overcoming limited generalization abilities of CNNs to perform large-scale classification. We also explore a way of leveraging multiple data sets collected at different times with annotations for separate classes to train CNNs that can generate maps for all the classes.

In the second part, we propose a method that vectorizes raster maps to integrate them into geographic information systems applications, which completes our processing pipeline. Throughout this thesis, we experiment on a large number of very high resolution satellite and aerial images. Our experiments demonstrate robustness and scalability of the proposed methods.

Keywords: Deep learning, Semantic segmentation, Satellite images, Domain adaptation, Incremental learning, Image vectorization

To my beloved father Süleyman TAŞAR and my mother Fadime TAŞAR.

Contents

Chapter 1

Introduction

1.1 Context and Motivations

Over the years, the continuous proliferation and the significant improvements of satellite sensors enabled to collect huge volume of remote sensing images from a significant part of the Earth's surface with high spatial and temporal resolution, as well as rich spectral information. Such massive data contain crucial information, which has opened the door to a wide range of applications such as monitoring natural disasters, urban planning, autonomous driving, navigation, and precise agriculture. However, a substantial portion of these massive data is still unused. In order to efficiently process such data for many real-world applications, it is of paramount importance to devise efficient representations for these images.

For the aforementioned representations, it is of great interest to develop novel methods that can generate vector maps containing geometric structures that precisely delineate contours of the objects. It is also necessary to enrich such geometric structures with semantic information to query which semantic class each structure belongs to. Considering that remote sensing images usually cover a large geographic extent and contain a large number of pixels, it would be unfeasible to generate maps manually. Hence, this process needs to be automatized.

Over the last couple of decades, semantic segmentation or dense labeling of remote sensing images have gained a great interest for the purpose of such automation. This task consists in assigning a semantic label to each pixel in the image. Especially after convolutional neural networks (CNNs) have been invented and became popular, in many benchmarks such as INRIA [111], ISPRS [85], SpaceNet [177], it has been proven that precise raster maps can be automatically generated. Although CNNs achieve an excellent segmentation performance when training and test data have similar data distributions, it has been shown that they have limited generalization abilities [175]. Therefore, we must develop methods that will overcome this issue to generalize CNNs to unseen geographic locations. Another problem is that it is not feasible for them to deal with training data obtained at different times and having annotations for separate classes. An ideal approach must be capable of learning from such heterogeneous training data

and generating maps for all the classes that it has learned over time.

Vectorization of raster maps has also been gaining a growing attention, since it is a crucial step for extracting an efficient vector representation. For this task, implementations of many polygon generalization algorithms are already available in various geographic information systems (GIS) software. However, we seek for a compact yet faithful representation with a good complexity/fidelity trade-off for fast querying and easy processing. We also wish the vectorization algorithm to fix the potential errors in the raster maps introduced by the learning approaches.

Given these recent advances, one can consider splitting the task of generating efficient representations into two sub-tasks. The first task consists in generating raster maps from remote sensing images using advanced machine learning techniques, and the second task aims at vectorizing the predicted raster maps using computational geometry methodologies. Both tasks have their own challenges that need to be solved.

1.2 Remote Sensing Images

For the purpose of this thesis, remote sensing can be defined as the measurement of object properties on the Earth's surface using data acquired from aircraft and satellites [153]. In the following sub-sections, we summarize the remote sensing image types that are used in this thesis.

1.2.1 Aerial Images

The interests in very high resolution remote sensing images are receiving a growing attention. Especially for the automatic cartography problem, very high resolution images are particularly useful, because they facilitate generating maps with high precision. Aerial images are collected by cameras mounted on aircraft, helicopters, unmanned aerial vehicles (UAVs), balloons, etc. Using these cameras, oblique or nadir images can be acquired. In oblique imagery, photographs are taken at certain angles relative to the normal of the Earth's surface. Nadir images correspond to orthophotos taken from vertically above. The oblique images are relevant for problems like 3-D reconstruction [87] and height estimation [131]. On the other hand, nadir images are useful for cartography.

For research purposes, there are public data sets consisting of aerial images such as INRIA [111], Potsdam and Vaihingen benchmarks [85]. However, there are fewer sources for aerial images than for satellite images. The main reason is that the flying devices need to be used every time when new images are intended to be collected. In addition, such devices have some restrictions in certain countries.

1.2.2 Satellite Images

Satellite images are the data acquired over the Earth's surface via satellites operated by governments and various businesses around the world. The recent significant technological improvements in satellite sensors have allowed us to collect massive volume of satellite images covering the whole world. The abundance of satellite images makes them

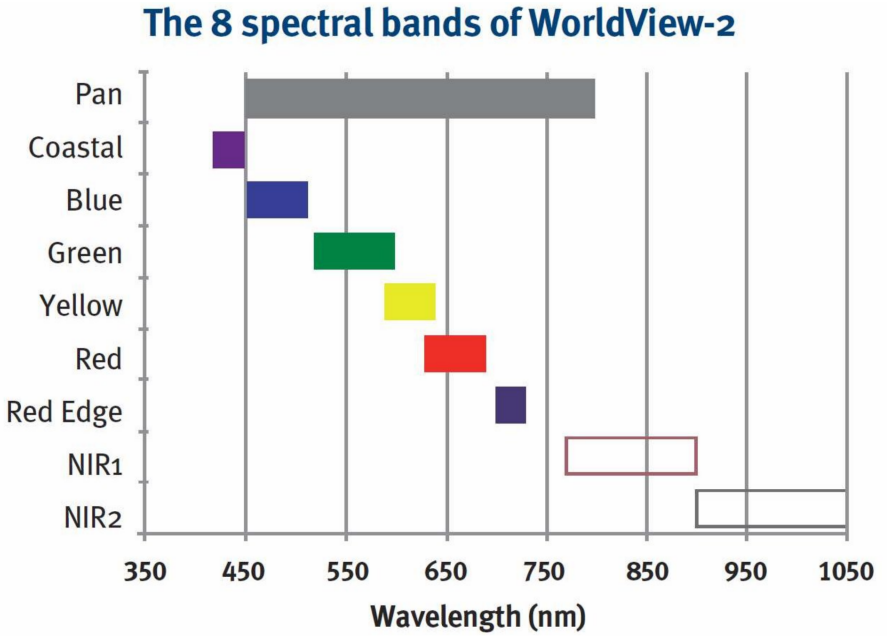


Figure 1.1: Wavelengths of the spectral bands of WorldView-2 satellite. The figure is taken from [132].

one of the most commonly used data source for Earth observation. Compared to aerial images, there is a significantly larger number of satellite images accessible. For instance, some public benchmarks such as SpaceNet [177] and DeepGlobe [45] are example public data sources of satellite images, and they cover relatively large geographic extents.

The images generated by Landsat, Sentinel, Pléiades, and Worldview missions are examples of satellite images. Landsat 8 mission was launched in 2013. The satellite images captured by this sensor consist of 11 bands, in which their resolutions range between 15 and 30 meters. Sentinel-2A and Sentinel-2B missions were launched in 2015 and 2017, respectively. They collect MS data with 13 bands. The spatial resolutions of the bands are 10 m, 20 m, or 30 m. Since the medium resolution Sentinel-2 images are publicly accessible, they are commonly used for land cover classification [83]. Pléiades and Worldview are able to acquire both panchromatic and MS images. The panchromatic data correspond to a high resolution single band image. For instance, spatial resolutions of the panchromatic images captured by Pléiades and Worldview are 0.5 m and 0.31 m, respectively. On the other hand, MS images usually comprise red, green, blue, and near-infrared bands with four times lower resolution.

One disadvantage of the satellite images is that due the technical limitations in satellite sensors, there must be a tradeoff between the spatial and the spectral resolutions [122, 154]. Instead of capturing high resolution multi-spectral images, satellite sensors are usually capable of collecting high resolution panchromatic (PAN) images and lower resolution multi-spectral (MS) images concurrently. This problem originates from

Figure 1.2: Pan-sharpening. (a) Multi-spectral image, (b) panchromatic image, (c) pan-sharpened image.

two reasons. The first one is the radiation energy collected by the sensor. As depicted in Fig. 1.1, PAN images have broader and MS images have narrower bandwidths. In order to collect more photons, the size of the MS detector must be larger [122]. The second issue is the size of the acquired data. Since high resolution MS images occupy significantly larger space that PAN and MS bundle, it is easier to collect the images as a PAN and MS bundle.

In the context of semantic segmentation, both PAN and MS images carry important information. To benefit from the advantages of both types of data, it is a common practice to merge them so that high resolution MS images can be obtained. This operation is referred to as image pan-sharpening [6, 7, 89]. The pan-sharpening process is depicted in Fig. 1.2. Effective pan-sharpening is a research topic in itself.

The second disadvantage of satellite data is that in general satellite databases are huge and require significant image processing to create useful images from the raw data. The third issue is that various weather conditions and different sensor characteristics make the processing of satellite images challenging. For example, due to the atmospheric conditions, the images depicted in Figs. 1.3a and 1.3b have substantially different color distributions. In addition, depending on the data acquisition time, the geographic location from which data are captured, and the pre-processing technique, collected satellite images might be cloudy, hazy, or noisy as respectively shown in Figs. 1.3c, 1.3d, and 1.3e. Besides, depending on the satellite sensor, taken satellite images may consist of different types of spectral bands. For example, the image in Fig. 1.3f comprises near infrared, red, and green bands whereas the other images in Fig. 1.3 consist of red, green, and blue bands. In the context of dense labeling problem, the above-mentioned problems prevent machine learning models from generating precise maps.

This thesis mainly focuses on satellite images, especially the data collected by Pléiades satellite.

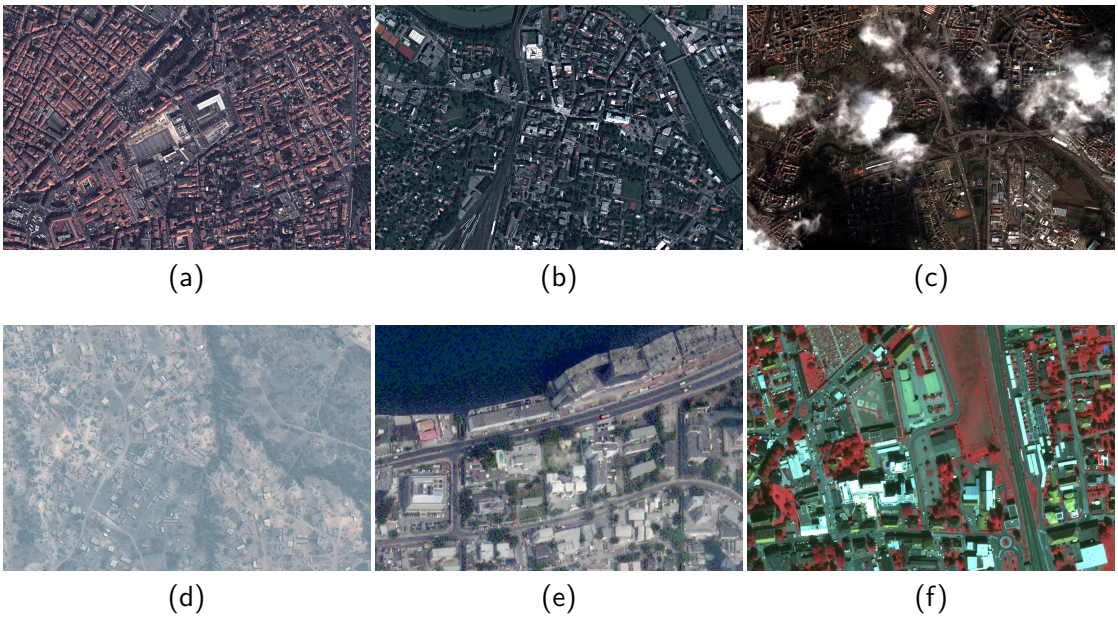


Figure 1.3: Challenges in satellite images. (a-b) Images affected by various atmospheric conditions, and (c) cloudy, (d) hazy, (e) noisy, (f) NIRRG images.

1.3 Overview of the Supervised Learning Approaches for Semantic Segmentation of Remote Sensing Images

Semantic segmentation, pixel-wise classification, or dense labeling of remote sensing images is a long-standing problem, since it is a crucial process for a wide range of applications such as urban planning, mapping, navigation, and autonomous driving. In supervised semantic segmentation setting, a classifier is trained on labeled training images and used to segment unlabeled test images. A large part of this thesis is dedicated to semantic segmentation of satellite images using advanced machine learning techniques, which constitutes the first stage of our overall pipeline. In this section, we review the supervised learning based approaches for semantic segmentation of remote sensing images.

1.3.1 Traditional Learning Based Approaches

Among the traditional learning based approaches, Support Vector Machine (SVM) [41], Random Forest (RF) [17], and Artificial Neural Networks (ANNs) [186] have been the most commonly used methods in remote sensing literature.

The SVM is a kernel based classifier, which maps data samples into a higher dimensional feature space. In the new feature space, the data samples belonging to two different classes are separated by a hyper-plane. The data points nearest to the hyper-plane are referred to as support vectors. To find the right hyper-plane, margin between

the support vectors is maximized. To perform multi-class classification, multiple hyperplanes are used instead of only one. The SVM historically has been one of the most widespread used segmentation approach for remote sensing images [121]. Its major limitation is that it performs pixel-wise classification without taking the spatial information into account [163]. As a consequence, the output segmentation is usually noisy. To overcome this limitation, certain regularization approaches have been introduced such as markov random field (MRF) [163], conditional random field (CRF) [70], and graph-cut based methods [164, 165]. Remote sensing literature describes the SVM based methods for other dense labeling problems as well, such as active learning [18] and domain adaptation [20].

The RF is an ensemble learning approach consisting of multiple decision trees. The decision tree is a rule-based and tree-like learning approach, where each node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node corresponds to a class label. In RF, the outputs of multiple decision trees are aggregated to predict class labels of the data samples. In remote sensing literature, the RF has been used for land cover [40], urban building [10] and impervious surface classification [47]. The biggest advantage of RF is that it does not require as much data as CNNs. It can achieve a relatively good performance with much less training data. Therefore, it has been extensively used for large-scale land cover mapping. For instance, Inglada *et al.* generated a land cover map consisting of 17 classes for the whole France by leveraging Sentinel time series images.

The ANNs are learning methods that mimic the way biological networks in the brain process information. ANNs consist of several layers, where each layer comprises a certain number of neurons. The neurons in adjacent layers are connected to each other, and each connection has a weight. The weights of an ANNs are used to activate each neuron. The first, the last, and the intermediate layers are referred to as input, output, and hidden layers respectively. The input and the output layers correspond to the input data samples and the predictions by the network. The hidden layers represent the features learned by the network in the training stage. The ANNs have two stages: forward propagation and backward propagation. Given some data samples, the ANNs output predictions in the forward propagation. In the backward propagation, weights of the ANNs are updated via certain optimizers such as gradient descent [43] or Adam [90] by computing and minimizing the mismatch between the ground-truth as well as the predictions by the ANNs. In the training stage, both forward and backward propagations are conducted, whereas only forward propagation is performed in the test stage. The biggest advantage of the ANNs is that they do not require feature engineering; they learn useful features themselves. Benediktsson *et al.* introduced the early stage usage of ANNs for remote sensing image classification [11, 12].

1.3.2 Deep Convolutional Neural Networks

Deep convolutional neural networks (CNNs) are network architectures comprising a stack of convolutional filters [94]. There is a vast literature on segmentation of remote sensing images using only spectral information of each pixel. As mentioned earlier, there are

alternative methods that aim to incorporate spatial information with spectral information to generate better maps. However, these approaches are not effective when it comes to generating accurate maps for large-scale images. On the other hand, CNNs are able to take the contextual information into account as well. Due to the recent significant developments in hardware, deeper and higher performing CNNs can be trained on the current GPUs in an end-to-end manner.

Initially, the CNNs have been designed for image categorization problem [38, 92]. These methods consist of multiple convolutional filters to extract the features followed by several fully connected layers for class prediction. In remote sensing, Mnih proposed a specific patch based CNN architecture to perform dense labeling [124]. In the architecture, the network takes an image as input, and outputs a predicted map with a smaller size centered in the input image. To perform dense labeling, the output size of the final fully connected layer is increased. However, due to its patch-based architecture, this approach generates inaccurate maps. Later on, Long and Shelhamer introduced fully convolutional neural networks for semantic segmentation [106]. Maggiori *et al.* extended this approach and applied their architecture to remote sensing images [110, 112].

Recently, U-net architecture [144] has gained a significant attention in the field of remote sensing. This network architecture consists of a contracting path that captures the context and a symmetric expanding path, enabling accurate localization. In addition to traditional encoder-decoder layers, the U-net architecture utilizes skip connections, which combine low level features with the high level ones in the expanding path to increase precision of localization. It has been shown that concatenating low level features with high level ones via skip connections allows to generate better segmentation, and the choice of loss function plays an important role [75]. The variants of U-net such as TernausNet [81], TernausNetV2 [80], D-LinkNet [197], and AlbuNet [159] have exhibited a great success in many remote sensing benchmarks.

The other popular network architectures are the variants of DeepLab (i.e., DeepLabv1 [31], DeepLabv2 [32], DeepLabv3 [33], and DeepLabv3+ [34]). Different from U-net and other CNNs, these architectures use atrous convolutions to enlarge the view of the filters. Although U-net is the most widespread used architecture for classes requiring high precision such as buildings and roads, variants of DeepLab are commonly used for land cover classification [93]. Another common architecture is Mask R-CNN [67], which tackles instance segmentation problem. It generates a bounding box around each object as well as the segmentation mask for the object inside each bounding box. In remote sensing, it has been used for building [195], ship [127], and airplane [161] segmentation.

Although in this thesis, we split the task of generating efficient vector representations into two stages, there is also an intense research activity aiming to develop CNNs that can generate such vector representations directly from satellite images. In this context PolygonRNN [27] and PolygonRNN++ [1] are the first attempts to output vector segmentation. However, they require human interaction. The end user needs to draw a bounding box around the object that would be segmented. Zuoyue *et al.* have recently extended these approaches to make them end-to-end trainable. They applied their solution to extract vector representations of building and road classes. The literature also

describes several deep learning approaches that learn parameters of the active contour models to perform building segmentation [66, 117].

1.3.3 Transition to Large-scale Classification

Until the last decade, among researchers, it has been a common practice to use some portion of a single remote sensing image as training and the rest as test data to evaluate the performance of their machine learning models. In remote sensing, there are public benchmarks that adopt such an experimental setting. For example, in the Vaihingen and the Potsdam benchmarks provided by International Society for Photogrammetry and Remote Sensing (ISPRS) [85], an aerial image taken from a city is split into several tiles. Some of the tiles are provided as training and the rest as test data. Particularly deep learning approaches have exhibited an impressive performance for semantic segmentation in such an experimental setup. However, the main limitation of such a setting is that training and test images have very similar data distributions. As a result, and even if the machine learning model overfits to the training data, it performs extremely well on test data.

However, especially for dense labeling problem, preparing an annotated training data set is extremely labor-intensive, since the contour of each object needs to be precisely delineated. As mentioned before, due to the various weather conditions, intra-class variability, and different sensor characteristics, objects in the images collected from different parts of world tend to greatly differ. As a consequence, it is challenging to have massive volume of annotated images that are representative for the images collected from different geographic locations. Hence, to perform large-scale semantic segmentation, it is crucial to develop novel learning approaches that offer high generalization abilities.

Our main focus in the first stage of our overall pipeline is to propose novel methods that can generate accurate maps even when objects in training and test images look significantly different.

1.4 Objective and Challenges

In this thesis, we seek for generating efficient vector representations for remote sensing images. To achieve this task, we follow a two step procedure. The former aims to assign a thematic label in the image using advanced deep learning methods, and the latter vectorizes the raster map obtained in the first step. This thesis addresses several challenges arising in both steps. They are as follows:

- **Automation:** It is the key for real-wold practical applications. Since we aim at dealing with massive volume of data, in both stages of our overall procedure, it is essential to propose hassle-free methods that do not require any human interaction.

- **Generalization:** As we will discuss in great detail in Chapters 3, 4, and 5, remote sensing images collected from different geographic locations at distinct times usually have largely different data distributions due to various reasons such as

atmospheric conditions, differences in acquisitions, etc. Such differences between images make machine learning models likely to fail to generate high quality maps. Therefore, we must devise efficient machine learning methods that are robust to such distribution differences.

- **Adaptability:** With the help of significant technological developments in remote sensing sensors and many satellite missions launched in the couple of decades, we are able to obtain huge amounts of annotated and unlabeled images every day. Hence, it is of great importance to develop advanced deep learning models that can efficiently and effectively adapt to continuously growing data.
- **Heterogeneous annotations:** Oftentimes, from different data set providers, we retrieve satellite images with annotations for separate classes. In order to maximize the information gain, an ideal machine learning model should be able to learn from data having such heterogeneous annotations. If this task can be accomplished, when new images are given, the model can generate maps for all the classes that it has learned over time.
- **Scalability:** As nowadays we have huge amounts of remote sensing data, we need to develop scalable solutions that can deal with a large number of images.
- **Representation Power:** The final vector representation must be efficient and effective. It must be powerful enough to precisely delineate contours of the Earth's objects but also effective enough to allow fast processing and easy querying. To this end, it is of paramount importance to generate vector maps with a good complexity/fidelity tradeoff.

1.5 Contributions

In the following, we describe the main contributions of each chapter. Notice that every chapter has its own problem definition, motivations, related work, methodology, and concluding remarks.

Chapter 2 We propose an incremental learning methodology enabling to learn segmenting new classes without hindering dense labeling abilities for the previous classes, although the entire previous data are not accessible. Our experimental results prove that it is possible to add new classes to the network, while maintaining its performance for the previous classes, despite the whole previous training data are not available.

Chapter 3 Although convolutional neural networks have been proven to be an effective tool to generate high quality maps from remote sensing images, their performance significantly deteriorates when there exists a large domain shift between training and test data. To address this issue, we propose two new data augmentation approaches that transfer the style of test data to training data using generative adversarial networks. Our semantic segmentation framework consists in first training

a classifier from the real training data and then fine-tuning it on the test stylized fake training data generated by the proposed approaches. Our experimental results prove that our data augmentation approaches enable classifiers to achieve a substantially better performance.

Chapter 4 The vast majority of the domain adaptation methods tackle single-source and single-target case, where the model trained on a single source domain is adapted to a target domain. However, these methods have limited practical real-world applications, since usually one has multiple source domains with different data distributions. In this chapter, we deal with multi-source domain adaptation problem. Our method standardizes each source and target domains so that all the data have similar data distributions. We then use the standardized source domains to train a classifier and segment the standardized target domain. Our experimental results show that the standardized data allow the classifiers to generate significantly better segmentation.

Chapter 5 We propose a novel approach for unsupervised, multi-source, multi-target, and life-long domain adaptation of satellite images. It consists of a classifier and a data augmentor. The data augmentor, which is a shallow network, is able to perform style transfer between multiple satellite images in an unsupervised manner, even when new data are added over time. In each training iteration, it provides the classifier with diversified data, which makes the classifier robust to large data distribution difference between the domains. Our extensive experiments prove that our approach significantly better generalizes to new geographic locations than the existing approaches.

Chapter 6 We propose a novel approach, which recasts the polygonization problem as a mesh-based approximation of the input classification map, where binary labels are assigned to the mesh triangles to represent the building class. A dense initial mesh is decimated and optimized using local edge and vertex-based operators in order to minimize an objective function that models a balance between fidelity to the classification map in ℓ_1 norm sense, right angle regularity for polygonized buildings, and final mesh complexity. Experiments show that our approach outperforms commonly used polygon generalization methods in remote sensing literature for similar numbers of vertices.

In the last chapter, we draw the final conclusions for this thesis and discuss the possible future directions.

1.6 Publications

Book Chapters

- B. Kellenberger, **O. Tasar**, B. B. Damodaran, N. Courty, D. Tuia, *"Deep Domain Adaptation in Earth Observation"*, Deep Learning for Earth Sciences, 2020

Refereed Journals

- **O. Tasar**, A. Giros, Y. Tarabalka, P. Alliez, S. Clerc, *"DAugNet: Unsupervised, Multi-source, Multi-target, and Life-long Domain Adaptation for Semantic Segmentation of Satellite Images"*, IEEE Transactions on Geoscience and Remote Sensing **(TGRS)**, July, 2020

- **O. Tasar**, S L Happy, Y. Tarabalka, P. Alliez, *"ColorMapGAN: Unsupervised Domain Adaptation for Semantic Segmentation Using Color Mapping Generative Adversarial Networks"*, IEEE Transactions on Geoscience and Remote Sensing **(TGRS)**, March, 2020

- **O. Tasar**, Y. Tarabalka, P. Alliez, *"Incremental Learning for Semantic Segmentation of Large-Scale Remote Sensing Data"*, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing **(JSTARS)**, September, 2019

- S. Lefévre, D. Sheeren, **O. Tasar**, *"A Generic Framework for Combining Multiple Segmentations in Geographic Object-Based Image Analysis"*, ISPRS International Journal of Geo-Information **(IJGI)**, January, 2019

Conference Proceedings

- **O. Tasar**, S. L. Happy, Y. Tarabalka, P. Alliez, *"SemI2I: Semantically Consistent Image-to-Image Translation for Domain Adaptation of Remote Sensing Data"*, IEEE International Geoscience and Remote Sensing Symposium **(IGARSS)**, Waikoloa, Hawaii, USA, September, 2020

- **O. Tasar**, Y. Tarabalka, A. Giros, P. Alliez, S. Clerc, *"StandardGAN: Multi-source Domain Adaptation for Semantic Segmentation of Very High Resolution Satellite Images by Data Standardization"*, Conference on Computer Vision and Pattern Recognition EarthVision Workshop **(CVPRW)**, Seattle, USA, June, 2020

- **O. Tasar**, Y. Tarabalka, P. Alliez, *"Continual Learning For Dense Labeling of Satellite Images"*, IEEE International Geoscience and Remote Sensing Symposium **(IGARSS)**, Yokohama, Japan, July, 2019

- A. Khalel, **O. Tasar**, G. Charpiat. Y. Tarabalka *"Multi-task Deep Learning For Satellite Image Pansharpening and Segmentation"*, IEEE International Geoscience and Remote Sensing Symposium **(IGARSS)**, Yokohama, Japan, July, 2019

- **O. Tasar**, E. Maggiori, P. Alliez, Y. Tarabalka, *"Polygonization of Binary Classification Maps Using Mesh Approximation with Right Angle Regularity"*, IEEE International Geoscience and Remote Sensing Symposium **(IGARSS)**, Valencia, Spain, July, 2018

Chapter 2

Progressively Learning to Segment New Classes

2.1 Problem Definition

In this chapter, we propose an incremental learning approach for progressively learning to segment new classes without forgetting the previous ones. The inputs for the proposed approach are multiple satellite images acquired at different times and having annotations for separate classes. Every time new training data are obtained, we use new data to learn to segment new classes and a small portion of the previous data in order for the approach not to forget the previous classes. The desired output is a precise raster map containing new as well as old classes.

2.2 Motivations

In the last decade, with the great advances in deep neural networks, notably convolutional neural networks (CNNs), it has been possible to obtain accurate segmentations [112]. Among the CNN-based approaches, U-net architecture [144] has gained a particular attention due to its success for various segmentation problems in different domains (e.g., medical imaging and remote sensing).

The major drawback of U-net or other CNNs is their assumption that the whole training data are available in the beginning, which is not the case in real-world remote sensing applications, as new images are collected from all over the world everyday. Besides, having large amounts of standard and unique label maps is almost impossible, because the label maps retrieved from different sources usually have distinct classes. In addition, it is not always possible to store massive volume of training data. For the reasons described above, designing an incremental learning methodology, which can learn from new training data while retaining performance for old classes without accessing to the entire previous training data is crucial. Although a good solution for this problem is needed to generate high-quality maps from satellite images that cover a large geographic

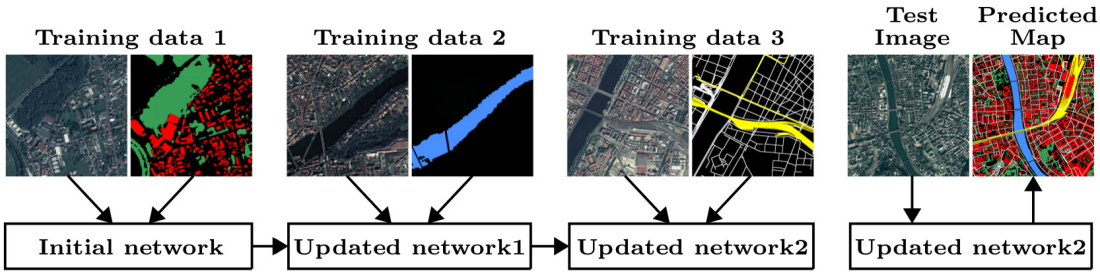


Figure 2.1: Incremental learning scenario. Firstly, satellite images as well as their label maps for building and high vegetation classes are fed to the network. Then, from the second training data, the network learns water class without forgetting building and high vegetation classes. Finally, road and railway classes are taught to the network. Whenever new training data are obtained, we store only a small part of the previous ones for the network to remember. When a new test image comes, the network is able to detect all the classes.

extent, it has remained unexplored in remote sensing community.

Rather than assuming that we initially have all the training data, we aim to design an incremental learning methodology. Let us explain with an example of a real-world problem (see Fig. 2.1) where, in the beginning, we are provided images from several cities in Austria with correspondent label maps for building and high vegetation classes. Later on, we are given other training data, having label maps for water class, collected from different areas in Germany. Finally, we receive new satellite images and their annotations for road and railway classes from certain cities in France. Every time new data arrive, we assume that only a small portion of the previous data is stored. In such a scenario, our goal is to add segmentation capabilities for new classes to the previously trained network, without forgetting the already learned information so that maps for all the learned classes could be generated by the network. In addition to the described problem, because labeling satellite images covering a large geographic area requires a lot of manual work, it is quite common that annotations of different classes for the same images are provided sequentially in time. In this kind of situation, it is not feasible to train a new classifier from scratch every time new label maps are obtained. The limitations pointed out in this section have motivated us to design an incremental learning methodology.

2.3 Related Work

The biggest challenge in incremental learning problem is that when new tasks are intended to be added to a classification system, performance of the system for the previously learned tasks degrades abruptly, which is referred to as "catastrophic forgetting" in the literature [54,62]. Incremental learning has been a historically important problem. Even before neural networks have become popular, researches had been studying this

issue [21, 29, 134, 172]. In the following sub-sections, we summarize the recent related work on incremental learning.

2.3.1 Enlarging the Network Architecture

These methods usually enlarge the network architecture to learn new tasks. One of the most commonly used approaches is progressively growing the network architecture by sharing early layers and splitting later ones by adding new convolutional kernels [151]. Such architectural growth can be horizontal [148,187] or hierarchical via a tree-structured model [181]. The problem of determining the number of filters to be added to each layer can be learned by reinforcement learning [183]. The major weakness of these approaches is that since the network grows during training, the number of parameters increases drastically as new tasks are added to the network.

2.3.2 Using a Small Portion of the Previous Data

To learn new tasks, these methodologies use not only the new training data but also a small portion of the old data [73,107,138]. These methods usually consist in learning new tasks from the current training data and remembering old tasks from a small fraction of the previous data. Which samples to keep from the previous training data can be determined by training a Support Vector Machine (SVM) [99]. The samples in the previous data that correspond to support vectors of the SVM can be stored to remember the former classes when the network is adapted to the new training data. Instead of using the old data directly, fake data resembling the previous data can be generated by generative adversarial networks (GANs) as well [158, 180].

2.3.3 Regularizing the Network

These methods mostly regularize the network by finding the important neurons for old tasks and preventing them from changing drastically to maintain the performance for the previous tasks. It has been proven that many configurations of the network parameters may produce the same result [162]. Inspired by this idea, several approaches, which try to find a configuration of the network parameters that well represents both the previous and the new training data have been published. Kirkpatrick *et al.* introduced an elastic weight consolidation (EWC) term, which is a multiplication of the importance values of the parameters for the old tasks and quadratic penalty on difference between the parameters of the previous and the updated networks [91]. The importance values of the parameters are measured by the estimated diagonal Fisher information matrix. The same work has been extended by rotating the Fisher matrix [105]. Another extension is combining the trained models for all the tasks via incremental moment matching (IMM) [97]. The elastic weight consolidation can be performed in online fashion as well [191]. Aljundi *et al.* determine the importance of each neuron by averaging gradients of the network outputs with respect to the parameters of the neurons [5]. Rannen *et al.* reconstruct features from the previous data using auto-encoders and use them

to preserve the information, which the old tasks rely upon [137]. Some methods aim at learning a mask marking the important neurons for old tasks [114,115]. They update only the masked out neurons when learning new tasks. Fernando *et al.* find paths through the network, which represent a subset of parameters by using tournament selection genetic algorithm [52]. In the training stage, only the neurons that are located along the paths are updated. Srivastava *et al.* proposed a method based on adaptive compression. There are also relevant contributions based on Bayesian inference [126, 142, 192].

2.3.4 Knowledge Distillation

In the context of neural networks, knowledge distillation means transferring the knowledge from a network or an assembly of several networks to a smaller network [69]. This strategy has inspired several approaches on incremental learning. It is quite common to facilitate a distillation loss in order to maintain performance for the previous tasks [28, 55, 101]. Shmelkov *et al.* proposed another knowledge distillation based approach that deals with incremental object detection and classification tasks at the same time.

2.4 Contributions

The contributions of this chapter are as follows:

- We propose an incremental learning methodology for semantic segmentation problem, where the network learns segmenting new classes without deteriorating performance for the previously learned classes, even when the entire previous training data are not stored. To the best of our knowledge, this is the first work which proposes a solution for the incremental semantic segmentation problem.
- We validate effectiveness of our approach on both satellite and aerial images. Our experiments address two common real-world problems, in which the former is the situation of retrieving stream of training data, where in each time step the data contain satellite images collected from different locations in the world and annotations for separate classes, the latter is the case where label maps for the same geographic area are provided sequentially.

2.5 Method

2.5.1 Network Architecture

Our network (see Fig. 2.2) is a variant of U-net, which consists of an encoder that is architecturally the same as the first 13 convolutional layers of VGG16 [160], a corresponding decoder, mapping low resolution encoder feature maps to original input image size of outputs, and two center convolutional layers. We prefer to use VGG16 as the encoder, because it provides a good compromise between complexity and performance,

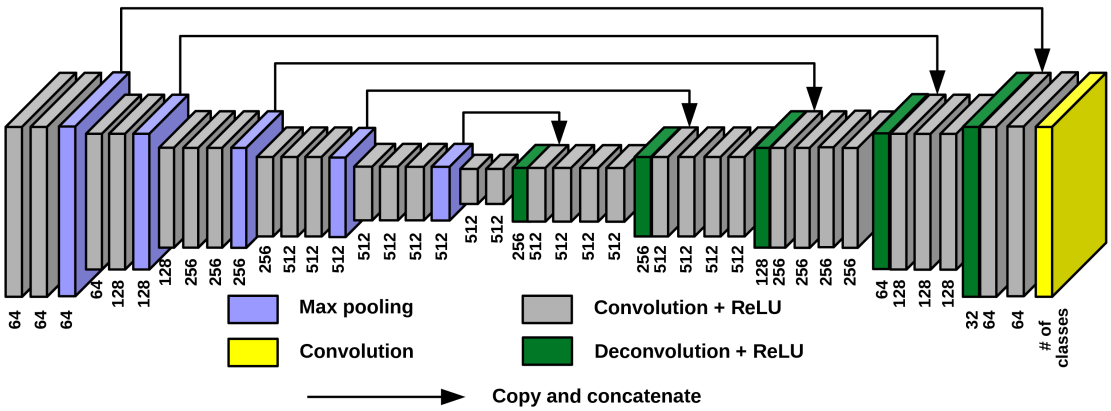


Figure 2.2: The network structure. The number below each layer corresponds to number of filters. We refer the last layer shown by yellow color as the classification, and the rest as the shared layers.

as it is not as deep as e.g., VGG19 but still it is one of the best performers on famous benchmark challenges (e.g., ImageNet [147]).

The output of each pooling layer in the encoder is concatenated with the output of the symmetric deconvolutional layer in the decoder through skip connections to combine higher level features with the lower ones. Kernel size and stride in all the convolutional layers are 3 and 1 respectively. Padding parameter in the convolutional layers is set to 1 so as to keep height and width of output the same as output of the previous layer. The max-pooling layers having 2×2 window with stride 2 are used to halve width and height of the previous layer. In order to upsample the output of the previous layer by factor of 2, both kernel size and stride parameters are set to 2 in deconvolutional layers. Except the last convolutional layer, all the convolution and deconvolution operations are followed by a ReLU. Since batch normalization uses the memory inefficiently, we prefer not to use it. We add more patches in a batch instead.

Multi-task learning is the learning strategy which solves multiple problems at the same time by learning all the tasks jointly. In deep neural networks, bottom layers enable to share information for all the tasks, whereas the last layers are dedicated to provide a solution for each task [26, 146]. In incremental semantic segmentation problem, since the label maps of a remote sensing image for a class or several classes come sequentially, we consider the segmentation tasks as a multi-task learning problem, where performing a binary classification for each class corresponds to a different task. The output of our network is a 3-D matrix that is a stack of binary predicted maps for all the classes. In the test stage, to generate a binary segmentation for each class, we first convert outputs of the final convolutional layers to probability maps using sigmoid; then, we threshold the probabilities at 0.5.

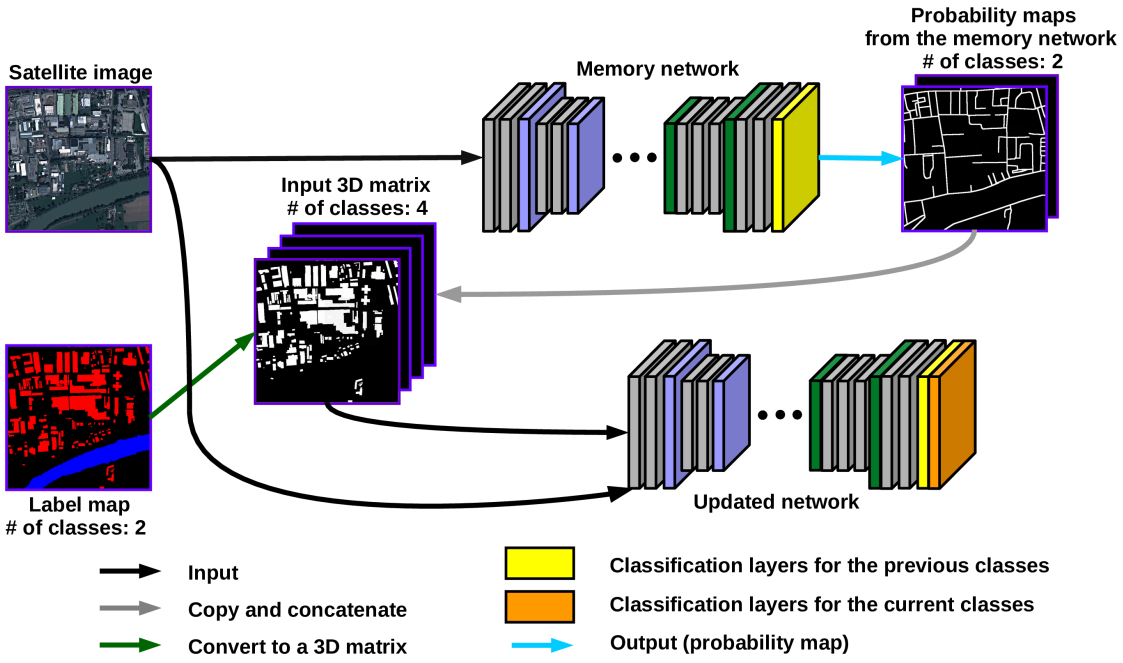


Figure 2.3: Adapting the network to new training data. Although annotations for only 2 classes are provided, the updated network is still able to learn the current classes as well as the previously learned 2 classes with the help of the memory network.

2.5.2 Adapting the Network to the New Training Data

To explain the adaptation phase, let us assume that the current training data are indicated by D_{curr}. We denote sets of the previously learned classes and the classes in D_{curr} by $\mathcal{L}_{prev}$ and $\mathcal{L}_{curr}$, where $\mathcal{L}_{prev} \cap \mathcal{L}_{curr} = \emptyset$. The main goals we try to achieve during adaptation are to update the formerly trained network so that segmentation capabilities for $\mathcal{L}_{curr}$ are added, and to fine-tune the network on D_{curr} for $\mathcal{L}_{prev}$, although annotations for $\mathcal{L}_{prev}$ are not available in D_{curr}. The output of the updated network is the matrix consisting of binary segmentations for $\mathcal{L}_{updated} = \mathcal{L}_{prev} \cup \mathcal{L}_{curr}$.

We use the knowledge distillation from the previously trained network, which we refer to as memory network, as a proxy in absence of the ground-truth for $\mathcal{L}_{prev}$ in D_{curr}. We create an updated network, having exactly the same structure except the last classification layer, which has $|\mathcal{L}_{updated}|$ filters instead of $|\mathcal{L}_{prev}|$. During creation of the updated network, additional $|\mathcal{L}_{curr}|$ filters in the last classification layer are initialized using Xavier initialization [59], and the rest of the parameters are loaded from the memory network. When D_{curr} arrive, the incoming label map is first converted to a 3-D matrix, consisting of binary ground-truth for $\mathcal{L}_{curr}$. The probability maps generated by the memory network are concatenated with this 3-D matrix to provide information about $\mathcal{L}_{prev}$ to the updated network. The final 3-D matrix as well as the input image in D_{curr} are fed to the network as the new training data. While concatenating output of the memory network with the new ground-truth, we prefer to use soft probability maps generated by the memory network rather than hard classification maps in order to reduce the propagated error rate, caused by imprecision in output of the memory network, at each time step of incremental learning.

Let us denote the binary target label vectors for n training samples $i = 1 \dots n$ in a batch from D_{curr} by $\mathbf{y}^{(i)}_{curr}$ and the predicted probabilities for $\mathcal{L}_{prev}$ from the memory network by $\hat{\mathbf{y}}^{(i)}_{mem}$. We denote by $\hat{\mathbf{y}}^{(i)}_{up\_curr}$ and $\hat{\mathbf{y}}^{(i)}_{up\_prev}$, the predicted probabilities for $\mathcal{L}_{curr}$ and $\mathcal{L}_{prev}$ from the updated network. The classification loss L_{class} quantifies mismatch between $\mathbf{y}^{(i)}_{curr}$ and $\hat{\mathbf{y}}^{(i)}_{up\_curr}$. In order to compute L_{class}, since we deal with generation of a binary segmentation for each class as a separate task, we use sigmoid cross entropy loss defined as:

$$L_{class} = -\frac{1}{n|\mathcal{L}_{curr}|} \sum_{i=1}^{n} \sum_{k=1}^{|\mathcal{L}_{curr}|} \left[y^{(i)}_{curr(k)} \log\left(\hat{y}^{(i)}_{up\_curr(k)}\right) + \left(1 - y^{(i)}_{curr(k)}\right) \log\left(1 - \hat{y}^{(i)}_{up\_curr(k)}\right) \right]. \tag{2.1}$$

In order for the updated network to learn $\mathcal{L}_{prev}$ on D_{curr}, we try to keep discrepancy between $\hat{\mathbf{y}}^{(i)}_{up\_prev}$ and $\hat{\mathbf{y}}^{(i)}_{mem}$ as small as possible. The distillation loss L_{distil}, which measures this disparity is defined as:

$$L_{distil} = -\frac{1}{n|\mathcal{L}_{prev}|} \sum_{i=1}^{n} \sum_{k=1}^{|\mathcal{L}_{prev}|} \left[\hat{y}^{(i)}_{mem(k)} \log\left(\hat{y}^{(i)}_{up\_prev(k)}\right) + \left(1 - \hat{y}^{(i)}_{mem(k)}\right) \log\left(1 - \hat{y}^{(i)}_{up\_prev(k)}\right) \right]. \tag{2.2}$$

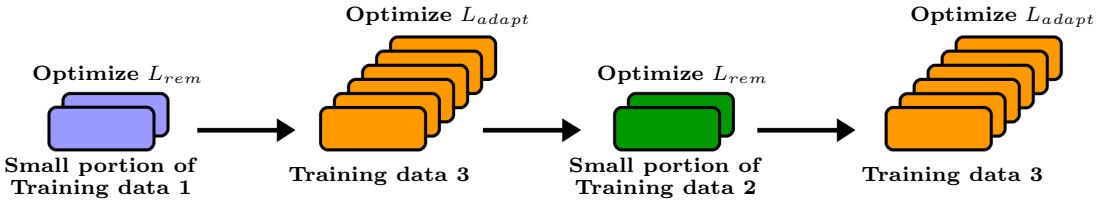


Figure 2.4: An example optimization sequence. The new classes are added on Training data 3 to the network, which was already trained on Training data 1 and Training data 2. The optimization sequence is as follows: L_{rem} on Training data 1, L_{adapt} on Training data 3, L_{rem} on Training data 2, and L_{adapt} on Training data 3 again.

The overall adaptation loss L_{adapt} that is optimized during adaptation is computed by adding these two terms:

$$L_{adapt} = L_{class} + L_{distil}. \tag{2.3}$$

Fig. 2.3 depicts how the network is adapted to the new data.

2.5.3 Remembering From the Previous Training Data

We denote the previous training data by $D_{prev} = D_{prev}^{(1)} \cup D_{prev}^{(2)} \cup \ldots \cup D_{prev}^{(m)}$, where $D_{prev}^{(1)}$ corresponds to the first data, $D_{prev}^{(2)}$ is the second data, and so forth. If the training data are captured sequentially from different geographic locations, in order for the network not to overfit on D_{curr} for $\mathcal{L}_{prev}$, we remind the previously learned information by systematically showing patches from the stored, small portion of D_{prev}. Since in most cases classes in the training data are highly imbalanced, when determining which training patches to store in $D_{prev}^{(j)}$, random selection may cause storing no samples for less frequent classes. For this reason, we take the class imbalance problem into account. We first compute weight w_c of each class $c \in \mathcal{L}_{prev}^{(j)}$ in $D_{prev}^{(j)}$ as:

$$w_c = \frac{\text{median}(f_c | c \in \mathcal{L}_{prev}^{(j)})}{f_c}, \tag{2.4}$$

where f_c denotes frequency of the pixels that are labeled as class c. We then assign an importance value $I^{(l)}$ to the l^{th} training patch in $D_{prev}^{(j)}$ as:

$$I^{(l)} = \sum_{c \in \mathcal{L}_{prev}^{(j)}} w_c f_c^{(l)}, \tag{2.5}$$

where $f_c^{(l)}$ denotes the number of pixels, belonging to c in the patch. We store certain number of patches that have the highest I value, which we denote by $D_{prev\_imp}^{(j)}$. In order to diversify the patches that are fed to the network, we randomly select a small fraction of the remaining patches. We denote the randomly chosen patches by $D_{prev\_random}^{(j)}$. The

Table 2.1: Advantages and disadvantages of each method.

Method	Static l.	Multiple l.	Fixed repr.	Fine-tuning	Inc. L. (ours)
Tr. Time (1 iter.)	fast	fast	very fast	fast	medium
Test Time	fast	very slow	fast	fast	fast
Performance for new classes	inc. learn. is not supported	good	very bad	good	good
Performance for old classes	inc. learn. is not supported	good	good	very bad	good
Convergence time for new classes	inc. learn. is not supported	medium	cannot learn	very fast	very fast
Number of Classifiers	1	N	1	1	1

data to be stored from $D^{(j)}_{prev}$ for remembering are $D^{(j)}_{prev\_rem} = D^{(j)}_{prev\_imp} \cup D^{(j)}_{prev\_random}$. The number of patches that is selected randomly and using the importance value needs to be determined by the end user.

Let us denote the target vector for the i^{th} sample among n samples in a batch from $D^{(j)}_{prev\_rem}$ by $\mathbf{y}^{(j)(i)}_{prev}$. We denote by $\hat{y}^{(j)(i)}_{up\_prev}$ the predicted vector from the updated network for the same sample. The remembering loss L_{rem} is calculated as:

$$L_{rem} = -\frac{1}{n|\mathcal{L}^{(j)}_{prev}|}\sum_{i=1}^{n}\sum_{k=1}^{|\mathcal{L}^{(j)}_{prev}|}\left[y^{(j)(i)}_{prev(k)}\log\left(\hat{y}^{(j)(i)}_{up\_prev(k)}\right)+\left(1-y^{(j)(i)}_{prev(k)}\right)\log\left(1-\hat{y}^{(j)(i)}_{up\_prev(k)}\right)\right]. \tag{2.6}$$

During remembering from $D^{(j)}_{prev}$, we freeze the classification layers that are responsible for $c \notin \mathcal{L}^{(j)}_{prev}$ and optimize the rest of the network. The user needs to determine how often and on which data L_{rem} is optimized. An example optimization sequence is depicted in Fig. 2.4.

2.6 Experiments

2.6.1 Methods Used for Comparison

Table 2.1 compares our methodology with the following approaches:

Static learning. This is the traditional learning approach, where we assume that all the training images and annotations for the same classes are available at the time of

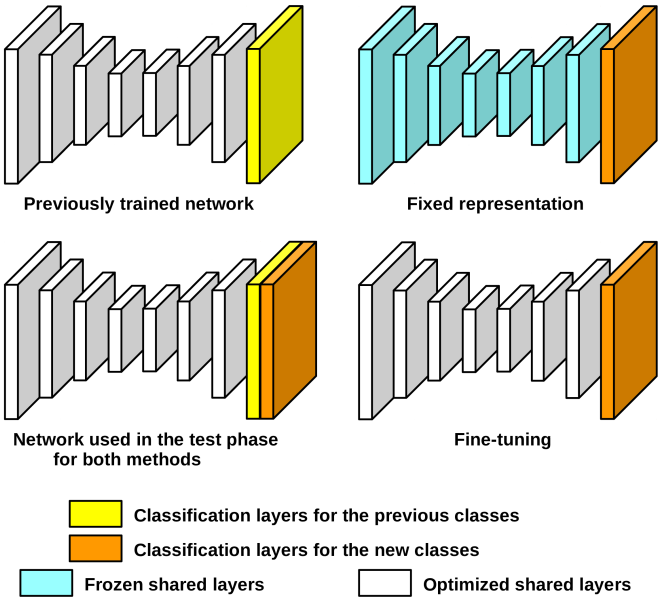


Figure 2.5: Example network structures for fixed representation and fine-tuning. During the test stage, classification layers for the previous classes are appended to the network to generate label maps for all the classes.

training. In real-world segmentation problems, this condition is extremely hard to meet. This method does not support learning new classes continually.

Multiple learning. In this learning strategy, we train an additional classifier whenever the new training data are obtained. The number of classifiers that needs to be stored increases linearly. In addition, because the test images have to be segmented using all the trained classifiers to generate a map for each class, the test stage might be extremely long. Therefore, this approach is overly expensive in terms of storage and segmentation efficiency.

Fixed representation. To learn new classes, we remove the classification layers, which were optimized for the previous classes, and plug in new classification layers dedicated for the new classes. The newly added classification layers are initialized with Xavier method [59]. When new training data arrive, we optimize only the newly added classification layers and freeze the rest of the network. Hence, training is very fast. During testing, we append the formerly trained classification layers back to the network to generate label maps for all the classes. The major issue is that although performance for the initial classes is preserved, the network struggles in learning new classes, because the previously extracted features are not optimized to represent the new classes.

Table 2.2: Training and validation cities in the first experiment.

Data Type and Classes for				City	Area (km^2)	Class Frequency (%)				
mult. learning fixed rep. fine-tuning inc. learning		static learning				Build.	High ve.	Road	Rail.	Water
Train1	building	Train	building	Bad Ischl	27.71	5.51	35.38	5.87	0.16	2.58
				Osttirol	28.38	6.96	15.37	7.72	0.44	0.84
				Voitsberg	28.70	7.47	30.21	6.54	0.44	0.98
				Bayonne-Bi.	66.58	15.21	19.45	12.66	0.45	1.26
				Bourges	72.20	9.81	14.83	10.10	0.42	0.92
	high veg.		high veg.	Draguignan	25.54	9.64	34.99	10.24	0.00	0.08
				Nîmes	26.62	21.78	19.65	14.10	1.30	0.04
Train2	road		road	Enns	64.49	6.25	12.54	6.81	1.36	2.82
				Innsbruck	132.50	8.92	22.78	7.00	0.90	2.97
				Klagenfurt	67.73	10.96	18.89	9.05	0.65	1.20
				Sankt Pö.	87.17	6.68	25.13	5.40	0.99	1.70
	railway		railway	Béziers	25.75	19.09	10.91	16.10	1.52	0.78
				Lyon	187.14	18.48	16.82	12.59	1.41	2.83
Train3	water		water	Albi	25.76	17.20	15.19	13.93	0.55	1.65
				Villach	43.59	9.26	19.91	9.61	1.02	2.69
				Salzburg	134.71	9.44	23.88	7.90	0.79	2.41
				Angers	74.16	15.78	15.97	10.40	0.63	1.39
				Douai	58.10	13.31	14.62	8.63	0.93	2.09
Validation	building high veg. road railway water			Amstetten	14.26	11.11	15.61	9.67	1.85	1.72
				Leibnitz	32.72	6.96	16.84	6.99	0.34	3.30
				Lille	117.58	18.36	15.40	11.39	1.32	1.02
				Roanne	25.84	18.44	8.33	14.00	0.78	0.95

Fine-tuning. We use a similar strategy that we follow in fixed representation. The only difference is that while training the network, instead of only the classification layers, we optimize the whole network using only the new training data. In this methodology, although the network performs a remarkable performance for the new classes, it suffers from catastrophic forgetting. Example network structures for fixed representation and fine-tuning, for both training and test phases, are illustrated in Fig. 2.5.

For our approach, it is required for the memory network to generate probability maps from the training patches to optimize L_{distil}. Therefore, training time for our approach is slightly longer than the others. This can be considered as the only disadvantage of the proposed methodology.

2.6.2 Data Sets and Evaluation Metrics

The first data we use contain 8-bit satellite images collected from 22 different cities in Europe. 11 of these cities are located in France and the other 11 are in Austria. The cities cover the total area of approximately 1367 km^2. The images were collected from

Table 2.3: F1 scores on the first data set. The numbers with a star (*) denote the quantitative results for static learning (the upper bound).

Method	**Epoch**	**Building**	**High ve.**	**Road**	**Rail.**	**Water**	**Overall**
static l.	**500**	80.74*	71.26*	66.21*	61.72*	82.74*	72.54*
multiple l.	**500**	71.25	68.88	59.28	**55.65**	79.83	66.98
fixed repr.	1000	71.25	68.88	2.71	0.00	—	
	1500	71.25	68.88	2.71	0.00	0.11	28.59
fine-tuning	1000	28.91	0.17	59.30	60.06	—	
	1500	27.90	7.71	0.14	0.01	**90.20**	25.19
inc. l.	1000	74.19	66.32	56.57	50.87	—	
w/o L_{rem}	**1500**	74.91	66.87	58.14	51.70	82.32	66.79
inc. l.	1000	75.98	72.38	57.29	50.18	—	
	1500	**76.78**	**72.06**	**59.58**	53.07	78.94	**68.09**
		Training Set 1		**Training Set 2**		**Tr. Set 3**	

the following cities: Amstetten, Enns, Leibnitz, Salzburg, Villach, Bad Ischl, Innsbruck, Klagenfurt, Osttirol, Sankt Pölten, Voitsberg in Austria, and Albi, Angers, Bayonne-Biarritz, Béziers, Bourges, Douai, Draguignan, Lille, Lyon, Nîmes, Roanne in France. The spectral bands of the images are composed of Red (R), Green (G), and Blue (B) channels. The spatial resolution is 1 m. Since the images were captured over different geographic locations, they have different color distributions and visual features. The annotations for building, road, high vegetation, water, and railway classes are provided.

The other two data sets on which we conduct our experiments are the Vaihingen and the Potsdam benchmarks provided by the ISPRS [85]. Both data sets contain 8-bit aerial images. The Vaihingen data set consists of 33 image tiles (of average size 2494 $\times$ 2064), where 16 of them are provided as training and the rest as test. The images comprise 3 spectral bands: Near Infrared (NIR), R, and G. The spatial resolution is 9 cm. The Potsdam data set includes 38 tiles (of size 6000 $\times$ 6000), out of which 24 are dedicated for training and the remaining for test. The images contain 5 channels: NIR, R, G, B, and the normalized DSM (nDSM) data. The resolution of the images in this benchmark is 5 cm. Both data sets contain full annotations for 6 classes: impervious surfaces, building, low vegetation, high vegetation, car, and clutter. However, since only 0.78% of the pixels in the Vaihingen data set is labeled as clutter, we ignore this class in the experiments on this benchmark. As of 2018 summer, the competition for these benchmarks is over, and all the reference data are publicly available. Hence, we use all the training tiles for training, and test tiles for validation. To account for the labeling mistakes while the data sets are annotated, the eroded ground-truth is also provided. We use this ground-truth to assess the performance on the benchmarks.

To quantitatively assess the performance for each class, we compare the binary predicted map and the binary ground-truth using two evaluation metrics: intersection over union (IoU) [42] and F1-score [74].

(a) $k = 1.4$ (b) $k = 0.6$ (c) Image (d) $\gamma = 0.6$ (e) $\gamma = 1.4$

Figure 2.6: Illustration of the contrast change (a - b) and the gamma correction (d - e) for an example input image (c).

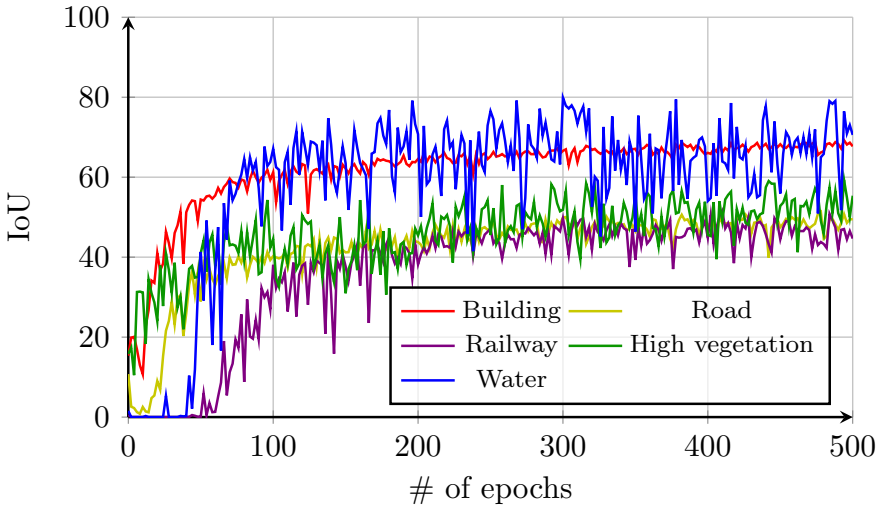


Figure 2.7: IoU vs. number of epochs plots for static learning on the 4 validation cities in the first experiment.

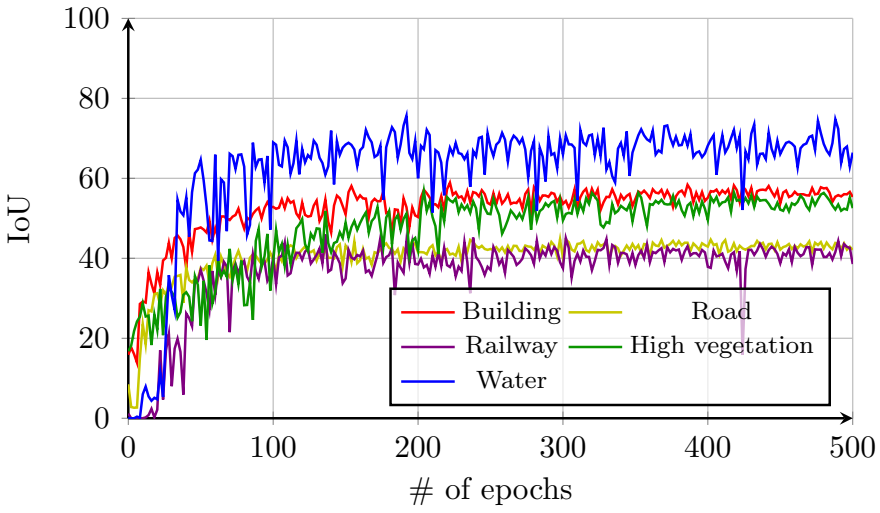


Figure 2.8: IoU vs. number of epochs plots for multiple learning on the 4 validation cities in the first experiment.

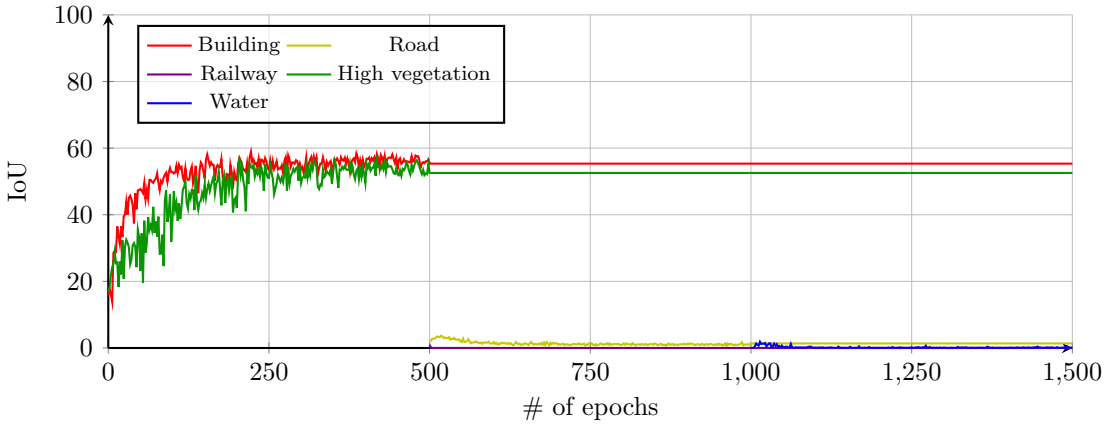


Figure 2.9: IoU vs. number of epochs plots for fixed representation on the 4 validation cities in the first experiment.

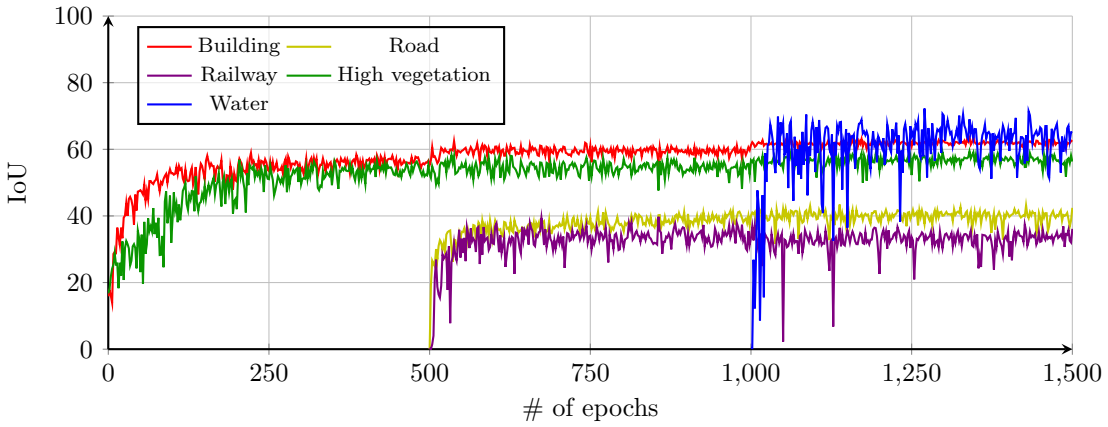


Figure 2.10: IoU vs. number of epochs plots for incremental learning on the 4 validation cities in the first experiment.

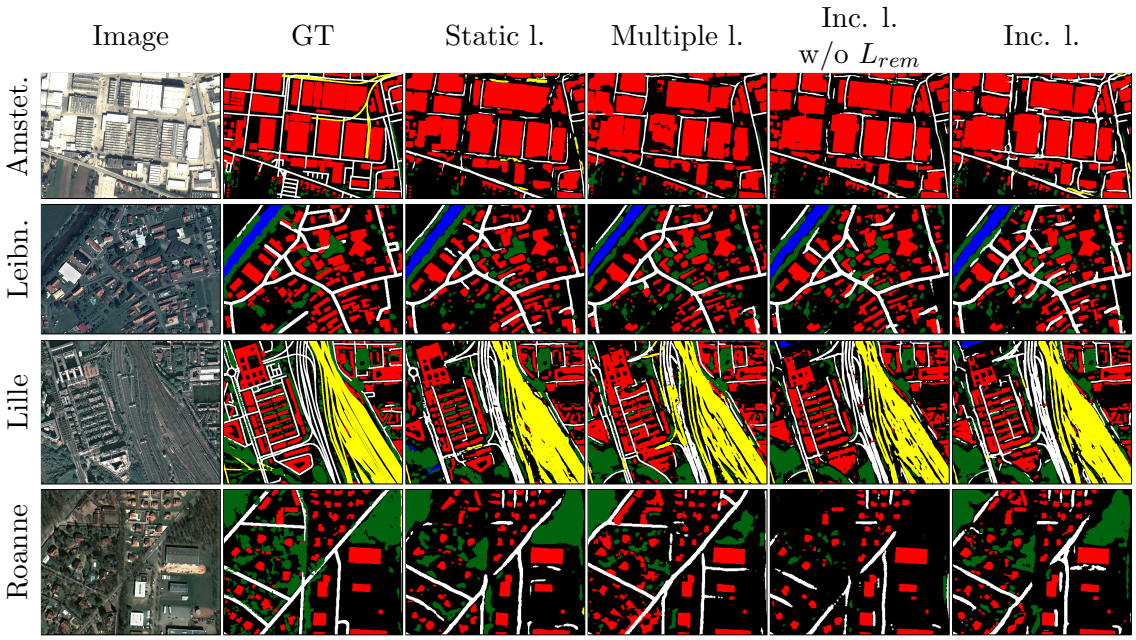


Figure 2.11: Close-ups from validation cities in the first data set. Classes: background (black), building (red), road (white), railway (yellow), high vegetation (green), and water (blue).

2.6.3 Experiments on the First Data Set

In this experimental setup, we suppose that the training data are obtained sequentially in time, and every snapshot of the streaming training data contains the satellite images from different cities and label maps for separate classes. We use 9 cities located in Austria and 9 cities in France as training data. We use 2 cities from each country for validation. We split the training cities into three sets as reported in Table 2.1 by paying attention that the cities in each set are the ones, which contain a reasonable amount of samples for the given annotations, and whose color distributions are as diverse as possible. We assume that the training cities are streamed in this order: Train1, Train2, Train3. For multiple learning, fixed representation, and fine-tuning we assume that the previous data are not accessible. For incremental learning, we store only 30% of the training patches in the previous data, out of which 15% are selected using the importance value and 15% are chosen randomly, as explained in section 2.5.3. We also evaluate our approach without accessing to the previous data (i.e., without optimizing L_{rem}), which we refer as incremental learning w/o L_{rem}. Since static learning does not support adding new classes continually, for this approach, we use all the training images from 18 different cities and label maps of all 5 classes for each image when training a network. For this reason, we expect it to be an obvious upper bound of the other methods.

During the pre-processing step, we split all the training images into 384 $\times$ 384 patches with an overlap of 32 $\times$ 32 pixels between the neighboring patches. The validation images are divided into 2240 $\times$ 2240 patches with 64 $\times$ 64 pixels of overlap. After all the validation patches are classified, they are combined back to get the original size classification maps. Because the satellite images arrive sequentially (except for static learning), it is not possible to compute mean values for the image channels. Hence, for the normalization, we subtract 127 from all the pixels, as the images are 8 bit.

We train a single model for static learning using the whole training data for 500 epochs, in which each epoch has 100 iterations. For multiple learning, we train 3 separate models from scratch on Train1, Train2, and Train3 with the same hyper-parameters. For fixed representation, fine-tuning, and the proposed incremental learning methodologies, every time when new classes are added from new data, we optimize the network for the same number of epochs and iterations as for static learning and multiple learning. In every 5 training iterations of the network for incremental learning approach on Train2, we optimize L_{rem} on Train1 for 1 iteration and L_{adapt} for the next consecutive 4 iterations. During the training on Train3, since the network has already learned information from both Train1 and Train2, we prefer to remind the network the previously learned information more often. On Train3, the optimization sequence as follows: L_{rem} on Train1 for 1 iteration, L_{adapt} for 2 iterations, L_{rem} on Train2 for 1 iteration, and L_{adapt} for 2 iterations again.

To update parameters of the network, we use Adam optimizer, where the learning rate is 0.0001, exponential decay rate for the first and the second moment estimates are 0.9 and 0.999, respectively. In every training iteration, a mini-batch of 12 patches is used for the optimization. When sampling a patch, we first select a random country (i.e., Austria or France). We then sample a random patch belonging to the city, which

is also randomly chosen from the selected country. While training the network we apply online data augmentation to enrich the training data. The patches are augmented by random vertical/horizontal flips, 0/90/180/270 degrees of rotations, and distorting their radiometry by random contrast change and gamma correction. The contrast of each channel in the image is changed as:

$$x_{curr} = (x_{prev} - \mu) * k + \mu, \tag{2.7}$$

where x_{prev} and μ are the pixel value and mean of all the pixels before the change, x_{curr} is the pixel value after the change, and k is the distortion factor, for which we generate a random value between 0.75 and 1.5. Gamma correction is formulated as:

$$x_{curr} = {x_{prev}}^{\gamma}, \tag{2.8}$$

where γ is the correction factor, which is drawn uniformly between 0.75 and 1.25. In Eqs. 2.7 and 2.8, we assume that the pixel values range between [0-1]. Fig. 2.6 illustrates the effect of gamma correction and the contrast change.

The overall F1-scores of all the classes on the first data set for each method are reported in Table 2.3. The method achieving the closest performance to the performance of static learning is highlighted. Figs. 2.7 to 2.10 depict the change of IoU values on the validation cities as the training progresses. Visual close-up results for static learning, multiple learning, incremental learning w/o rem, and incremental learning generated by the final models are shown in Fig. 2.11. Although our network generates a binary label map for each class, for the sake of concise and better visualization, we provide multi-class predicted maps obtained by assigning each pixel to the class, for which the highest probability is produced. In the figure, the pixels, having no probability higher than or equal to 0.5 are labeled as background.

As expected, static learning outperforms the other approaches on the first data set (see Table 2.3), because in the training stage, we feed much more and diverse training data to the model compared to the other approaches. Although static learning is superior to the other approaches on the first data set, it is applicable only if the data are static and the annotations are unique, which is almost never the case in real-world applications. In multiple learning, even if the previous data are not accessible, predicted maps for all the presented classes can be generated. However, because of the growing number of classifiers, this approach is inefficient in terms of test efficiency and storage. In addition, for each individual classifier, learning is limited to the data, on which the classifier was initially trained. For instance, building - high vegetation classifier trained on Train1 can not be fine-tuned on Train2, as annotations for these classes are not available on Train2.

In fixed representation methodology, the exact performance for the initially introduced classes is retained as neither the shared nor the classification layers for these classes change. On the other hand, the network performs extremely poorly for the new classes as shown in Fig. 2.9 and reported in Table 2.3. Overall, we conclude that shared layers of the network must be adapted to the new training data. When we apply fine-tuning, since instead of initializing all the parameters randomly, the extracted features

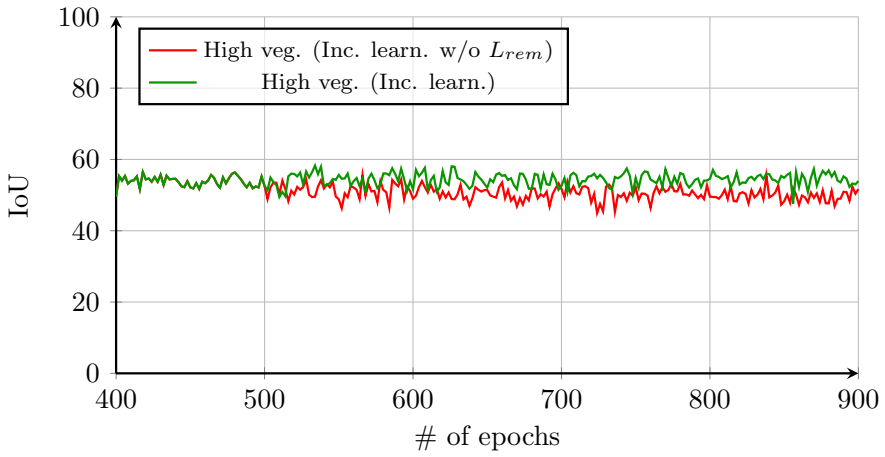


Figure 2.12: Comparison of incremental learning and incremental learning w/o L_{rem} for high vegetation class.

for the previous classes are used, performance for the new classes is remarkable, especially when there is only one class to be added. For instance, it is the best performer for water class. However, the results justify that the network catastrophically forgets the previously learned information.

As reported in Table 2.3, incremental learning exhibits the closest performance to static learning. Since our approach enables the network to learn the old classes on the new data and remember them from the previous data, the performance for the previous classes improves over time. If the previous data are never shown, the performance for the old classes may decrease as a result of adapting the network to the new data completely and imprecision of output of the memory network on the new data for the previous classes. Fig. 2.12 compares incremental learning and incremental learning w/o L_{rem} for high vegetation before and after adding road and railway classes on Train2 (i.e., before and after the 500^{th} epoch) to the building & high vegetation classifier trained on Train1. The close-ups from Roanne in Fig. 2.11 show that incremental learning w/o L_{rem} fails to detect a lot of high vegetation, whereas incremental learning exhibits a good performance. We also observe that incremental learning significantly outperforms multiple learning for building class. The reason is that the network in multiple learning learns building only on Train1, while incremental learning facilitates learning the same class from all the training data sequentially. Although when buildings are small and regular shaped as in Leibnitz and Roanne, both approaches generate similar outputs, multiple learning is not able to delineate the borders very well when buildings cover a large area as in Amstetten. Road and Railway classes turn out to be the most difficult classes, as the numeric results for them are substantially lower than for the other classes. As can be seen in the close-up from Lille, they visually look quite similar, which makes the classifiers confuse between them in some cases. Incremental learning seems detecting the roads and railways that are mis-classified by incremental learning w/o L_{rem}.

Table 2.4: F1 scores on the Vaihingen benchmark data set. The numbers with a star (*) denote the quantitative results for static learning (the upper bound).

Method	**Epoch**	**Build.**	**High v.**	**Imp. s.**	**Low v.**	**Car**	**Overall**
static l.	**500**	93.61*	87.87*	91.55*	81.05*	82.83*	87.38*
multiple l.	**500**	**94.43**	**88.12**	90.71	80.41	**87.90**	**88.31**
fixed repr.	1000	94.43	88.12	87.09	76.39	—	
	1500	**94.43**	**88.12**	87.09	76.39	13.37	71.88
fine-tuning	1000	52.40	0.03	91.83	80.99	—	
	1500	0.02	0.00	43.81	0.01	86.18	26.00
inc. l.	1000	94.34	88.02	91.42	81.65	—	
w/o L_{rem}	**1500**	94.31	88.07	**91.51**	**81.60**	81.69	87.44
		Training Set 1		**Training Set 2**		**Tr. Set 3**	

Table 2.5: F1 scores on the Postdam benchmark data set. The numbers with a star (*) denote the quantitative results for static learning (the upper bound).

Method	**Epoch**	**Build.**	**High v.**	**Clut.**	**Imp. s.**	**Low v.**	**Car**	**Ovr.**
static l.	**500**	96.83*	85.04*	54.57*	92.62*	85.69*	94.84*	84.93*
mult. l.	**500**	96.59	85.25	50.82	92.07	84.82	**95.36**	84.15
fixed repr.	1000	96.59	85.25	50.82	86.76	79.98	—	
	1500	96.59	85.25	50.82	86.76	79.98	72.14	78.59
fine-tuning	1000	0.00	44.53	3.23	92.13	85.45	—	
	1500	1.62	24.73	0.00	65.00	0.01	94.60	30.99
inc. l.	1000	96.91	86.12	50.23	92.20	85.64	—	
w/o L_{rem}	**1500**	**96.86**	**85.28**	**51.56**	**92.10**	**85.28**	94.43	**84.25**
		Training Set 1			**Training Set 2**		**T. S. 3**	

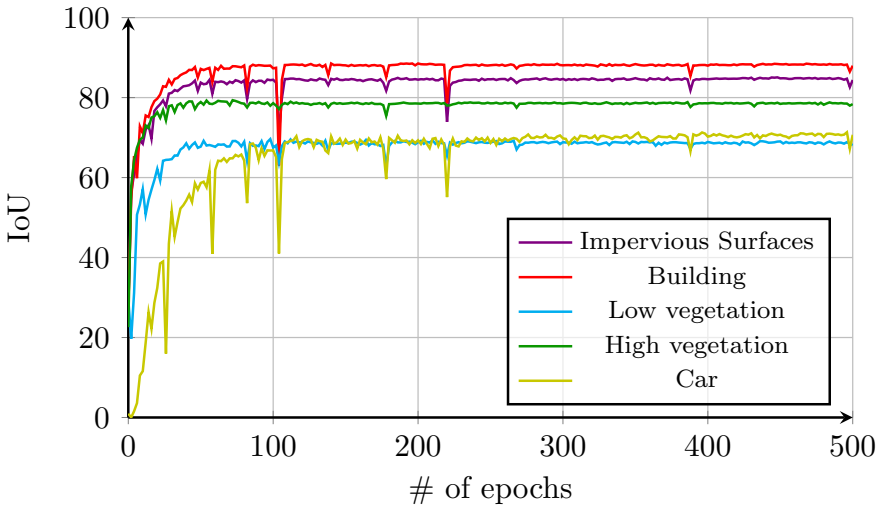


Figure 2.13: IoU vs. number of epochs plots for static learning on the Vaihingen benchmark.

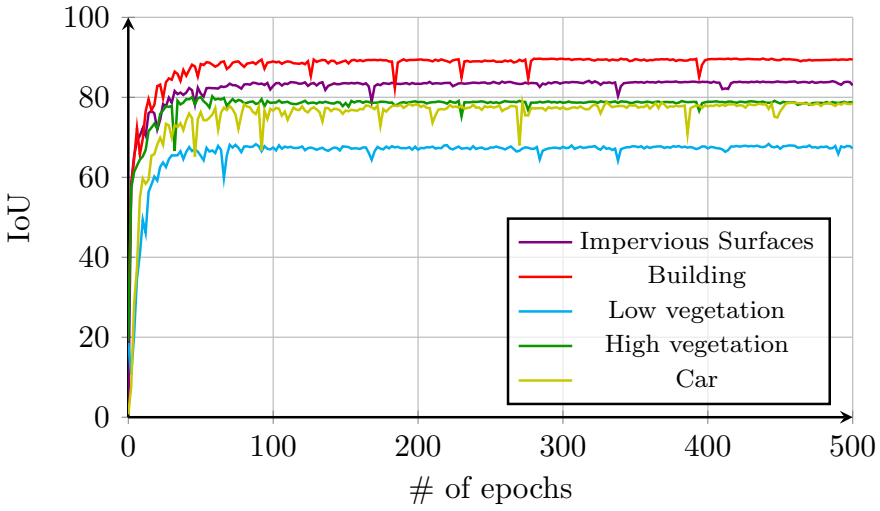


Figure 2.14: IoU vs. number of epochs plots for multiple learning on the Vaihingen benchmark.

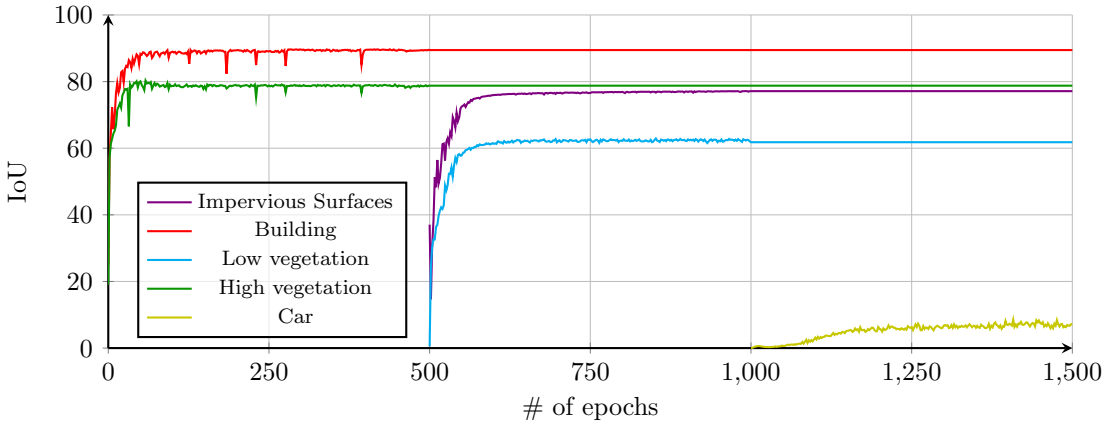


Figure 2.15: IoU vs. number of epochs plots for fixed representation on the Vaihingen benchmark.

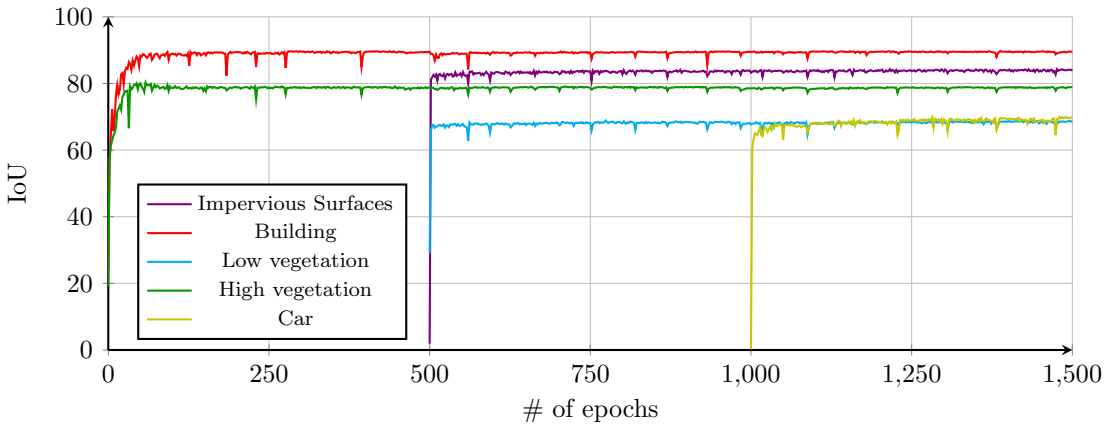


Figure 2.16: IoU vs. number of epochs plots for incremental learning w/o L_{rem} on the Vaihingen benchmark.

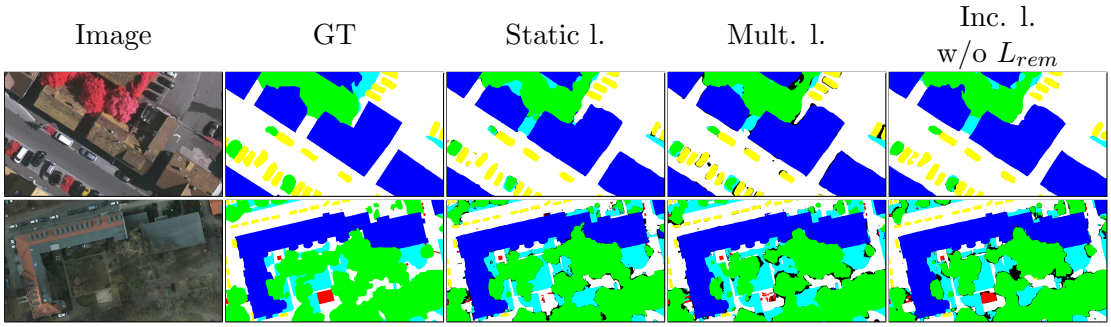


Figure 2.17: Close-ups from the benchmarks. Classes: background (black), impervious surfaces (white), building (blue), low vegetation (cyan), high vegetation (green), car (yellow), and clutter (red). The results in the upper row are from the Vaihingen data set, and the bottom row are from the Potsdam benchmark.

2.6.4 Experiments on the Benchmarks

In the experiments on the benchmarks, we assume that we have access to the whole training tiles, but we are provided the annotations sequentially. We suppose that every time when new annotations are retrieved, the previous ones are not accessible. On the Vaihingen data set, we consider that we retrieve label maps for building and high vegetation classes in the beginning. We are then given the ground-truth for impervious surfaces and low vegetation. Finally, we receive the annotations for car class. On the Potsdam data set, since there is an additional clutter class, we assume that the label map for this class is also available in the initial training data. For our approach, since we always use the same training images, we remind the network the old classes using output of the memory network (i.e., we only optimize L_{adapt}). On contrary the other approaches, for static learning, we use all training tiles as well as annotations for all the classes at once in the training stage.

Because the images in the benchmarks are of much higher resolution than the satellite images in the first data set, the patches need to be larger to cover a reasonable area. Therefore, we divide the training tiles into 512 $\times$ 512 patches. The validation tiles are split into 2000 $\times$ 2000 patches. The training and validation tiles have 64 $\times$ 64 and 120 $\times$ 120 pixels of overlap, respectively. We compute a global mean for each channel from the training tiles and subtract it from all the pixels.

For each approach, we train the same number of models for the same number of epochs and iterations using the same optimizer with the same parameters as in the experiments on the first data set. As size of the training patches is larger than in the previous experiments, we randomly sample 8 patches instead of 12. Another difference is that since both training and validation patches are from the same city, we augment the patches by only random flips and rotations.

We present the qualitative and quantitative experimental results on the benchmarks in a similar way described in Sec. 2.6.3. We report F1-score for each class in Tables 2.4 and 2.5. We depict the plots for IoU vs. number of epochs on the Vaihingen benchmark

in Figs. 2.13 to 2.16, and close-ups from both benchmarks in Fig. 2.17. As we use all the annotations at once for static learning, we again choose this approach as the reference method.

From the IoU vs. # of epochs plots in both experiments, our first observation is that IoU values for each model, as the training iterations continue, fluctuate much more on the first data set than on the Vaihingen benchmark. We also observe that models, trained from the Vaihingen data set converge faster. The reason for these two conclusions is that in the Vahingen data set, a single aerial image was split into smaller tiles, while images in the first data set were taken from different cities at different dates; therefore, they have distinct color variations and visual features. Furthermore, satellite images in the first data set are of much lower resolution, and the validation set consists of the cities that are not seen by the network during training. For all these reasons, accuracies for the same classes (i.e., building and high vegetation) are significantly lower in the first experiment.

Our observations for fixed representation and fine-tuning are similar to the first experiment. As can be seen in Fig. 2.15, for fixed representation, although some classes such as impervious surface and low vegetation can be learned relatively well, the network performs poorly when the newly added class represents small objects such as cars.

Since training as well as test tiles are from the same city, the output of the memory network becomes almost the ground-truth for the previous classes. As a result, even if annotations for the previous classes are not accessible, new classes can be learned while exhibiting a similar performance for the former classes. We justify this claim in Fig. 2.16, in which it is demonstrated that IoU plots for the previously learned classes remain quite flat over time. The predicted maps of the close-ups from Vaihingen in Fig. 2.17 for 3 approaches look very similar. The advantage of our approach is that with the help of the features for the previous classes, the network converges very fast for the new classes. For instance, as illustrated in Fig. 2.13, it takes roughly 50 epochs in order for the network to converge for low vegetation class when static learning is applied, whereas with the proposed approach, a similar accuracy for the same class can be achieved in only a few epochs, as depicted in Fig. 2.16.

In this experimental setup, if the classes have distinct visual appearance and features like in the Vaihingen benchmark, as the classification tasks are shared between several classifiers, multiple learning performs better especially when the class (e.g., car) has a low number of samples. As training tiles of the Potsdam data set contain the nDSM data, detecting car class is easier on this data set than on the Vaihingen benchmark. As reported in Table 2.5, the gap between multiple learning and the other approaches is smaller for this class. On the contrary, as can be seen in the last row in Fig. 2.17, clutter class has high visual similarities with some pixels labeled as impervious surfaces or low vegetation. Hence, a single classifier that is trained jointly for all the classes, performs better in distinguishing these classes. Unlike multiple learning, where several isolated classifiers are trained, our approach allows joint training via the memory network. Therefore, our approach performs better for these classes, as confirmed by Table 2.5. The last row in Fig. 2.17 exemplifies some mis-classified clutter pixels by

Table 2.6: Training times for each approach

Method	Patch Size	Batch Size	Time for 1 Iter. (seconds)
Inc. learn.	384	12	1.03
Fixed. Repr.	384	12	0.31
The Others	384	12	0.72
Inc. learn.	512	8	1.21
Fixed. Repr.	512	8	0.32
The Others	512	8	0.87

multiple learning but correctly detected by our approach.

2.6.5 Running Times

We have implemented all the approaches in Tensorflow, and conducted all the experiments on an Nvidia Geforce GTX1080 Ti GPU with 11 GB of RAM. Table 2.6 reports the training times for incremental learning, fixed representation, and the others. Let us remark that the training times for fine-tuning, multiple learning, and static learning are almost the same; therefore, we refer to these approaches as the others and report the average of their training times in the table. Since we optimize only the classification layer for fixed representation, its training is extremely fast. For incremental learning, probability maps from the current training batch need to be generated by the memory network for the knowledge transfer. Hence, the training time in each iteration of incremental learning is about 0.3 seconds longer than those of fine-tuning, multiple learning, and static learning.

2.7 Concluding Remarks

In remote sensing, it is quite often that data set providers are interested in generating maps for separate classes. Hence, it is of crucial importance to propose methods that are able to learn from multiple data sets having annotations for heterogeneous classes. If a learning system can accomplish this task when a new test image arrives, the classifier can generate a map for all the classes that it has incrementally learned over time.

In this chapter, we proposed a novel incremental learning methodology, which enables the neural network to learn segmentation capabilities for new classes while retaining dense labeling abilities for the formerly trained classes without using the entire previous training data. In our experiments, we first showed that the common learning approaches are extremely inefficient or inapplicable to learn from streaming data. We then demonstrated why using only the features extracted for the previous classes is inefficient to learn new classes. We also provided the results, showing that when the network is trained using only the new data without having any regularization, the learned information for the previous classes is catastrophically forgotten. Finally, on three different data sets we proved that the proposed approach achieves a high performance for new classes without

forgetting old classes.

The major drawback of the proposed incremental learning approach is that the memory network generates imperfect probability maps for old classes in the adaptation phase, when there exists a large data distribution difference between the current and the previous data sets. Such distribution differences usually originate from various atmospheric conditions such as sensor characteristics or differences in acquisition. This problem is referred to as domain adaptation in the literature [175]. The next three chapters tackle the domain adaptation problem.

As future work, it is worth exploring how to incorporate domain adaptation techniques with our incremental learning methodology so that the trained network could better generalize to the data collected from new geographic locations.

Chapter 3

City-to-city Domain Adaptation

3.1 Problem Statement

In this chapter, we deal with unsupervised city-to-city domain adaptation problem. Our goal is to devise efficient methods enabling to train a classifier on an annotated source city and to generate high quality maps for an unlabeled target city, under a large data distribution difference between the cities. The inputs for the proposed approaches are a satellite image from a source city, its ground-truth in raster format, and a satellite image from a target city. The desired output is an automatically generated precise raster map for the satellite image captured over the target city. In the context of domain adaptation, a satellite image collected over a geographic location can be considered as a domain. Hence, we use "domain" and "a satellite image" statements interchangeably.

3.2 Motivations

Although various benchmarks such as INRIA [111] and SpaceNet [177] have proven that convolutional neural networks (CNNs) are one of the most promising methods for semantic segmentation of remote sensing data, it is a known issue that their performance crucially depends on the representativeness of the source data [175]. However, in real-world remote sensing applications, it is often the case that the source domain is not representative for the target domain, due to a large data distribution difference between the domains caused by various reasons such as atmospheric effects, intra-class variations, and differences in acquisition. The large data distribution difference between the source and the target domains causes the current state-of-the-art supervised learning approaches to output unsatisfactory maps [175]. For the same reason, the memory network introduced in the previous chapter outputs imperfect predictions for the previously learned classes from new satellite images.

The above-mentioned issues have motivated us to develop new methods that are robust to a considerable data distribution difference between the domains. The most naive approach to hinder the performance deficiency under a large distribution difference between the domains is to manually annotate some portions of the target domain to fine-

tune [110] the already trained model on the source domain. However, such an approach requires human intervention in the process, and labeling even a small portion of a satellite image is time-consuming. To increase the generalization capabilities of CNNs, one of the most commonly used methods is to perform online data augmentation with random color changes [25]. For instance, gamma correction and random contrast changes are widespread used in remote sensing [170, 171]. However, these augmentation approaches are limited when it comes to adapting the model from one domain to another, when there exists a significant domain shift between the images.

A more powerful data augmentation approach would be to use generative adversarial networks (GANs) [61] to generate a fake source domain with a similar data distribution to that of a target domain [166]. If this task can be achieved, one can use the fake source domain and the ground-truth for the real source domain to train a classifier or to fine-tune the classifier already trained on the real source domain. The main challenge here is to generate a fake source domain that looks like the target domain while keeping it semantically consistent with the real source domain. For example, if the method replaces some objects with others or adds artificial structures to the fake domain in the generation process, the fake domain would not match with the ground-truth of the real source domain, and we could not train a classifier.

In this chapter, we tackle the aforementioned challenge to address the city-to-city domain adaptation problem.

3.3 Related Work

The existing approaches on domain adaptation problem can be split into two main categories as adapting the classifier and adapting the inputs.

3.3.1 Adapting the Classifier

The goal of the approaches falling under this category is to adapt a classifier from an annotated source domain to an unlabeled target domain, without modifying the data.

A common approach is to perform multi-task learning, where one of the tasks is to train a classifier from the source domain via common supervised learning approaches, and the other one is to align the features extracted from both source and target domains by adversarial training [72, 76, 173]. A similar approach [48] has also been applied to remote sensing data (SpaceNet challenge [177]).

The adaptation can also be performed by expectation-maximization algorithm (EM), variants of support vector machine (SVM) [41], and covariance and tensor alignment. Bruzzone *et al.* described an approach that updates the parameters of the previously trained classifier on the unlabeled target domain via the EM algorithm [22]. This approach has then been extended to a cascade classification framework [23] and multiple cascade classifiers pipeline [19]. Bruzzone and Marconcini proposed the domain adaptation support vector machine (DASVM) [20]. Ma *et al.* introduced an approach based on centroid and covariance alignment [109]. Qin *et al.* used tensor alignment for domain

adaptation [135].

Other attempts in the literature aim at solving the domain adaptation problem by regularizing or normalizing specific layers of the network, or by self-learning. The fully convolutional tri-branch network (FCTN) [193] and the class-balanced self learning approach (CBST) [200] are examples of self-learning based methods. Romijnders *et al.* proposed a new domain agnostic normalization layer [143]. Saito *et al.* introduced a new adversarial dropout regularization technique [150]. The IBN-Net [129] combined the batch normalization with the instance normalization [176]. Zhu *et al.* proposed a new conservative loss [199]. Gross *et al.* presented a non-linear feature normalization for domain adaptation of hyper-spectral data [63].

3.3.2 Adapting the Inputs

These methods usually aim at either modifying the source domain to make its data distribution similar to that of the target domain, or mapping the domains into a common space to make their distributions close to each other. If the former approach is adopted, a classifier is trained on the modified source domain and evaluated on the real target domain. If the latter approach is preferred, a classifier needs to be trained on the modified source domain and evaluated on the modified target domain.

To match the distribution of the source domain with the distribution of the target domain, histogram matching [60, 82] and graph matching [174] are commonly used. Another way is to use style transfer or image-to-image translation (I2I) methods to generate a fake source domain stylized as the target domain. For instance, CyCADA [71] generates a fake source domain with CycleGAN [198] and uses it to train a classifier. Similarly, Benjdira *et al.* use CycleGAN for I2I between aerial images [13]. Other state-of-the-art I2I approaches such as UNIT [104], MUNIT [78], DRIT [96] are also relevant for domain adaptation.

To bridge the distribution gap between the domains by mapping them into a common space, one may think of applying color constancy algorithms [3] such as gray-world [24] and gamut [53]. These algorithms assume that colors of the objects are highly affected by the color of the illuminant and try to remove this effect. In some cases, the domain shift can be corrected by radiometric correction [86, 128] as well.

3.4 Contributions

The unsupervised domain adaptation assumes that annotations for any part of the target domain are not available, and aims at generating a high quality map even when there is a significant distribution difference between the domains. To tackle this challenge, the way we approach the problem is methodologically based on adapting the inputs. The contributions of this chapter are as follows:

- ColorMapGAN and SemI2I: We propose two novel GANs based I2I approaches, coined ColorMapGAN and SemI2I, which are able to generate a target stylized fake source domain that is semantically consistent with the real source domain.

After generating the fake domain, we fine-tune the already trained classifier using the fake source domain and the ground-truth of the real source domain.

- Lower complexity: The training time of both approaches are shorter than the existing GANs due to their simple architectural designs. Especially the training time of ColorMapGAN is considerably short, because its generator does not perform any convolution or pooling operations unlike the GANs in the literature. It transforms the colors of the source domain with only one element-wise matrix multiplication and one matrix addition.

- Validation on Pléiades data: We perform city-to-city adaptation between two image pairs collected over four European cities by Pléiades satellite. We compare our approaches with nine competitive methods and verify that our techniques outperform the others.

3.5 Background on Generative Adversarial Networks

In machine learning, we can divide the models trained in a supervised setting into two groups: discriminative and generative models. In the field of image analysis, the discriminative models are usually trained to learn a mapping from a high dimensional input to class labels as in image categorization and segmentation problems. On the other hand, the generative models aim to estimate the distribution of the data samples so that new samples can be drawn from the estimation. In 2014, Goodfellow *et al.* proposed the generative adversarial networks (GANs) [61], which is a combination of generative and discriminative models.

GANs usually comprise a generative model G and a discriminative model D. The goal of G is to estimate the distribution of the real data and to output fake data from the estimation. G takes a random noise z as input, and represents a mapping to data space $G(z)$. We denote the distribution of the real data x by $p(x)$ and a prior on input noise variables by $p(z)$. Let us assume that the real data x and the fake data $G(z)$ are indicated by 1 and 0, respectively. D outputs a scalar between 0 and 1, and aims to maximize the probability of labeling x and $G(z)$ correctly. In other words, the goal of D is to discriminate between the real and the fake data. The objective function for D that is maximized during training is described as:

$$\max_{D} V(D) = \mathbb{E}_{x\sim p(x)}\log[D(x)] \ + \ \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))], \tag{3.1}$$

where $\mathbb{E}$ is the expected value. G is simultaneously trained to minimize the objective function defined as:

$$\min_{G} V(G) = \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]. \tag{3.2}$$

As a result, the minimax game played between G and D could be formulated as:

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim p(x)}\log D(x) \ + \ \mathbb{E}_{z\sim p_{(z)}}\log(1 - D(G(z))). \tag{3.3}$$

Once D and G are simultaneously trained for a sufficiently long time, D becomes good at discriminating the fake and the real data, and G becomes better at generating fake data that are indistinguishable from the real data.

Although GANs work well with shallow multi-layer perceptrons, they suffer from instability problems during training when a more complex network is used. Several approaches have been presented to address the instability issues. In DCGAN [136], instead of multilayer perceptrons, deep convolutional networks were used in both G and D, and certain architectural constraints were introduced for a more stable training. In WGAN [8], rather than the logarithms in Eq. (3.3), Wasserstein distance was used to compute distance between the distributions, and gradient clipping was applied in the training stage. WGAN-GP [64] is an extension of WGAN, where a gradient penalty is performed to solve the limitations of the gradient clipping. LSGAN [116] proved that adopting the least squares loss function in Eq. (3.3) allows more stabilized training.

Finally, the original GANs have been extended to conditional GANs [123], where instead of generating the fake data from noise z, both G and D are conditioned on some extra information y. y can be class labels or data from other modalities. In this architecture, G learns a mapping from combination of z and y to the data space. If we consider y as the source domain and x as the target domain, G aims to learn a mapping from source domain to target domain. Inspired from this idea, conditional GANs have been used for several I2I approaches [84, 198].

3.6 Methods

3.6.1 Overall Segmentation Framework

Fig. 3.1 depicts the overall segmentation framework consisting of 4 steps as follows:

- Training the initial classifier: We train a classifier on the real source domain.
- Image-to-image translation: We generate a target stylized fake source domain that is semantically consistent with the real source domain using the proposed ColorMapGAN or SemI2I.
- Fine-tuning: We fine-tune the model obtained in step 1 using the fake source domain and the ground-truth for the real source domain.
- Classification: Finally, we generate a map for the target domain.

We use a slightly modified version of U-net [144] as the classifier. We replace rectified linear activation units (ReLU) by leaky rectified linear activation units (Leaky-ReLU) for a better performance [182]. We also remove the batch normalization operation in each layer, since it uses the memory inefficiently. In the framework, the steps 1, 3, and 4 are self-explanatory, whereas step 2 needs further explanation.

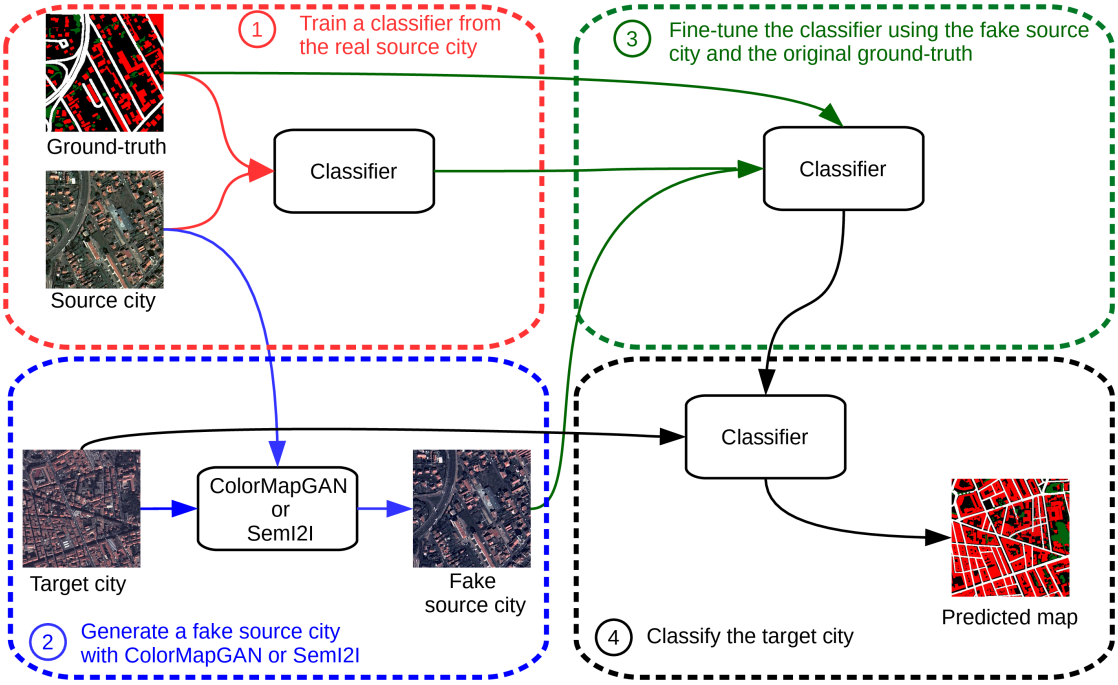


Figure 3.1: Overall segmentation framework.

3.6.2 ColorMapGAN

The novelty of the proposed ColorMapGAN lies in the architecturally simple but powerful design of its generator.

We denote a set of patches from the source domain by $S = \{s_1, s_2, \ldots, s_N\}$ and a set of patches from the target domain by $T = \{t_1, t_2, \ldots, t_M\}$. $G(S)$ corresponds to the set of fake source domain patches generated by G. The goal of G is to generate $G(S)$, whose spectral distribution is as similar as possible to the distribution of T, while keeping $G(S)$ and S semantically exactly the same. Contrary to the existing GANs in the literature, we do not use convolutional or pooling layers in G to preserve the exact semantics of S in $G(S)$. Fig. 3.2 depicts the overall flowchart for the feedforward pass of G.

Let us assume that S and T are composed of 8-bit images comprising red, green, and blue channels. We denote by $R = \{0, 1, \ldots, 255\}$, $G = \{0, 1, \ldots, 255\}$, and $B = \{0, 1, \ldots 255\}$, the values each color band of the pixels can take. We denote all the possible $16,777,216$ $(256 \times 256 \times 256)$ colors by RGB, which is defined as:

$$RGB = R \bigtimes G \bigtimes B, \tag{3.4}$$

where $\bigtimes$ stands for the Cartesian product. In order to transform RGB to another color matrix $R'G'B'$, we use a scale W and a shift K matrices with the same size as RGB. $R'G'B'$ could be computed as:

$$R'G'B' = RGB \circ W + K, \tag{3.5}$$

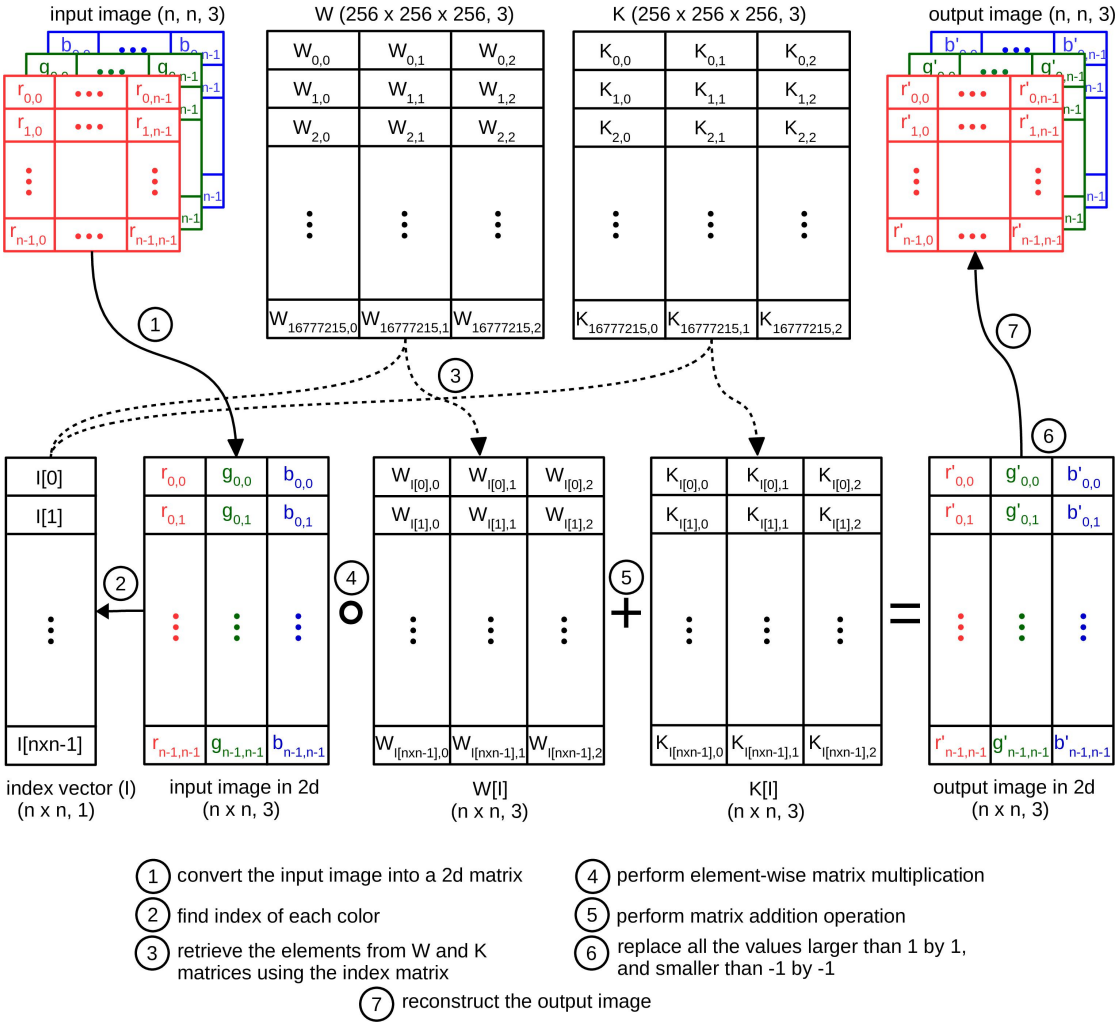


Figure 3.2: Overall flowchart for the feedforward pass of the generator in ColorMapGAN.

where $\circ$ denotes the element-wise product. The only learnable parameters of our G are W and K. Before the training starts, we initialize W with ones and K with zeros. Hence, at the end of the first iteration, the input and the output of G are exactly the same.

The main bottleneck for computing Eq. (3.5) is the computational complexity. Since each of RGB, W, and K matrices has more than 50 millions of elements ($256 \times 256 \times 256 \times 3$), it is not feasible to perform the operation defined in Eq. (3.5) on a GPU. However, the number of colors in a training image patch is substantially lower than the number of all the possible colors. Therefore, it is sufficient to update only the elements of W and K which transform the colors that are available in the training patch. To do this, we use an index vector I that is defined as:

$$I = r \times 256 \times 256 + g \times 256 + b, \tag{3.6}$$

where r, g, b are red, green, blue values of all the pixels in the training patch. After the elements of I are found, we normalize and center each $s_i \in S$ and $t_j \in T$ by first dividing by 127.5 and then subtracting 1 so that each color channel ranges between -1 and 1. We then partially update $R'G'B'$ as:

$$R'G'B'[I] = RGB[I] \circ W[I] + K[I], \tag{3.7}$$

where $[\cdot]$ operation corresponds to retrieving the rows of an arbitrary 2-D matrix indexed by the given vector. Then, in $R'G'B'[I]$, we replace the elements that are bigger than 1 and smaller than -1 by 1 and -1, respectively. To range all the values in $R'G'B'[I]$ between 0 and 255, we then use the denormalization function DN that is defined as:

$$DN(p) = \lfloor (p + 1) \times 127.5 \rfloor, \tag{3.8}$$

where p is a 2-D input matrix. The final output of G can be obtained by reshaping $DN(R'G'B'[I])$ back to the shape of the input patch. In each training iteration, we update only $W[I]$ and $K[I]$.

The discriminator in ColorMapGAN is architecturally the same as the discriminator of CycleGAN [198] (see Fig. 3.3). Instead of outputting a single scalar for the whole image patch to determine if the patch is real or fake, this discriminator generates a two-dimensional matrix. Each element of the matrix is used to locally determine whether the input patch is real or fake. We then take average of all the elements of the matrix to yield a final scalar quantifying the realness of the input image patch.

As mentioned in Sec. 3.5, GANs suffer from the instability issues; therefore, other objective functions have been proposed as an alternative to Eqs. 3.1, 3.2, and 3.3. We prefer to use the functions presented in LSGAN [116], as they are commonly used in state-of-the-art I2I methods [78, 96, 104, 198]. In LSGAN, the adversarial loss for D is defined as:

$$\mathcal{L}_{D_{adv}}(X, Y) = \mathbb{E}_x[(D(x) - 1)^2] \; + \; \mathbb{E}_y[(D(G(y)))^2], \tag{3.9}$$

where x and y denote randomly sampled patches from X and Y. The adversarial loss for G is defined as:

$$\mathcal{L}_{G_{adv}}(X, Y) = \mathbb{E}_y[(D(y) - 1)^2]. \tag{3.10}$$

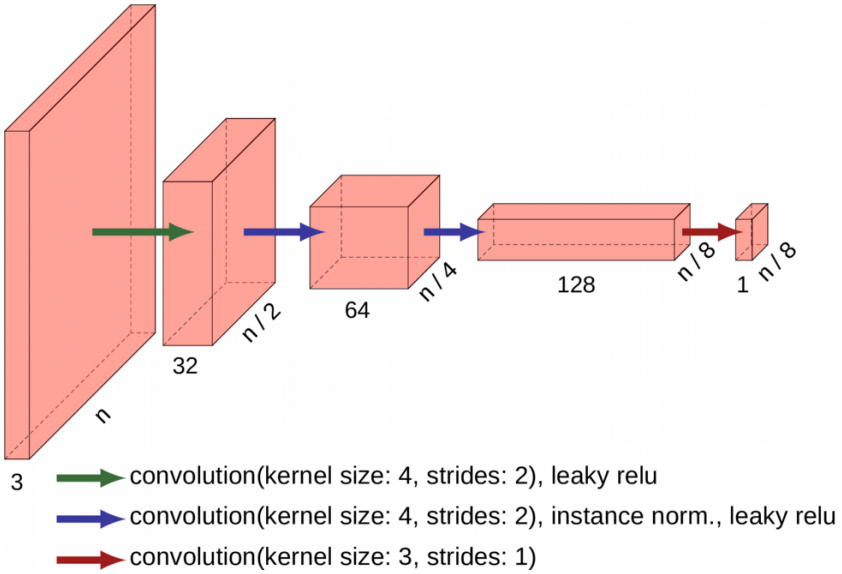


Figure 3.3: The architecture of the discriminator used in both ColorMapGAN and SemI2I. n denotes the patch size, and the number below each layer corresponds to the number of channels in the activation.

We compute the losses for the discriminator and the generator in ColorMapGAN as:

$$\mathcal{L}_{ColorMapGAN\_D} = \mathcal{L}_{D_{adv}}(T, S) \tag{3.11}$$

and

$$\mathcal{L}_{ColorMapGAN\_G} = \mathcal{L}_{G_{adv}}(T, S). \tag{3.12}$$

We simultaneously train D and G in ColorMapGAN by minimizing Eqs. 3.11 and 3.12.

3.6.3 SemI2I

Let us assume that both domains are denoted by A and B, respectively. The essential goal of SemI2I is to generate a fake A with the style of B, and a fake B with the style of A. The design of SemI2I is depicted in Fig. 3.4. As can be seen in the figure, there are two encoders and two decoders. One of the encoders extracts embeddings from A and fake B, and the other one encodes B and fake A. Similarly, the decoders are used to decode the embeddings of A and fake B, and B and fake A. In SemI2I, there are also two discriminators, where one discriminates between A and fake B, and the other one distinguishes B from fake A. As in ColorMapGAN, we use the same discriminators in CycleGAN [198].

The crucial components for the style transfer are adaptive instance normalization (AdaIN [77]) and adversarial losses. To generate a B stylized fake A and an A stylized fake B, we first encode both A and B using domain specific encoders. In the encoded embeddings, we compute mean μ and standard deviation σ values of each feature channel.

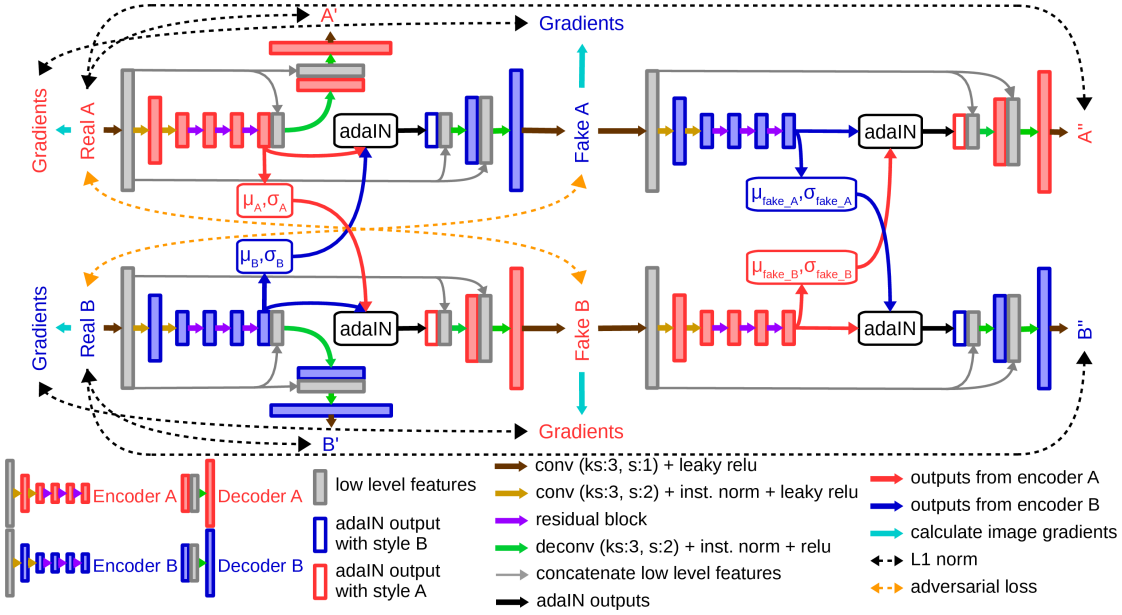


Figure 3.4: SemI2I. ks and s stand for kernel size and strides. μ and σ are mean and standard deviation values for each feature channel of the embeddings. Since SemI2I performs I2I, one can consider A as source and B as target domains, or vice versa.

The embedding itself has the content information, whereas μ and σ carry the style information. To combine the content of A with the style of B and the content of B with the style of A, we first normalize the embeddings by using AdaIN [77] that is defined as:

$$\text{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y), \tag{3.13}$$

where x and y denote the embeddings for both domains. AdaIN aligns mean and standard deviation for each channel of x with those of the channels of y. In other words, it allows us to switch the styles of A and B. We then decode the normalized embeddings to generate fake A and fake B. After fake A and fake B are obtained, we need to ensure that fake A looks like B, and fake B is visually similar to A. To enforce such constraint, we use the same adversarial losses defined in Eqs. 3.9 and 3.10. The adversarial losses for the generators and the discriminators in SemI2I are computed as:

$$\mathcal{L}_{SemI2I\_D\_adv} = \mathcal{L}_{D_{adv}}(\text{A}, \text{B}) + \mathcal{L}_{D_{adv}}(\text{B}, \text{A}) \tag{3.14}$$

and

$$\mathcal{L}_{SemI2I\_G\_adv} = \mathcal{L}_{G_{adv}}(\text{A}, \text{B}) + \mathcal{L}_{G_{adv}}(\text{B}, \text{A}). \tag{3.15}$$

As mentioned in Sec. 3.2, one of the main challenges is to keep A and fake B, and B and fake A semantically the same. To impose such consistency, we define several constraints. First of all, we switch the styles of fake A and fake B to obtain A″ and B″,

by following the same steps when we generate fake B and fake A from A and B. In an ideal transformation A and A$''$, and B and B$''$ have to be exactly the same. Hence, we minimize the cross reconstruction loss $\mathcal{L}_{SemI2I\_cross}$ defined as:

$$\mathcal{L}_{SemI2I\_cross} = |\mathrm{A} - \mathrm{A}''| + |\mathrm{B} - \mathrm{B}''|. \tag{3.16}$$

Secondly, when we decode the embeddings of A and B with their own decoders, we obtain A$'$ and B$'$. Here, since the embedding of a domain is decoded with its own decoder, the domain itself needs to be reconstructed. Therefore, we force A and A$'$, and B and B$'$ to be as similar as possible. We minimize the self reconstruction loss $\mathcal{L}_{SemI2I\_self}$ that is computed as:

$$\mathcal{L}_{SemI2I\_self} = |\mathrm{A} - \mathrm{A}'| + |\mathrm{B} - \mathrm{B}'|. \tag{3.17}$$

In addition, the textural features of A and fake A, and B and fake B must be very close. Let us assume that $Gr(\cdot, \cdot)$ is a function that takes two three-band images as inputs, converts them to gray-scale images, computes their horizontal and vertical gradients by Sobel filter, and sums L1 norm between the horizontal and L1 norm between the vertical gradients. We minimize the edge loss $\mathcal{L}_{SemI2I\_edge}$ as:

$$\mathcal{L}_{SemI2I\_edge} = Gr(\mathrm{A}, \text{fake A}) + Gr(\mathrm{B}, \text{fake B}). \tag{3.18}$$

Note that other textural features such as Haralick features [65] can be considered as well. However, we prefer Sobel operator mainly because of its short execution time.

Finally, as it has been proven that the filters in the first convolutional layer of each encoder learns domain independent low level features such as edges [190], we re-size and concatenate these low level features from each encoder with the input to each deconvolution layer in the corresponding decoder (see gray arrows in Fig. 3.4). Such concatenation allows the decoder to have a footprint of the objects in the real data; therefore, it guides the decoder to place the right objects in the correct locations. The overall generator loss for SemI2I becomes:

$$\mathcal{L}_{SemI2I\_G} = \lambda_1 \mathcal{L}_{SemI2I\_cross} + \lambda_2 \mathcal{L}_{SemI2I\_self} + \lambda_3 \mathcal{L}_{SemI2I\_edge} + \lambda_4 \mathcal{L}_{SemI2I\_G\_adv}, \tag{3.19}$$

where $\lambda_1, \lambda_2, \lambda_3$, and λ_4 are used to adjust the relative importance of each loss. The discriminator loss is defined as :

$$\mathcal{L}_{SemI2I\_D} = \lambda_4 \mathcal{L}_{SemI2I\_D\_adv}. \tag{3.20}$$

To train SemI2I, we simultaneously minimize $\mathcal{L}_{SemI2I\_G}$ and $\mathcal{L}_{SemI2I\_D}$.

The combination of an encoder and a decoder constitutes a generator. In the test stage, to generate a B stylized fake A, we need the encoder A and the decoder B (see the top left generator in Fig. 3.4 that generates fake A). However, as can be seen in Fig. 3.4, before feeding the embedding encoded by the encoder A from real A to the decoder B, it needs to be normalized by AdaIN using μ_B and σ_B calculated from the embedding of B. When generating fake A, one may think of randomly sampling a patch from B, encoding it by the encoder B, computing its μ and σ values, and using them to normalize the

embedding of A. However, this is not a good idea, because depending on which patch is sampled from B, we may end up generating a fake A having a different style in each run. For instance, a patch sampled from a forest and another patch sampled from an industrial area will have substantially different μ and σ values, as their styles are quite different. We have exactly the same problem when generating fake B. In the test stage, we want SemI2I to generate exactly the same fake data every time when we run it. To do this, we propose to estimate the global μ and σ values for the embeddings of both domains via Alg. 1 during the training. In Alg. 1, d_rate is a parameter ranging between 0 and 1. Note that this parameter needs to be set to a value that is close to 1 (e.g., 0.95), so that the current patches would not change the global mean and standard deviation too much. Once the training is completed, we use the estimated μ_{gB} and σ_{gB} when generating fake A. Similarly, we use μ_{gA} and σ_{gA} while generating fake B.

Algorithm 1: Estimation of the global μ and σ values for the embeddings of both domains.

output: global mean μ_{gA}, μ_{gB} and global standard deviation σ_{gA}, σ_{gB} values for the embeddings of both domains.

$\mu_{gA} \leftarrow 0$, $\sigma_{gA} \leftarrow 0$
$\mu_{gB} \leftarrow 0$, $\sigma_{gB} \leftarrow 0$
d_rate $\leftarrow 0.95$; `// decay rate parameter`
foreach *training iteration* **do**
  sample a patch from A, compute μ_A and σ_A of its embedding
  sample a patch from B, compute μ_B and σ_B of its embedding
  train SemI2I by minimizing Eqs. 3.19 and 3.20
  `/* update the global estimations as: */`
  $\mu_{gA} \leftarrow \text{d\_rate} \times \mu_{gA} + (1 - \text{d\_rate}) \times \mu_A$
  $\sigma_{gA} \leftarrow \text{d\_rate} \times \sigma_{gA} + (1 - \text{d\_rate}) \times \sigma_A$
  $\mu_{gB} \leftarrow \text{d\_rate} \times \mu_{gB} + (1 - \text{d\_rate}) \times \mu_B$
  $\sigma_{gB} \leftarrow \text{d\_rate} \times \sigma_{gB} + (1 - \text{d\_rate}) \times \sigma_B$
end

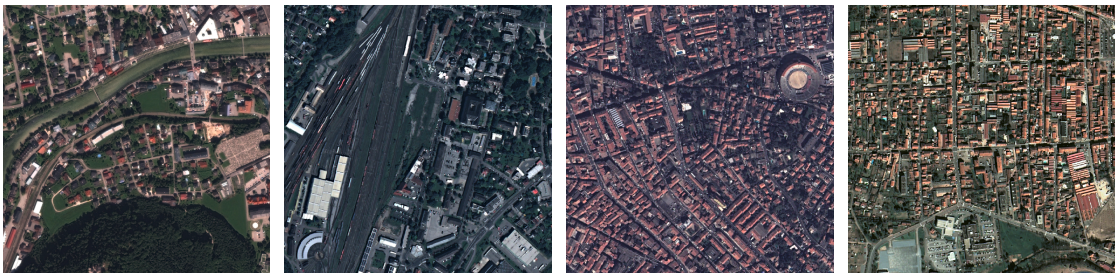
3.7 Experiments

3.7.1 Data Set

We conduct our experiments over Pléiades images collected over four cities: Bad Ischl and Villach from Austria, Béziers and Roanne from France. Fig. 3.5 depicts a close-up from each city. The images have been converted to 8-bit, and their spatial resolution has been reduced to 1 m by the data set providers. The images in the data set contain red, green, and blue channels. The full annotations for building, road, and tree classes have been provided. We split the cities into two pairs, where the first pair consists of Bad Ischl and Villach, and the second pair comprises Béziers and Roanne. To make the

Table 3.1: The data set.

City	# of patches	Area (km^2)	Class frequency (%)		
			building	road	tree
Bad Ischl	457	27.71	5.51	6.03	35.38
Villach	749	43.59	9.26	10.63	19.91
Béziers	407	25.75	19.09	17.62	10.91
Roanne	384	25.84	18.44	8.33	14.78



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

Figure 3.5: Close-ups from the four cities used in the experiments. (a) Bad Ischl, (b) Villach, (c) Béziers, (d) Roanne. The first pair is composed of Bad Ischl and Villach. Béziers and Roanne constitute the second pair.

experimental setup suitable for the unsupervised domain adaptation problem, when we split the cities into pairs, we pay attention that radiometry of both cities in each pair is as different as possible, and the objects belonging to the same class (e.g., building) have similar structural characteristics. For instance, buildings in Béziers and Roanne are densely grouped and have mostly rectangular shape, whereas buildings in Bad Ischl and Villach are more sparsely distributed and mostly square-like shaped. In both pairs, we first use one of the cities as source and the other one as target. We then switch the source and the target cities.

In the pre-processing step, we split each satellite image into 256×256 training patches with an overlap of 32 pixels between neighboring patches. Table 3.1 reports for each city the number of patches, the total area covered, and the class frequencies. For the quantitative performance assessment, we use Intersection over Union (IoU) [42] as the evaluation metric.

3.7.2 Methods Used for Comparison

We compare our segmentation framework with the following approaches:

- **U-net** [144]: We simply train a U-net from the source city and segment the target city without performing any type of domain adaptation techniques.
- **CycleGAN** [198]: In this methodology, there are two generators, one is used

to learn a mapping from the source domain to the target domain, and the other one learns the reverse mapping. The methodology requires that after a domain is mapped to another one, applying a reverse mapping must reconstruct the real domain. This constraint is enforced by minimizing L1 norm between the real and the reconstructed domains.

- **UNIT** [104]: Let us assume that both domains are denoted by X and Y, and x and y correspond to patches sampled from the domains. The generator of UNIT has two encoders E_x and E_y and two decoders G_x and G_y. The encoders are used to embed X and Y to a common space. The fake images are generated by $G_x(E_y(Y))$ and $G_y(F_x(X))$. In an ideal transformation, X and $G_x(E_y(Y))$, and Y and $G_y(E_x(X))$ must have similar statistics.

- **MUNIT** [78]: It decomposes the images from both domains into content and style codes. To generate fake images, the content code of one domain and the style code of another domain are combined via AdaIN [77].

- **DRIT** [96]: Methodologically DRIT is almost the same as MUNIT. The only difference is that the content code of one domain and the style code of another domain are combined by concatenating them.

- **Histogram matching** [60]: For each color channel, we match the histogram of the source domain with the histogram of the target domain to correct the spectral shift between the images.

- **Gray-world** [24]: It is one of the color constancy algorithms [3]. This algorithm assumes that the average color of the image should be natural gray, and any deviation from gray is caused by the illuminant. This assumption is used to remove the effect of the illuminant. We use gray-world algorithm to standardize both domains.

- **AdaptSegNet single** [173]: It aims at training a domain invariant network that performs well in segmenting both domains. To do this, the classifier generates predicted maps for both domains, and the discriminator forces the predicted maps for the target domain to look like the predictions for the source domain. In the original paper, DeepLab v2 [32] is used as the classifier. However, Atrous Spatial Pyramid Pooling (ASPP) in this network reduces the segmentation performance on satellite images significantly, especially when the image contains objects covering a small area. Hence, we remove ASPP from the network and directly up-sample the final classification layer.

- **AdaptSegNet multiple** [173]: In the same paper, in addition to aligning the final predictions for both domains, the experimental results with 2 classification layers and 2 discriminators are also presented. We compare our method with this strategy as well.

To make a fair comparison between our approaches (i.e., ColorMapGAN and SemI2I) and CycleGAN, UNIT, MUNIT, DRIT, histogram matching, and gray-world, we replace

the step 2 in our segmentation framework (see Fig. 3.1) with these algorithms. For gray-world, we also modify the step 4. Instead of segmenting the real target city, we segment the target city in which the illuminant effect is removed.

3.7.3 Training Details

In the first step of the framework, we train a U-net with Adam optimizer, where the learning rate is 0.0001, exponential decay rate for the first and the second moment estimates are 0.9 and 0.999. In each iteration, we randomly sample a batch of 32 training patches. We apply online data augmentation with random horizontal/vertical flips and 0/90/180/270 degrees of rotations. We use sigmoid cross entropy as the loss function and ignore the background class while computing the loss. We optimize the network for 2,500 iterations.

When training both ColorMapGAN and SemI2I, we randomly sample only one patch from each city. We use Adam optimizer [90] to update the generators and the discriminators. Since the generator of ColorMapGAN is architecturally much simpler than its discriminator, we prefer to optimize it with a larger learning rate. For the generator, the learning rate is 0.0005, whereas we set it to 0.0001 for the discriminator. We train ColorMapGAN for 8,000 iterations, since we verify by visual inspection that visually appealing results are obtained for this number of iterations. We train SemI2I for 25 epochs, where the number of iterations in each epoch is the minimum of the number of patches from each domain. We optimize SemI2I with Adam optimizer. We set the learning rate to 0.0002 in the first 15 epochs, and compute it in the rest of the epochs as:

$$\mathrm{LR} = 0.0002 \times \frac{\mathrm{num\_epochs} - \mathrm{epoch\_no}}{\mathrm{num\_epochs} - \mathrm{decay\_epoch}}, \tag{3.21}$$

where LR, num_epochs, epoch_no are the current learning rate, the total number of epochs, and current epoch no. decay_epoch stands for the epoch, which is set to 15 in our experiments, where the learning rate is started to be reduced. We set $\lambda_1, \lambda_2, \lambda_3$ and λ_4 in Eqs. 3.19 and 3.20 to 10, 10, 10, and 1, respectively. We have found these values empirically.

In the third stage of our framework, we fine-tune the initially trained U-net for 2,500 iterations using the generated fake source city as well as the ground-truth for the real source city. When generating a fake source city with the compared I2I approaches explained in Sec. 3.7.2 and when generating a map for the target city via AdaptSegNet, we use the default hyper-parameters described in the papers.

3.7.4 Results

Tables 3.2 and 3.3 report IoU values for each method in two city pairs. Figs. 3.6 to 3.9 show a close-up from each city, the corresponding ground-truth, and the predictions by each compared method. To provide as reliable as possible results, we repeat the step 3 in our framework 20 times for each method that generates fake data, and report the average IoU values in the tables. Re-running the step 3 of the framework 20 times is

Table 3.2: IoU scores for the target cities in pair 1.

Method		Source: Bad Ischl, Target: Villach				Source: Villach, Target: Bad Ischl			
		Building	**Road**	**Tree**	**Overall**	**Building**	**Road**	**Tree**	**Overall**
U-net [144]		23.61	0.91	40.53	21.68	5.84	0.24	0.50	2.19
AdaptSegNet Single [173]		6.01	4.37	10.43	6.94	3.06	2.71	10.23	5.33
AdaptSegNet Multi [173]		24.59	9.02	56.08	29.86	14.26	4.46	24.66	14.46
Our proposed framework with	CycleGAN [198]	43.03	28.96	**68.86**	46.95	43.62	38.69	71.68	51.33
	UNIT [104]	30.86	15.84	63.00	36.57	19.29	36.83	35.57	30.56
	MUNIT [78]	0.02	1.38	47.23	16.21	6.20	0.13	0.05	2.13
	DRIT [96]	0.01	3.96	8.72	4.23	0.00	10.19	0.01	3.40
	Gray-world [24]	25.19	26.43	56.15	35.92	29.55	24.80	46.41	33.58
	Hist. matching [60]	24.95	29.34	61.59	38.63	6.45	0.92	1.28	2.88
	ColorMapGAN	**48.47**	37.82	58.92	48.40	49.16	41.75	59.84	50.25
	SemI2I	47.79	**38.22**	68.76	**51.59**	**53.38**	**47.59**	**79.17**	**60.05**

Table 3.3: IoU scores for the target cities in pair 2.

Method		Source: Béziers, Target: Roanne				Source: Roanne, Target: Béziers			
		Building	**Road**	**Tree**	**Overall**	**Building**	**Road**	**Tree**	**Overall**
U-net		26.13	11.16	7.79	15.03	19.85	0.00	0.00	6.62
AdaptSegNet Single [173]		6.61	11.05	3.37	7.01	11.07	4.19	3.71	6.32
AdaptSegNet Multi [173]		22.42	5.87	17.84	15.37	24.27	10.88	10.45	15.20
Our proposed framework with	CycleGAN [198]	18.19	24.28	0.19	14.22	9.92	16.00	0.54	8.82
	UNIT [104]	41.99	38.47	0.39	26.95	29.99	39.19	3.11	24.10
	MUNIT [78]	10.17	1.66	0.31	4.05	7.49	0.84	0.13	2.82
	DRIT [96]	42.16	41.77	1.18	28.37	25.73	36.54	0.68	20.98
	Gray-world [24]	51.47	40.42	18.25	36.71	14.61	31.32	21.99	22.64
	Hist. matching [60]	18.64	9.87	4.81	11.11	20.63	0.02	0.00	6.88
	ColorMapGAN	**55.60**	44.66	28.39	42.88	47.12	35.18	21.91	34.74
	SemI2I	53.48	**45.05**	**36.40**	**44.98**	**50.97**	**44.27**	**35.15**	**43.46**

feasible, since we fine-tune the classifier for only 750 iterations in this step. However, AdaptSegNet tries to adapt the classifier to the target city directly. Hence, we need to train a classifier from scratch every time when we repeat the experiment. For this reason, for AdaptSegNet Single and Multi, we show the average IoU values for only 3 runs.

Figs. 3.10 to 3.13 depict close-ups from each real source and fake source images generated by CycleGAN [198], UNIT [104], MUNIT [78], DRIT [96], gray-world [24], histogram matching [60], ColorMapGAN, and SemI2I. From the images, we can clearly see that MUNIT and DRIT spoil the semantic identity of the images completely. For instance, the structures indicated by yellow and green rectangles in the real images are either replaced by other objects or distorted in the fake images. In some cases, UNIT seems to generate better results. For instance, the fake cities generated by UNIT in the second pair are semantically relatively consistent with the real source images, and style-wise similar to the target cities. On the other hand, the fake cities in the first pair have plenty of artificial objects that do not exist in the real source cities. For UNIT, MUNIT, and DRIT, since the fake images do not match with the ground-truth of the real source images, the network learns wrong information in the step 3 of our

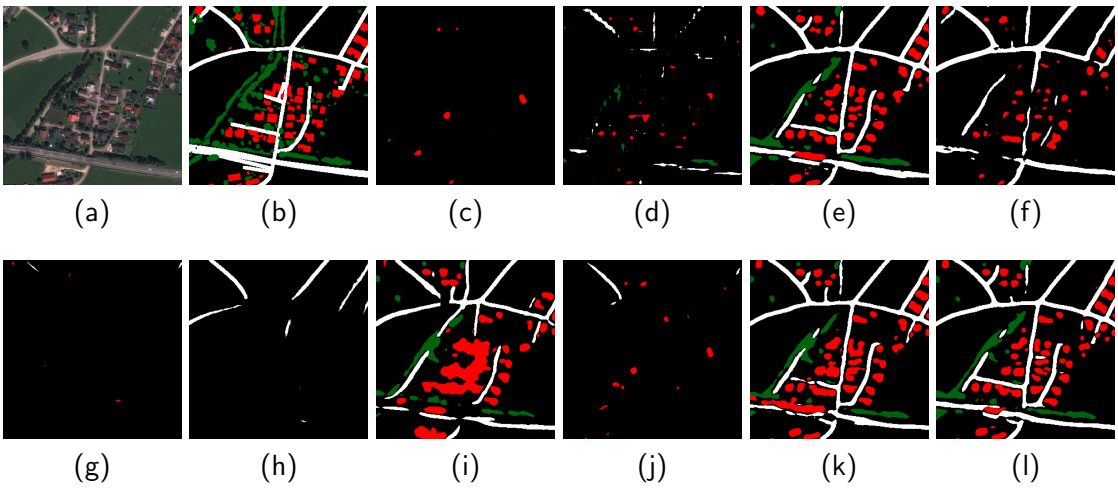


Figure 3.6: Bad Ischl, its ground-truth, and the predictions. (a) Bad Ischl, (b) ground-truth, (c) predictions by U-net [144], (d) by AdaptSegNet multi [173], by our framework with (e) CycleGAN [198], (f) UNIT [104], (g) MUNIT [78], (h) DRIT [96], (i) gray-world [24], (j) histogram matching [60], (k) ColorMapGAN, (l) SemI2I. Red, white, and green pixels represent building, road, and three classes, respectively. The pixels in black do not belong to a class.

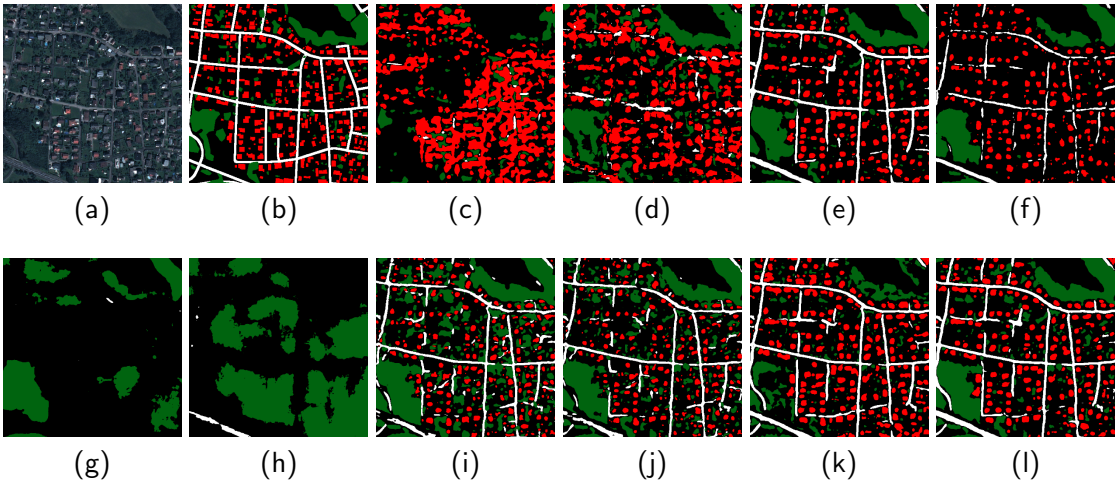


Figure 3.7: Villach, its ground-truth, and the predictions. (a) Villach, (b) ground-truth, (c) predictions by U-net [144], (d) by AdaptSegNet multi [173], by our framework with (e) CycleGAN [198], (f) UNIT [104], (g) MUNIT [78], (h) DRIT [96], (i) gray-world [24], (j) histogram matching [60], (k) ColorMapGAN, (l) SemI2I. Red, white, and green pixels represent building, road, and three classes, respectively. The pixels in black do not belong to a class.

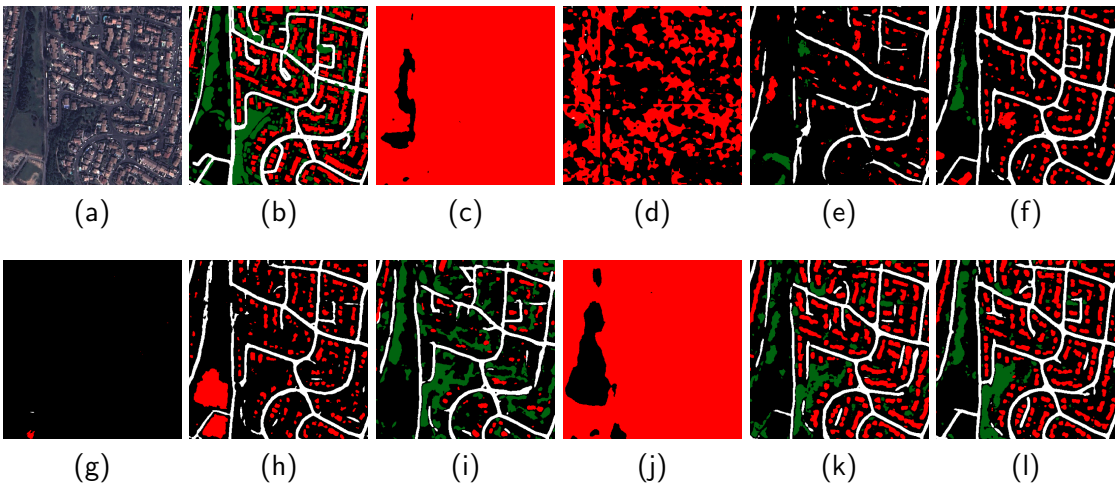


Figure 3.8: Béziers, its ground-truth, and the predictions. (a) Béziers, (b) ground-truth, (c) predictions by U-net [144], (d) by AdaptSegNet multi [173], by our framework with (e) CycleGAN [198], (f) UNIT [104], (g) MUNIT [78], (h) DRIT [96], (i) gray-world [24], (j) histogram matching [60], (k) ColorMapGAN, (l) SemI2I. Red, white, and green pixels represent building, road, and three classes, respectively. The pixels in black do not belong to a class.

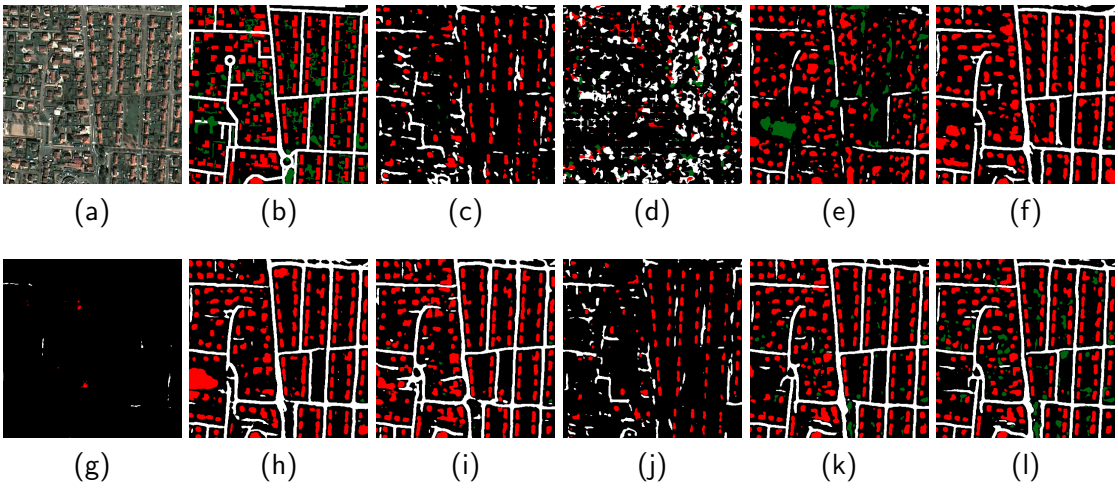


Figure 3.9: Roanne, its ground-truth, and the predictions. (a) Roanne, (b) ground-truth, (c) predictions by U-net [144], (d) by AdaptSegNet multi [173], by our framework with (e) CycleGAN [198], (f) UNIT [104], (g) MUNIT [78], (h) DRIT [96], (i) gray-world [24], (j) histogram matching [60], (k) ColorMapGAN, (l) SemI2I. Red, white, and green pixels represent building, road, and three classes, respectively. The pixels in black do not belong to a class.

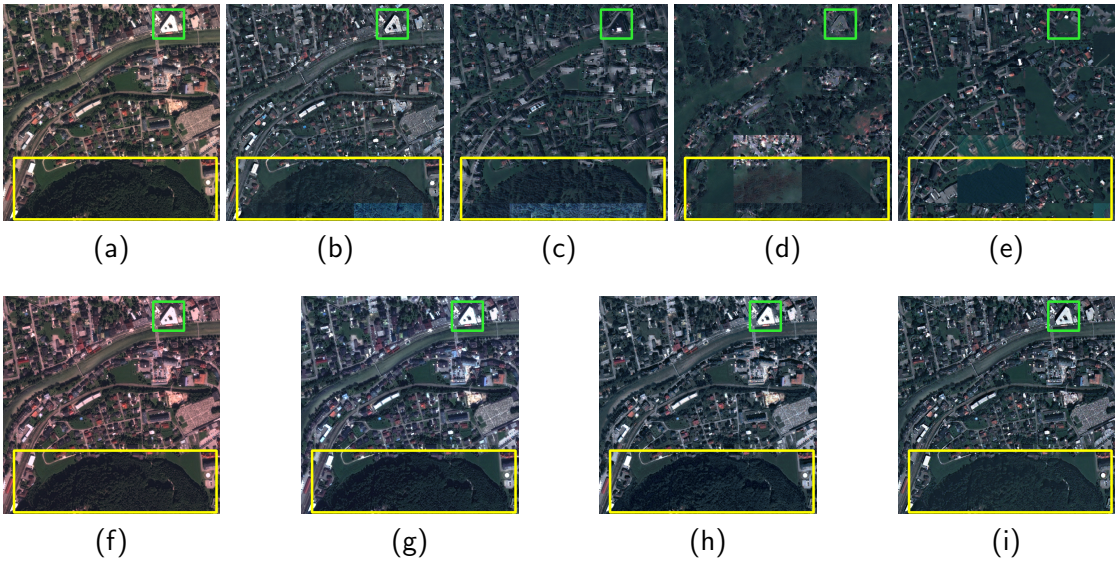


Figure 3.10: Real Bad Ischl and fake images from Bad Ischl to segment Villach. (a) Real Bad Ischl, fake Bad Ischl by (b) CycleGAN [198], (c) UNIT [104], (d) MUNIT [78], (e) DRIT [96], (f) gray-world [24], (g) histogram matching [60], (h) ColorMapGAN, (i) SemI2I.

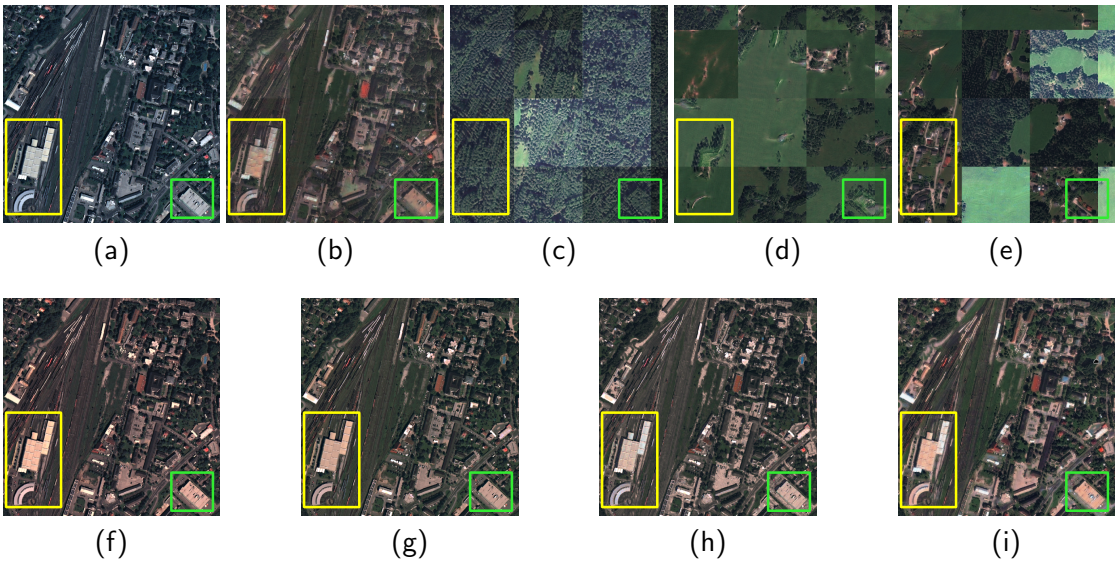


Figure 3.11: Real Villach and fake images from Villach to segment Bad Ischl. (a) Real Villach, fake Villach by (b) CycleGAN [198], (c) UNIT [104], (d) MUNIT [78], (e) DRIT [96], (f) gray-world [24], (g) histogram matching [60], (h) ColorMapGAN, (i) SemI2I.

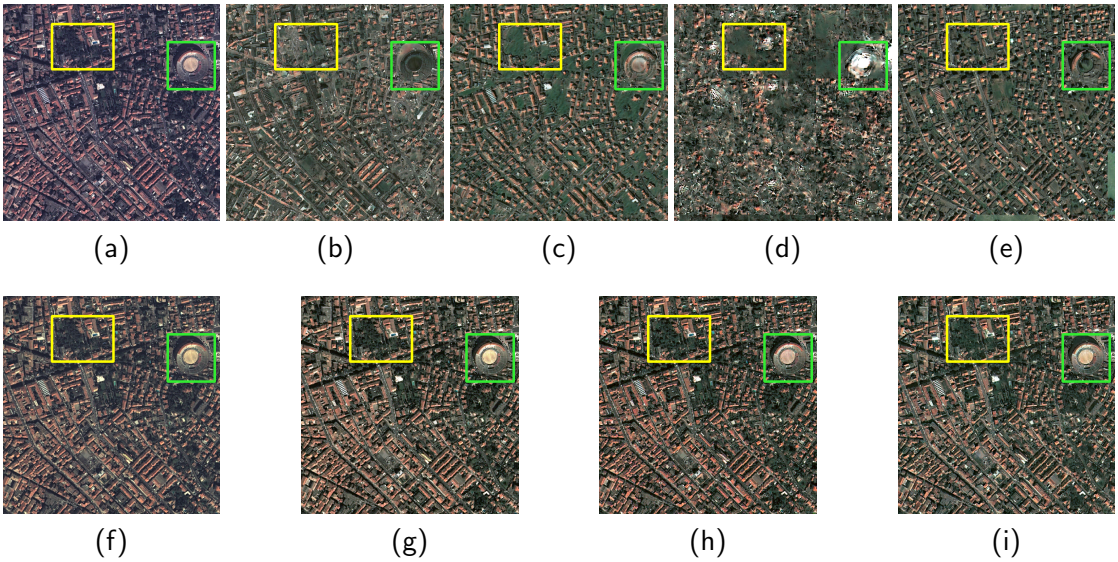


Figure 3.12: Real Béziers and fake images from Béziers to segment Roanne. (a) Real Béziers, fake Béziers by (b) CycleGAN [198], (c) UNIT [104], (d) MUNIT [78], (e) DRIT [96], (f) gray-world [24], (g) histogram matching [60], (h) ColorMapGAN, (i) SemI2I.

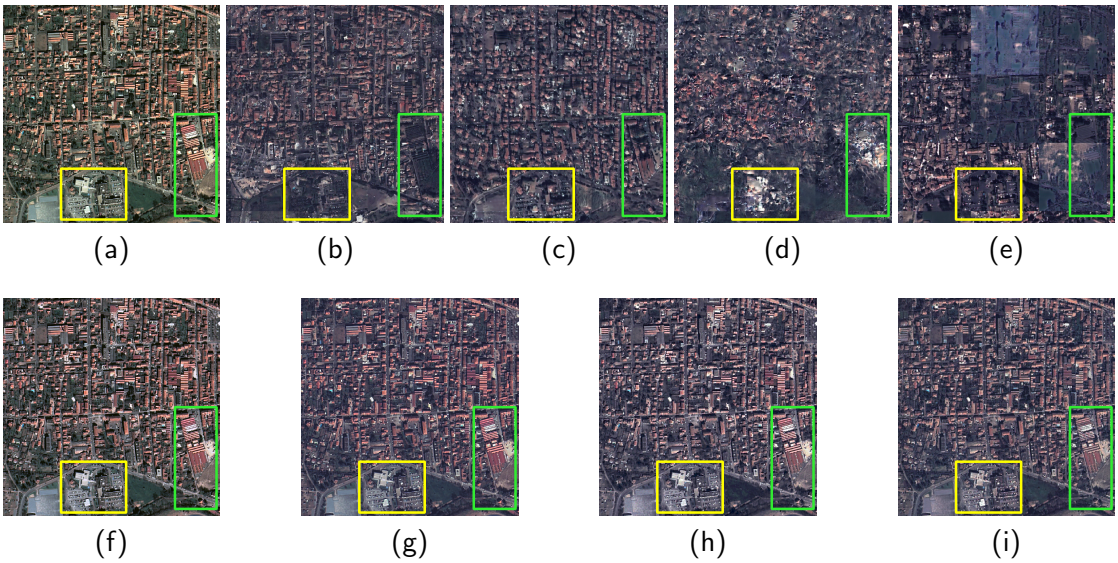


Figure 3.13: Real Roanne and fake images from Roanne to segment Béziers. (a) Real Roanne, fake Roanne by (b) CycleGAN [198], (c) UNIT [104], (d) MUNIT [78], (e) DRIT [96], (f) gray-world [24], (g) histogram matching [60], (h) ColorMapGAN, (i) SemI2I.

framework. Another problem with these approaches is that we observe tiling effect in the fake images. Especially in the fake images from Villach, the transition between the patches is not continuous. The reason is that since it is impossible to fit the large satellite images to GPU directly when generating fake images, we need to generate fake data from each patch and combine them to get the entire fake city. However, these approaches mostly generate irrelevant output for the neighboring patches. As a result, the proposed framework with these methods perform poorly on the target city, as confirmed by Tables 3.2 and 3.3.

As can be seen in Figs. 3.10 to 3.13, the spectral difference between the source and the target images can be reduced by standardizing them using gray-world algorithm. However, between the fake images in each city pair, we still observe a spectral shift; it is not completely corrected. As a consequence, the performance of gray-world algorithm is mostly better than UNIT, MUNIT, and DRIT, but it is not as good as desired.

The network architecture used in AdaptSegNet single is very deep; therefore, aligning only the outputs of the final classification layer for the source and the target images does not yield a good performance. However, if the alignment is performed in multiple layers, a better performance could be obtained, especially when the objects of interests cover a large area such as forests. For instance, most of the trees in Villach are located inside forests areas. For tree class in this city, IoU value for AdaptSegNet multi is 56.08%, which is slightly lower than the performance of our framework with ColorMapGAN. However, the performance of AdaptSegNet is not satisfactory in segmenting small objects such as building and thin objects like roads.

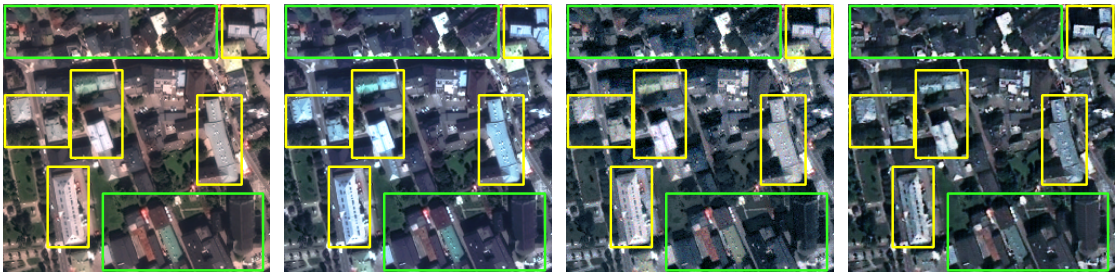
At the first sight, histogram matching seems to be working well; semantic structures of the training city are well preserved in the fake source city, and the style of the target city is perfectly transferred to the fake source city. However, it is noticeable in Tables 3.2 and 3.3 that the quantitative results for this approach are quite poor in most cases. Besides, the fake cities generated by CycleGAN, ColorMapGAN, and SemI2I in the first pair look similar. However, still there exists a large gap between the performances of the framework with CycleGAN, ColorMapGAN, or SemI2I. For example, when segmenting Villach, for road class, the IoU of CycleGAN is around 9% lower than that of ColorMapGAN. Similarly, ColorMapGAN outperforms CycleGAN by 6% for building class, when segmenting Bad Ischl. On the other hand, CycleGAN outperforms ColorMapGAN for tree class. SemI2I is the best performer in most cases. To better understand the reasons for aforementioned performance differences, the comparisons between histogram matching, CycleGAN, ColorMapGAN, and SemI2I need further analysis.

Detailed comparisons with CycleGAN: First of all, it is worth noting that the performance of CycleGAN is unstable. Our framework with CycleGAN performs unsatisfactorily on the second pair because of its semantically inconsistent outputs with the real source images. As highlighted by yellow rectangles in Figs. 3.12 and 3.13, CycleGAN removes some objects that exist in the real source cities. There are several reasons why it performs worse than ColorMapGAN and SemI2I in the first pair for building and road classes. Firstly, the resolution of its output is lower than the resolution of

Figure 3.14: Real and fake images from Villach. (a) Real Villach, fake images by (b) CycleGAN [198], (c) ColorMapGAN, (d) SemI2I.

the real source city and the outputs of ColorMapGAN and SemI2I. Fig. 3.14 depicts a closeup from Villach and the corresponding fake images generated by CycleGAN, ColorMapGAN, and SemI2I. The resolution difference between the fake images can easily be noticed in the outlined areas by yellow rectangles. Learning from blurrier data obviously deteriorates the performance. Secondly, the output of CycleGAN has some artifacts as shown in the same figure by a green rectangle. We do not observe such artifacts in the outputs of ColorMapGAN and SemI2I. Finally, since we generate fake cities patch by patch because of memory constraints, there exists a spectral difference between some of the neighboring patches in the fake images generated by CycleGAN (see bottom-right corner of the yellow rectangle in Fig. 3.10). This difference leads the network to exhibit a lower performance. ColorMapGAN does not have this problem, because it maps each color to another one. Irrespective of their locations, the pixels with the same color in the source images will have the same color in the fake source images. Therefore, the neighboring patches are spectrally consistent, and there is no tiling effect between them. We do not notice such tiling effect in the outputs of SemI2I either, since we use the estimated global μ and σ values computed via the proposed Alg. 1 when performing style transfer. One drawback of ColorMapGAN is that it seems to be slightly smoothing out trees (see the cyan rectangle in Fig. 3.14). This is probably why our framework with CycleGAN or SemI2I outperforms the framework with ColorMapGAN in the first pair for tree class.

Detailed comparisons with histogram matching: The main problem of histogram matching is that it does not take into account the contextual information, it only tries to match the histogram of the whole source city with the histogram of the whole target city. On the other hand, the discriminator of GANs based methods extracts high level features from the output of the generator and the target city to decide which one is real and which one is fake. In other words, the generator generates a fake source city in a way that its high level features align with the high level features extracted from the target city. For this reason, the proposed framework with ColorMapGAN or SemI2I yields substantially improved results. As highlighted in Fig. 3.15 by yellow rectangles,



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

Figure 3.15: Real and fake images from Villach. (a) Real Villach, fake images by (b) histogram matching [60], (c) ColorMapGAN, (d) SemI2I.

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

Figure 3.16: Real and fake images from Villach. (a) Real Villach, fake images by (b) histogram matching [60], (c) ColorMapGAN, (d) SemI2I.

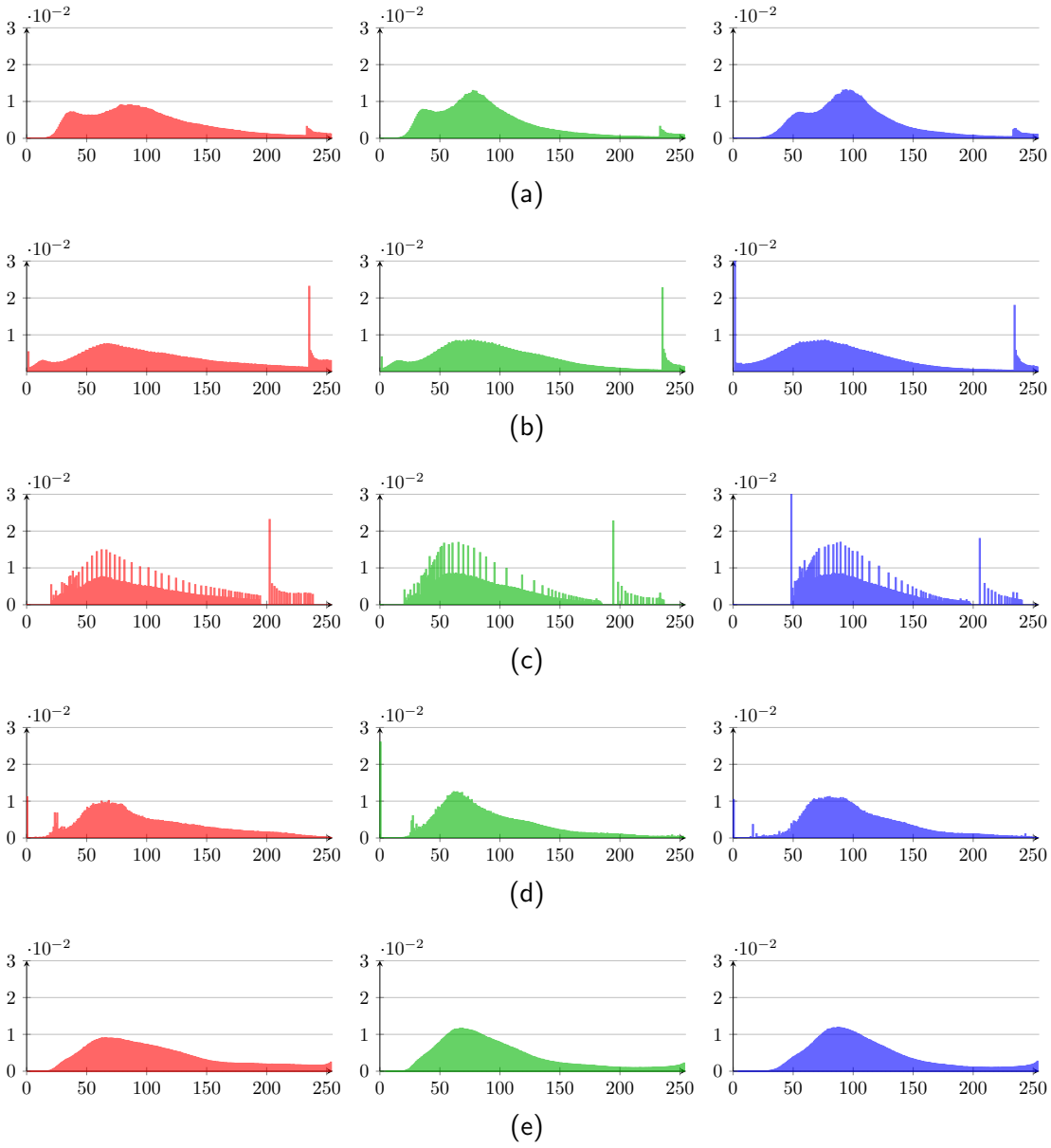


Figure 3.17: Color histograms of the pixels belonging to building class in Béziers and Roanne. Histograms for (a) Béziers, (b) Roanne, fake images from Roanne generated by (c) histogram matching, (d) ColorMapGAN, (e) SemI2I.

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

Figure 3.18: Failure cases for ColorMapGAN. (a-c) Real images from Villach, (b-d) noisy fake images generated by ColorMapGAN.

when generating fake Bad Ischl, histogram matching converts some gray rooftops to cyan ones, whereas ColorMapGAN and SemI2I keep them gray. Similarly, in the same figure, we observe that the buildings highlighted by green rectangles have dark violet rooftops in the output of histogram matching, and black rooftops in the images generated by ColorMapGAN and SemI2I. In Fig. 3.5, we can see that there is no building having a cyan or dark violet rooftop in Villach, but there exists many buildings with gray or black rooftops. If the generator of ColorMapGAN or SemI2I generated cyan or dark violet colored rooftops, the discriminator would easily understand that these buildings were fake. For this reason, such buildings do not appear in the outputs of ColorMapGAN and SemI2I. Similarly, in the process of generating fake Roanne, histogram matching algorithm generates some reddish roads, as shown in Fig. 3.16 by a yellow rectangle. In the same figure, we see that ColorMapGAN and SemI2I output gray roads. Moreover, ColorMapGAN and SemI2I generate buildings having brownish colored rooftops, which are probably more representative for the buildings in Béziers than the buildings with red rooftops generated by histogram matching. Furthermore, in Fig. 3.17, we depict color histograms of the buildings in Roanne, in Béziers, and in the fake images from Roanne generated by histogram matching, ColorMapGAN, and SemI2I. Since Roanne and Béziers are two different cities and the number of pixels belonging to each class in both cities is different, we cannot expect the histograms of an ideal fake Roanne and Béziers to be exactly the same. However, we expect them to resemble each other. Although histogram matching algorithm tries to match the histogram of the source city with the histogram of the target city, there exists a large difference between the class-wise histograms of the fake source and the target cities. As can be seen in Fig. 3.17, there is a large deviation between some of the neighboring bins in the histograms. In contrast, color histograms of the buildings in the fake images from Roanne generated by ColorMapGAN and SemI2I are more similar to the histograms of the buildings in Béziers. For histogram matching, we observe the same issue for the other classes as well.

Detailed comparisons between ColorMapGAN and SemI2I: Tables 3.2 and 3.3 prove that the proposed ColorMapGAN and SemI2I significantly outperform the existing

Figure 3.19: Failure cases for SemI2I. (a-c) Real images from Villach, (b-d) imperfect fake images generated by SemI2I.

approaches. Both ColorMapGAN and SemI2I have their own strengths and weaknesses. The biggest advantage of ColorMapGAN is its extremely short training time mainly because of its architecturally simple generator, which performs only one element-wise matrix multiplication and one addition operations. With the hyper-parameters used in our experiments, it takes only 6.5 minutes to train ColorMapGAN on an Nvidia Geforce GTX1080 Ti GPU. On the other hand, the training time of SemI2I is 17 minutes. Note that the training of SemI2I is quite short as well due to its shallow generators consisting of only a few layers, but it is longer than that of ColorMapGAN. One of the disadvantages of ColorMapGAN is that it is not capable of mapping the same color to different colors. ColorMapGAN assigns exactly the same color to the pixels having the same red, green, and blue channel values irrespective of where the pixels are located. For instance, it cannot map the colors of a building and a road pixels with the same color to two different colors. The second disadvantage is that since it maps each individual color to another one, its output is noisy. We observe the noisy output especially on top of the structures having bright colors. Fig. 3.18 illustrates two failure cases, where ColorMapGAN adds noise on top of rooftops. However, it is worth mentioning that deliberately adding some noise to training data (e.g., Gaussian noise) is a common practice in real-world applications to make the classifiers robust to noisy data. Therefore, the second disadvantage of ColorMapGAN may not be regarded as a big drawback. As reported in Tables 3.2 and 3.3, SemI2I is the best performer for most of the classes in both experiments. Its weakness is that it generates unnatural objects in some cases. Fig. 3.19 depicts such a failure case, where building rooftops have an odd mix of colors.

3.7.5 Running Times

The proposed framework, ColorMapGAN, SemI2I, CycleGAN, UNIT, MUNIT, and DRIT were implemented in Tensorflow[1]. We conducted all the experiments on an Nvidia Geforce GTX1080 Ti GPU with 11 GB of RAM. Table 3.4 reports the training times of image-to-image translation based methods for 1 iteration. The table demonstrates that

[1]https://www.tensorflow.org

Table 3.4: Training times of I2I based methods.

Method	**Training time for 1 iteration (seconds)**
CycleGAN [198]	0.11
UNIT [104]	0.47
MUNIT [78]	0.45
DRIT [96]	0.29
ColorMapGAN	**0.05**
SemI2I	0.08

Table 3.5: Execution times of non-learning based methods.

City	**Execution time (seconds)**	
	Gray-world [24]	Histogram matching [60]
Bad Ischl	**1.46**	19.77
Villach	**1.89**	26.78
Béziers	**1.37**	18.05
Roanne	**1.24**	20.32

the training times of ColorMapGAN and SemI2I are shorter than the other learning based approaches. It is also notable that non-learning based approaches generate an output in a substantially shorter time, as confirmed by Table 3.5. The execution time of gray-world algorithm is almost instant, and histogram matching needs less than half a minute. However, the quality of the results for non-learning based methods is deficient.

3.8 Concluding Remarks

Although convolutional neural networks have been proven to be an effective tool to generate high quality maps from remote sensing images, their performance significantly deteriorates when there exists a large domain shift between source and target domains. Especially in remote sensing, such domain shift occurs quite often for various reasons such as atmospheric conditions and some differences in acquisition that change spectral characteristics of objects.

To increase the generalization abilities of classifiers, in this chapter, we presented a 4-step segmentation framework dealing with city-to-city domain adaptation problem. The crucial stage of our framework is the second step, where we generate a fake source city with the style of target city. We then use the fake source city as well as the ground-truth of the real source city to fine-tune the classifier trained in the first step our framework. One limitation of our framework is that it cannot be trained end-to-end. Each step of our framework needs to be run consecutively by passing the outputs of each step to the next one as inputs.

For the second stage of our segmentation framework, we introduced two novel methods, namely ColorMapGAN and SemI2I, which are able to generate a target stylized fake

source city that is semantically consistent with the real source city. In two experimental setups, we verified that the fake cities generated by ColorMapGAN and SemI2I allow the classifiers to significantly better generalize to a target city than the existing approaches. Furthermore, we discussed the advantages and disadvantages of both methods over each other in great detail.

We believe that there are rooms to further improve the outputs of ColorMapGAN and SemI2I. For example, to overcome the issue of noisy data by ColorMapGAN, one can consider adding a regularization term enforcing neighboring pixels to have similar colors. Another improvement could be to modify only the most significant four bits of each color in the source image. This approach may lead to not only generating less noisy outputs by also reducing the number of parameters as well as the training time. To improve SemI2I, a possible future direction might be separating content and style information of each domain, and combining the content of one domain with the style of another one. In the next two chapters, we present new methods inspired by this idea.

Chapter 4

Multi-source Domain Adaptation by Data Standardization

4.1 Problem Statement

In this chapter, we tackle multi-source domain adaptation problem for semantic segmentation of satellite images. We seek for developing a novel method that learns from multiple source domains and generates a precise map for a target domain, even when the data distributions of all the domains are significantly different. The inputs for the proposed approach are multiple satellite images collected from different geographic locations and their annotations. The desired output is a high quality map for the target city, in raster format.

4.2 Motivations

With the help of significant technological developments over the years, it has been possible to access massive amounts of remote sensing data. For example, the constellations of Pléiades satellites are able to capture large amounts of images with high spatial resolution over the globe.

As mentioned in the previous chapter, it is quite often that images collected in separate locations and at different times have significantly large data distribution, as a result of various atmospheric conditions and some differences in acquisition that change spectral characteristics of objects, intra-class variability, and differences in the spectral characteristics of the sensor. Although nowadays we have access to massive amounts of data, because manual annotation is extremely labor-intensive, the annotated data are scarce. Besides, the above-mentioned distribution differences cause machine learning models to generate unsatisfactory maps for images with different distributions than those of the training data. Therefore, it is crucial to develop new methods having strong generalization abilities despite the scarcity of the labeled training data.

In this context, there are many methods in the literature aiming at adapting a

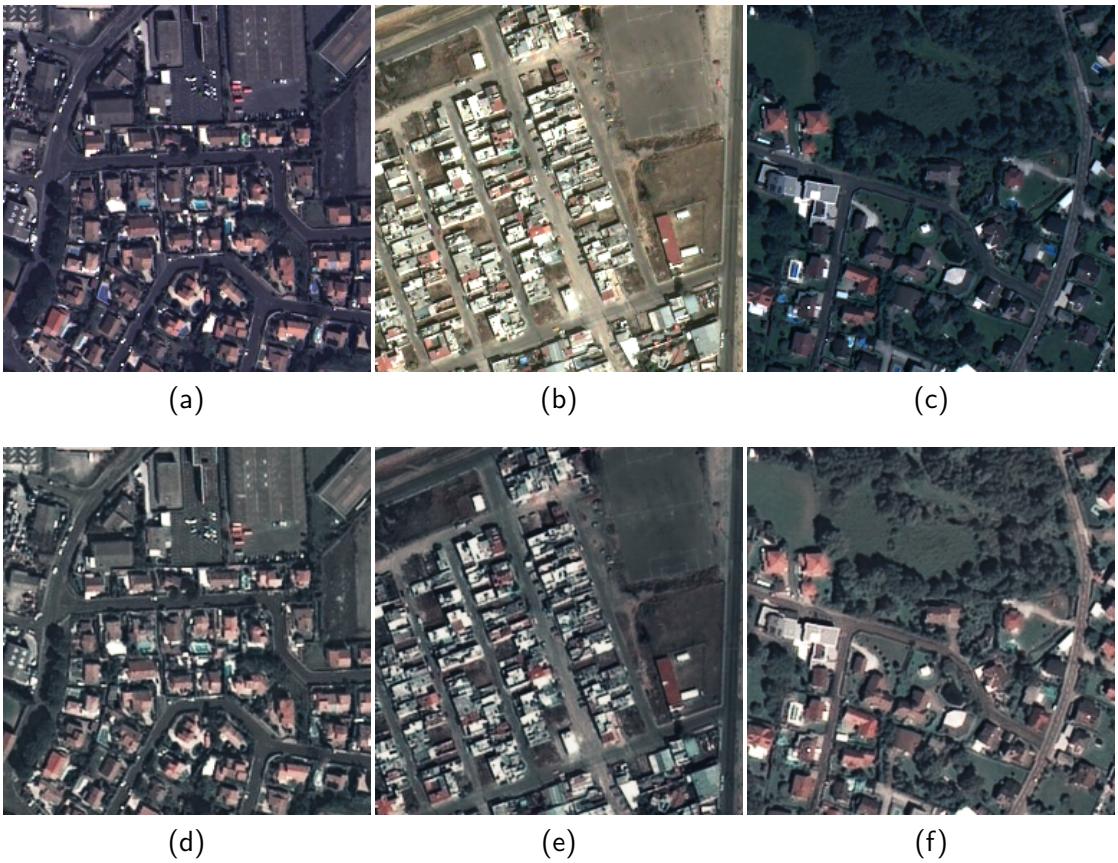


Figure 4.1: Real cities and their standardized versions.

classifier from one domain to another. In addition, in the previous chapter, we proposed two novel methods for city-to-city domain adaptation problem. However, these methods have limited practical real-world applications. Since nowadays huge volume of data are accessible, oftentimes one has multiple source domains with different data distributions. Hence, it is of paramount importance to develop new methods that can learn from multiple source domains and segment target domains well.

In our multi-source domain adaptation problem definition, we assume that each source and target domains have significantly different data distributions (see real data in the first row of Fig. 4.1). We aim at finding a common representation for all the domains by standardizing the samples belonging to each domain using GANs. As shown in Fig. 4.1, in a way, the standardized data could be considered as spectral interpolation across the domains. Adopting such a standardization strategy has two advantages. Firstly, in the training stage, it prevents the classifier from capturing the idiosyncrasies of each source domain. The classifier rather learns from the common representation. Secondly, since in the common representation the samples belonging to source domains and target domain have distributions close to each other, we expect the classifier trained

on the standardized source domains to segment well the standardized target domain.

Standardizing multiple domains using GANs raises several challenges. Firstly, when training GANs, one needs real data so that the generator can generate fake data with the distribution that is as close as possible to the distribution of the real data. However, in our case, the standardized data do not exist. In other words, we wish to generate data without showing samples drawn from a similar distribution. Secondly, all the standardized domains need to have similar data distributions. Otherwise, the advantages mentioned above would be lost. Thirdly, the standardized data and the real data themselves must be semantically consistent. For example, when generating the standardized data, the method should not replace some objects by others, add artificial objects, or remove some objects existing in the real data. Otherwise, the standardized data and the ground-truth for the real data would not match, and we could not train a model. Finally, the method should be efficient. If the number of networks and their structures are not kept as small as possible, depending on the number of domains, we could face issues in terms of memory occupation and computational time.

4.3 Related Work

4.3.1 Adapting the Classifier

Some of the methods that are based on adapting the classifier described in the previous chapter can be used for multi-source domain adaptation problem as well. The methods in this category aim at adapting the classifier to a target domain without modifying the data. For instance, methods performing multi-task learning, where one of the tasks is to train a classifier from the source domain via common supervised learning approaches, and the other task is to align the features extracted from both source and target domains by adversarial training [72,76,173], are suitable for this problem. Self learning methods [193, 200] can also be considered.

4.3.2 Adapting the Inputs

As mentioned in the previous chapter, the methods based on adapting the inputs usually aim at performing image-to-image translation (I2I) or style transfer to generate a fake source domain with a similar distribution to that of a target domain. For instance, CyCADA [71] uses the fake domain to train a classifier. The most straightforward approach for multi-source domain adaptation problem would be to perform I2I between each source and target domains to stylize all of the source domains as the target domain. However, this method is extremely cumbersome, because the training must be performed for each source domain and the target domain pair.

4.3.3 Multi-source Adaptation

Recently, specifically for multi-source adaptation, a few methods focusing on image classification have been proposed [130, 184, 194]. However, it may not be possible to

extend these approaches to semantic segmentation, as precisely structured output is required. To address the issue of multi-source adaptation for semantic segmentation, Zhao *et al.* have proposed MADAN [196], which is an extension of CyCADA, but it is extremely compute-intensive. JCPOT [139] investigates optimal transport for multi-source adaptation problem. Elshamli *et al.* have recently proposed a method consisting in patch based networks [50]. However, since the network architectures are not fully convolutional, the method may not be suitable to classes requiring high precision such as buildings and roads.

4.3.4 Data Standardization

In machine learning, one of the most commonly used data standardization approach is referred to as linear scaling to unit variance (a.k.a. z-score normalization) [4] and computed as:

$$x' = \frac{x - \mu}{\sigma}, \tag{4.1}$$

where x, μ, σ correspond to original data, mean value and standard deviation. In addition, histogram equalization [60] is also a common pre-processing step. However, these approaches do not take into account the contextual information and just follow certain heuristics. One may also think of applying color constancy algorithms [3] such as gray-world [24] and gamut [53] approaches. These algorithms assume that colors of the objects are highly affected by the color of the illuminant and try to remove this effect.

4.4 Contributions

In this chapter, we present novel StandardGAN, which deals with the challenges explained in Sec. 4.2 for standardization of multiple satellite images. After the images are standardized, we train a classifier on the standardized source images and we segment the standardized target image. The contributions of this chapter are three-fold:

- For the first time, we introduce the use of GANs in the context of data standardization. StandardGAN learns to take average of the styles of all the domains to generate standardized images.
- We present GANs that is able to generate data samples without providing it with data coming from the same or a similar distribution. Although no data with similar distributions to those of the standardized data are available, our GANs achieves to generate standardized data.
- Finally, we propose to apply this multi-source domain adaptation solution to the semantic segmentation of very high resolution satellite images collected over several geographic locations in the world.

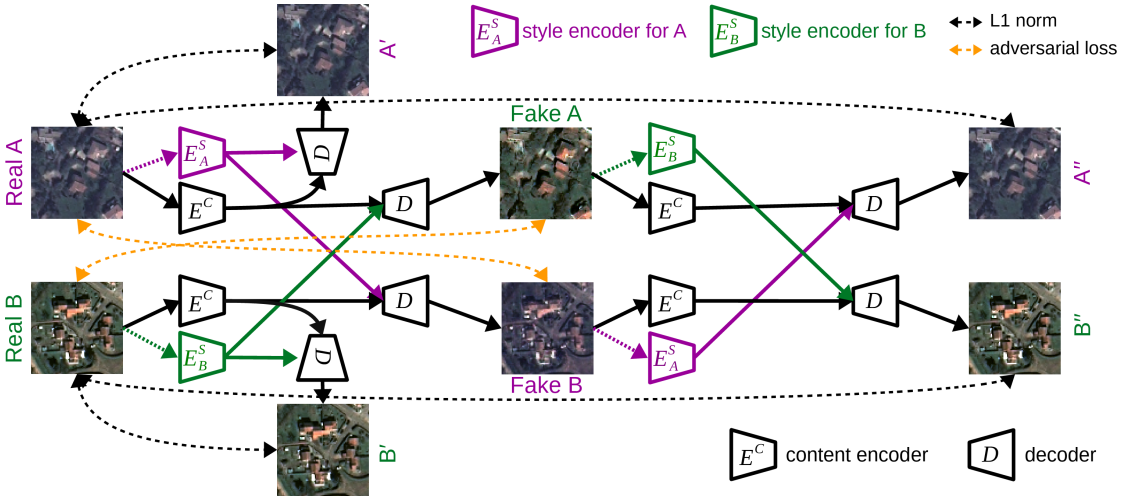


Figure 4.2: Style transfer between two cities. In this example, there are 2 style encoders, 1 content encoder, 1 decoder, and 1 discriminator.

4.5 Method

In this section, we first explain how to perform style transfer between two domains. We then describe how StandardGAN standardizes two domains. Finally, we detail how we extend StandardGAN to multi-domain case.

StandardGAN consists of one content encoder, one decoder, one discriminator and n style encoders, where n denotes the number of domains. Fig. 4.2 illustrates the generator to perform style transfer between two domains. The discriminator performs multi-task learning as in StarGAN [36] by adding an auxiliary classifier on top of the discriminator of CycleGAN [198]. The first task allows the fake source and the target domains to have as similar data distributions as possible, whereas the other task helps the discriminator to understand between which fake and real data it is discriminating. We provide detailed explanations for both tasks in the following sub-section.

4.5.1 Style Transfer Between Two Domains

The goal of style transfer is to generate a fake A with the style of B and a fake B having a similar data distribution as real A. To perform style transfer, we use two types of encoders. One is domain agnostic content encoder, and the other one is domain specific style encoder. The content encoder is used to map the data into a common space, irrespective of which domain the data come from. On the other hand, the style encoder helps the decoder to generate output with the style of its specific domain. We use adaptive instance normalization (AdaIN) [77] to combine the content of A with the style of B (or vice versa). AdaIN is defined as:

$$\text{AdaIN}(x, \gamma, \beta) = \gamma \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta, \tag{4.2}$$

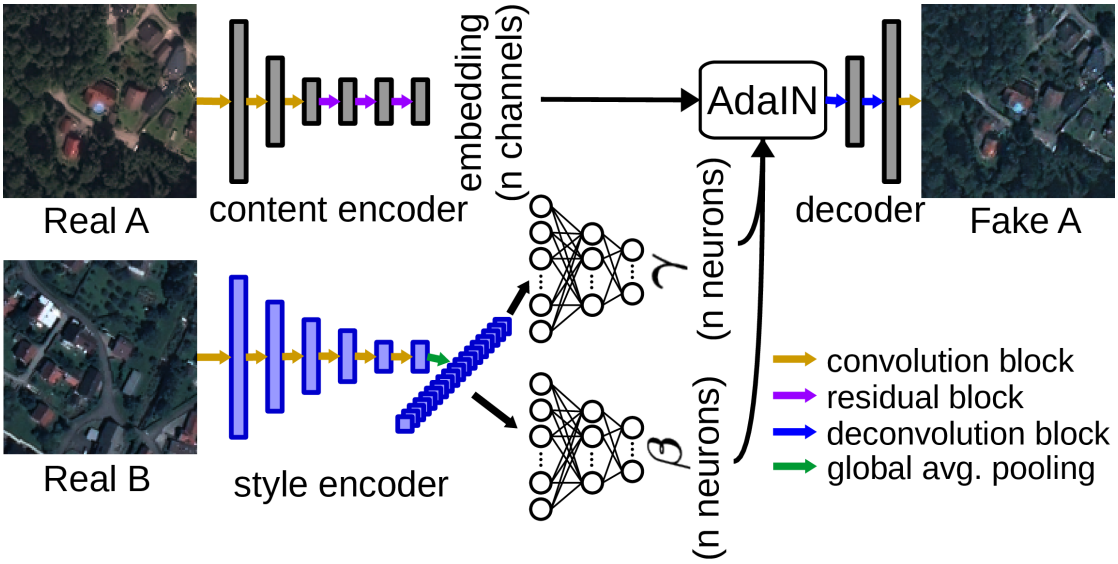


Figure 4.3: Combining the content of one city with the style of another city.

where x is the activation of the content encoder's final convolutional layer, and γ and β correspond to the parameters that are learned by the style encoder. As can be seen in Eq. 4.2, γ and β are used to scale and shift the activation, which results in changing the style of the output. After the activation is normalized by AdaIN, as depicted by Fig. 4.3, it is fed to the decoder to generate the fake data.

In order to force real A and fake B, and real B and fake A to have as similar data distributions as possible, we compute and minimize an adversarial loss between them. We use the adversarial loss functions described in LSGAN [116]. The discriminator adversarial loss between real A and fake B (or real B and fake A) is defined as:

$$\mathcal{L}_{adv\_D}(X,Y) = \mathbb{E}_x[(D_{adv}(x)-1)^2] \; + \; \mathbb{E}_y[(D_{adv}(G(y)))^2] \tag{4.3}$$

where $\mathbb{E}$ denotes the expected value, G and D_{adv} stand for the generator and the adversarial output of the discriminator (the first task), and x and y correspond image patches sampled from domains X and Y. The generator adversarial loss is computed as:

$$\mathcal{L}_{adv\_G}(X,Y) = \mathbb{E}_y[(D_{adv}(y)-1)^2]. \tag{4.4}$$

The overall generator adversarial loss $\mathcal{L}_{adv\_G}$ and the discriminator adversarial loss $\mathcal{L}_{adv\_D}$ are calculated as:

$$\mathcal{L}_{adv\_D} = \mathcal{L}_{adv\_D}(A,B) + \mathcal{L}_{adv\_D}(B,A) \tag{4.5}$$

and

$$\mathcal{L}_{adv\_G} = \mathcal{L}_{adv\_G}(A,B) + \mathcal{L}_{adv\_G}(B,A). \tag{4.6}$$

To force real A and fake B, and real B and fake A to have similar styles, normally, we need two discriminators. One is used for discriminating between real A and fake B, and

the other is responsible for distinguishing between real B and fake A. However, as mentioned in Sec. 4.2, we want to keep the number of networks as small as possible to easily extend StandardGAN to multi-domain case. In order to use only one discriminator, we adopt the approach described in StarGAN [36]. Let us assume that X is the source and Y is the target domain. We suppose that the labels of X and Y are indicated by c_s and c_t (e.g., $c\_s = 0$ and $c\_t = 1$), and the image patch sampled from X is denoted by x. On top of the discriminator, we add a classifier. Both the discriminator and the generator have a role on this classifier. On the one hand, the discriminator wants the classifier to predict the label of X correctly. On the other hand, the generator tries to generate a fake X in a way that the classifier predicts it as Y. The classification loss for the discriminator is defined as:

$$\mathcal{L}_{cls\_D}(X) = \mathbb{E}[-\log D_{cls}(c\_s \mid x)], \tag{4.7}$$

where $D_{cls}(c\_s \mid x)$ denotes the probability distribution over domain labels generated by D. By minimizing this function, D learns from which domain x come. The classification loss for the generator is computed as:

$$\mathcal{L}_{cls\_G}(X) = \mathbb{E}[-\log D_{cls}(c\_t \mid G(x))]. \tag{4.8}$$

The overall discriminator and the generator losses are computed as:

$$\mathcal{L}_{cls\_D} = \mathcal{L}_{cls\_D}(A) + \mathcal{L}_{cls\_D}(B) \tag{4.9}$$

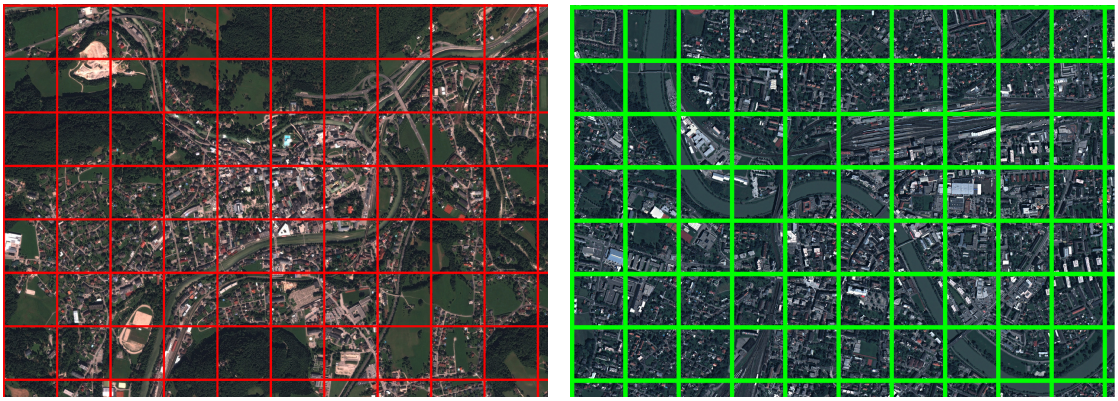
and

$$\mathcal{L}_{cls\_G} = \mathcal{L}_{cls\_G}(A) + \mathcal{L}_{cls\_G}(B). \tag{4.10}$$

In the training stage, minimizing Eqs. 4.9 and 4.10 allows the discriminator to understand whether it needs to distinguish between real A and fake B or between real B and fake A. As a result, the style transfer can be performed with only one discriminator. The classification loss is particularly useful when we extend StandardGAN to multi-domain adaptation case.

As mentioned in Sec. 4.2, it is crucial to perform the style transfer without spoiling the semantics of the real data. Otherwise, the fake data and the ground-truth for the real data would not overlap. Thus, they cannot be used to train a model. To preserve the semantic consistency, we propose several constraints and architectural designs. First of all, our decoder is architecturally quite simple. It consists of only one convolution and two deconvolution blocks (see Fig. 4.3). After scaling and shifting the content embedding of one domain with the AdaIN parameters learned by the style encoder from another domain, we directly decode the embedding, instead of adding further residual blocks. Moreover, we use the same constraints defined when explaining SemI2I in the previous chapter. As shown in Fig. 4.2, after we generate fake A with the style of B and fake B with the style of real A, we switch the styles once again to obtain A$''$ and B$''$. In an ideal case, A and A$''$, and B and B$''$ must be the same. Hence, we minimize the cross reconstruction loss $\mathcal{L}_{cross}$ that is defined as:

$$\mathcal{L}_{cross} = |\mathrm{A} - \mathrm{A}''| + |\mathrm{B} - \mathrm{B}''|. \tag{4.11}$$



(a) (b)

Figure 4.4: It is not possible to fit large satellite images into GPUs, they need to be processed patch by patch. (a) and (b) illustrate example patches from two cities.

Similarly, when we combine the content information of a domain with its own style information, we should be reconstructing itself (see A′ and B′ in Fig. 4.2). We also minimize the self reconstruction loss $\mathcal{L}_{self}$ computed as:

$$\mathcal{L}_{self} = |\mathrm{A} - \mathrm{A}'| + |\mathrm{B} - \mathrm{B}'|. \tag{4.12}$$

The overall generator loss is calculated as:

$$\mathcal{L}_G = \lambda_1 \mathcal{L}_{cross} + \lambda_2 \mathcal{L}_{self} + \lambda_3 \mathcal{L}_{cls\_G} + \lambda_4 \mathcal{L}_{adv\_G}, \tag{4.13}$$

where $\lambda_1, \lambda_2, \lambda_3$, and λ_4 denote the weights for the individual losses. The discriminator loss is defined as:

$$\mathcal{L}_D = \lambda_3 \mathcal{L}_{cls\_D} + \lambda_4 \mathcal{L}_{adv\_D}. \tag{4.14}$$

We minimize $\mathcal{L}_G$ and $\mathcal{L}_D$ simultaneously.

As can be seen in Fig. 4.3, to generate fake data, content encoder, decoder, and the AdaIN parameters learned by the style encoder of the other domain are required. The problem is that usually it is not possible to fit satellite images into GPUs because of their large number of pixels. Hence, when generating fake images, each image needs to be split into smaller patches as shown in Fig 4.4. The patches are then processed separately and combined again to obtain the whole standardized image. The issue here is that the style encoder produces different AdaIN parameters for each image patch depending on the context of the patch. For instance, we cannot expect patches from a forest and an industrial area to have similar parameters, because they have different styles. For each domain, to capture the global AdaIN parameters, we first initialize domain specific γ and β parameters with zeros. We then propose to update them in each training iteration as:

$$p = 0.95 \times p + 0.05 \times p\_\text{current}, \tag{4.15}$$

where p is the global domain specific AdaIN parameter (i.e., γ or β) and $p\_current$ is the parameter from the current training patch. After a sufficiently long training process, Eq. 4.15 estimates the global AdaIN parameters for each domain. These estimations can then be used to stylize each patch from one domain as the other domain in the test stage.

4.5.2 StandardGAN for Image Standardization

As mentioned previously, the domain agnostic content encoder learns to map domains into a common space. To generate target stylized fake source data, the content embedding extracted by the content encoder from the source domain is normalized with the global AdaIN parameters of the target domain. The normalized embedding is then given to the decoder to generate the fake data. We have discovered that instead of normalizing the embedding with the AdaIN parameters for one of the domains, if we normalize it with the arithmetic average of the global AdaIN parameters of both domains, StandardGAN learns to generate standardized data. The standardization process for two domains is depicted in Fig. 4.5. As shown in the figure, real A and real B have considerably different data distributions. On the other hand, standardized A and standardized B look quite similar, and their data distributions are somewhere between the data distributions of real A and real B.

To standardize multiple domains, we propose Alg. 2. In multi-domain case, c_s and c_t in Eqs. 4.9 and 4.10 can range between 0 and n - 1, where n is the number of domains. As shown in Fig. 4.6, we perform adaptation between each pair of domains. We then take the average of the global AdaIN parameters of each domain and use the average to normalize the embeddings extracted by the content encoder from all the domains. We finally decode the normalized embeddings via the decoder to generate the standardized data.

4.6 Experiments

4.6.1 Data Set & Experimental Setup

In our experiments, we use Pléiades images captured from 5 cities in Austria, 2 cities in France, and 1 city in Liechtenstein. The spectral channels consist of red, green, and blue bands. The spatial resolution has been reduced to 1 m by the data set providers. The annotations for building, road, and tree classes are provided. Table 4.1 records, for each city, the name of the city, the percentage of pixels belonging to each class as well as the total covered area.

We have two experimental setups. In the first experiment, we use the images from Salzburg Stadt, Villach, Lienz, and Sankt Pölten for training and the image from Bad Ischl for test. In the second experiment, we choose Salzburg Stadt, Villach, Bourges, and Lille as the training cities and Vaduz as the test city. In the first experiment, we want to observe how well our method generalize to a new city from the same country. On the other hand, the goal of the second experiment is to investigate the generalization abilities

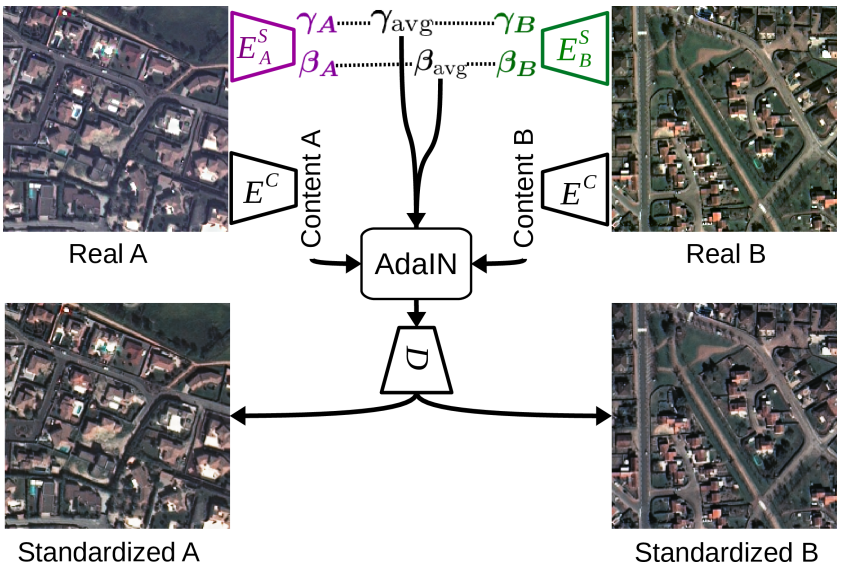


Figure 4.5: Standardizing two domains. Dashed lines correspond to arithmetic average.

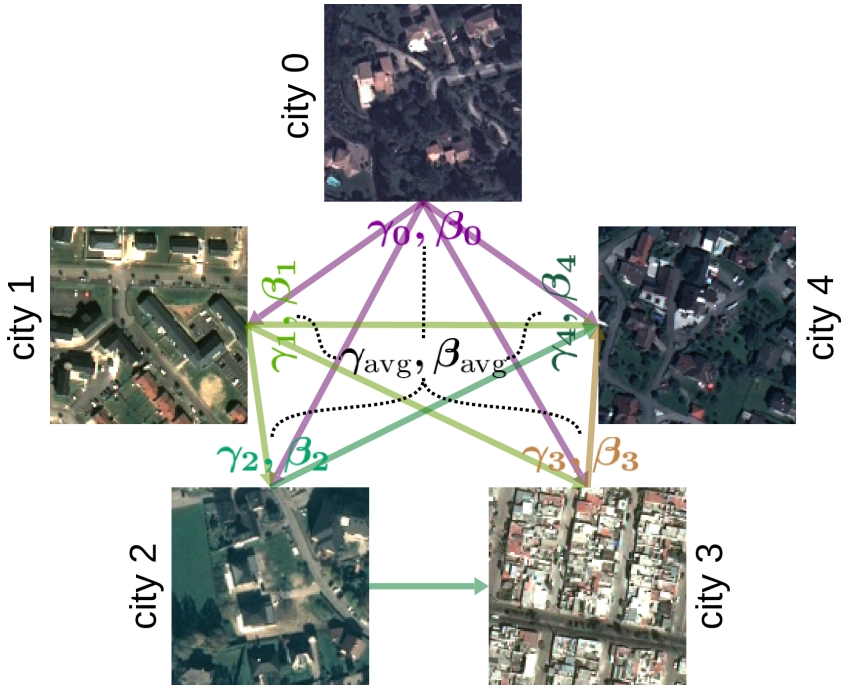


Figure 4.6: Standardizing multiple domains. Solid arrows represent adaptation between two domains. Dashed lines correspond to arithmetic average. γ_{avg} and β_{avg} are used for standardization.

Algorithm 2: The pseudocode for StandardGAN to standardize multiple domains.

create 1 content encoder, 1 decoder, and 1 discriminator
foreach *domain* **do**
  initialize domain specific AdaIN parameters with zeros
  create a domain specific style encoder
end
foreach *training iteration* **do**
  $\mathcal{L}_G \leftarrow 0$, $\mathcal{L}_D \leftarrow 0$; `// G and D losses`
  for $i \leftarrow 0$ **to** *(# of domains - 1)* **do**
    for $j \leftarrow (i+1)$ **to** *(# of domains - 1)* **do**
      $\mathcal{L}_{G\_current} \leftarrow$ G loss between domains $i\&j$ (Eq. 4.13)
      $\mathcal{L}_{D\_current} \leftarrow$ D loss between domains $i\&j$ (Eq. 4.14)
      $\mathcal{L}_G \leftarrow \mathcal{L}_G + \mathcal{L}_{G\_current}$
      $\mathcal{L}_D \leftarrow \mathcal{L}_D + \mathcal{L}_{D\_current}$
    end
  end
  backprop. $\mathcal{L}_G$ and $\mathcal{L}_D$, $\mathcal{L}_G \leftarrow 0$, $\mathcal{L}_D \leftarrow 0$
  foreach *domain* **do**
    update domain specific AdaIN parameters via Eq. 4.15
  end
  average AdaIN parameters $\leftarrow$ arithmetic average of domain specific AdaIN parameters
end

Table 4.1: The data set.

City (Country)	**Class percentages (%)**			**Area**
	Building	**Road**	**Tree**	**(km^2)**
Bad Ischl (AT)	5.51	6.0	35.38	27.71
Salzburg Stadt (AT)	9.44	8.69	23.88	134.71
Villach (AT)	9.26	10.63	19.91	43.59
Lienz (AT)	6.96	8.16	15.37	28.38
Sankt Pölten (AT)	6.68	6.39	25.13	87.17
Bourges (FR)	9.81	10.52	14.83	72.20
Lille (FR)	18.36	12.71	15.40	117.58
Vaduz (LI)	3.57	4.30	33.69	96.08

Table 4.2: Training time of StandardGAN for both experiments.

GPU	Experiment No	# of patches	Training time (h:m:s)
Nvidia Tesla V100 SXM2	1	5712	1:41:17
	2	8226	2:45:29

of our approach when training and test data come from different countries. Let us also remark that, as confirmed by Table 4.1, classes in the test cities (i.e., Bad Ischl and Vaduz) are highly imbalanced, which makes the domain adaptation problem even more difficult. For example, in both cases, the number of pixels labeled as tree is significantly larger than the number of pixels labeled as buildings and roads.

4.6.2 Training Details

In the pre-processing step, we split all the cities into 256×256 patches with 32 pixels of overlap. We set $\lambda_1, \lambda_2, \lambda_3$, and λ_4 in Eqs. 4.13 and 4.14 to 10, 10, 1, and 1, respectively. We train StandardGAN for 20 epochs with Adam optimizer, where the initial learning rate is 0.0002, the exponential decay rates for the moment estimates are 0.5 and 0.999, respectively. In each training iteration of StandardGAN, we randomly sample 1 patch from each domain. After the 10^{th} epoch, we progressively reduce the learning rate in each epoch as:

$$\text{learn. rate} = \text{init\_lr} \times \frac{\text{num\_epochs} - \text{epoch\_no}}{\text{num\_epochs} - \text{decay\_epoch}}, \tag{4.16}$$

where init_lr, num_epochs, epoch_no, and decay_epoch correspond to the initial learning rate (0.0002 in our case), the total number of epochs (we set it to 20), the current epoch no, and the epoch no in which we start reducing the learning rate (we determine it as 10). Table 4.2 reports the total number of training patches in both experiments and the training time of StandardGAN. We first standardize all the data. We then train a model on the standardized source data and classify the standardized target data. We compare our approach with the other standardization algorithms described in Sec. 4.3, namely gray-world [24], histogram equalization [60], and Z-score normalization [4]. We use U-net [144] as the classifier. We also provide the experimental results for naive U-net without applying any domain adaptation methods. For each comparison, we train a U-net for 35 epochs via Adam optimizer with the learning rate of 0.0001 and the exponential decays rates of 0.9 and 0.999. In each training iteration of U-net, we use a mini-batch of 32 randomly sampled patches. We perform online data augmentation with random rotations and flips.

4.6.3 Results

In Fig. 4.7, we depict close-ups from the cities used in the first experiment and the fake data generated by StandardGAN. Note that to train a model, we do not use the target stylized source data, we use only the standardized data that are highlighted by red

Figure 4.7: Real data used in the first experiment and the outputs generated by StandardGAN. Left column: the real data. Matrix on the right: The standardized data are highlighted by red bounding boxes. The rest of the cells depict the i^{th} domain with the style of j^{th} domain. The domain ids are indicated inside parentheses.

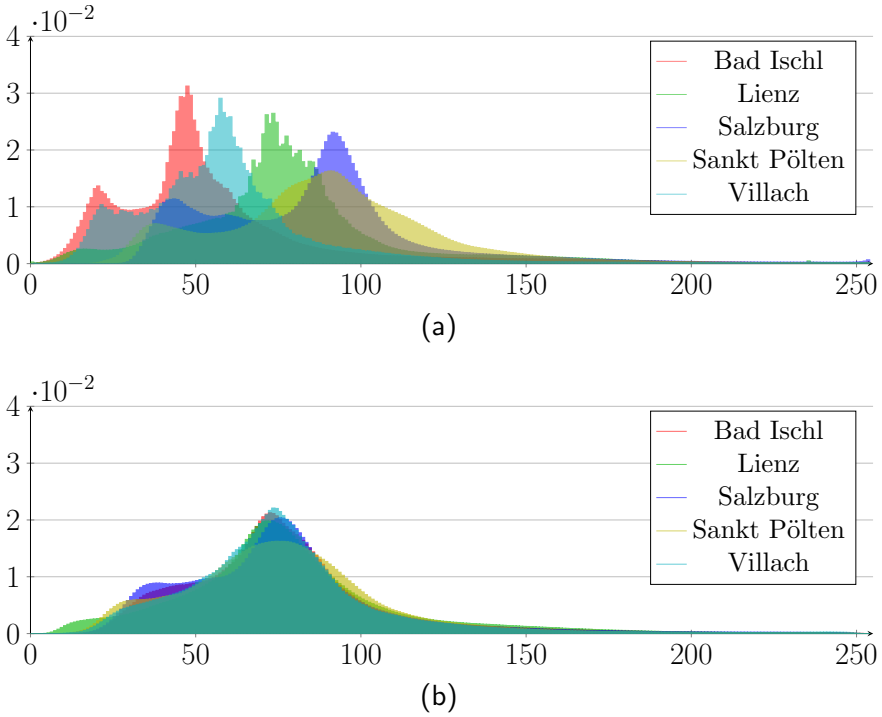


Figure 4.8: Histograms for green band of the cities used in the first experiment. (a) Before standardization, (b) After standardization.

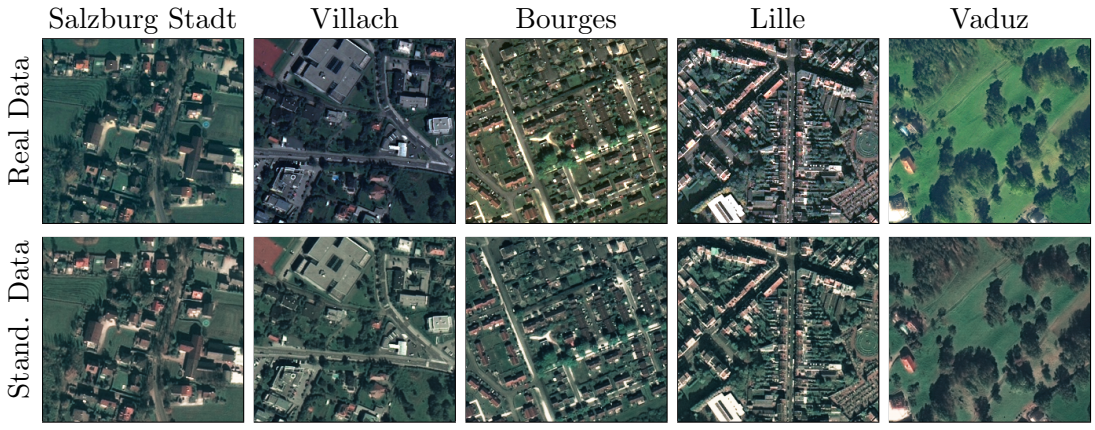


Figure 4.9: Real cities used in the second experiment, and the standardized data generated by StandardGAN.

Table 4.3: IoU scores for Bad Ischl (the first experiment).

	Method	building	road	tree	Overall
	U-net	45.36	18.81	**82.43**	48.87
U-net on data by	Gray-world	49.39	42.25	66.31	52.65
	Histogram Equalization	45.33	39.07	73.03	52.48
	Z-score normalization	51.22	46.56	77.62	58.47
	StandardGAN	**56.41**	**50.26**	80.59	**62.42**

Table 4.4: IoU scores for Vaduz (the second experiment).

	Method	building	road	tree	Overall
	U-net	29.83	26.42	46.25	34.16
U-net on data by	Gray-world	27.95	31.13	36.65	31.91
	Histogram Equalization	21.21	19.19	51.93	30.78
	Z-score normalization	29.94	29.87	41.98	33.93
	StandardGAN	**54.86**	**42.43**	**63.09**	**53.46**

bounding boxes in the figure. The style transfer between each domain is the prior step to the standardization. We can clearly observe that there exists a substantial difference between the data distributions of the real data, whereas the standardized data look similar. Moreover, Fig. 4.8 confirms that color histograms of the standardized data are considerably closer to each other than those of the real data. Fig. 4.9 shows closeups from the cities in the second experiment and their standardized versions by StandardGAN. The standardized and the real data for Salzburg Stadt and Lille seem quite similar. The reason is the data distributions of these two cities are already somewhere between the distributions of all five cities. However, the radiometry of Villach, Bourges, and Vaduz significantly changes after the standardization process. Besides, all the standardized data have similar data distributions.

Tables 4.3 and 4.4 report the intersection over union (IoU) [42] values on both experiments. The training data acquired over a single country are usually more representative for a city from the same country than a city from another country. For this reason, the quantitative results for the first experiment are generally higher. Besides, in some cases, the representativeness of the samples belonging to different classes may vary. For instance, in the first experiment, the traditional U-net already exhibits a relatively good performance for tree class, as the tree samples from the source domains represent well the samples in the target data. For this class, the performance of our method is slightly worse. It is probably because of some artifacts generated by the proposed GANs architecture when standardizing the domains. On the other hand, for the other classes, our approach achieves a better performance than all other methods. In the second experiment, unlike the first one, none of the class samples in the source domains are representative for the target domain. Hence, the performance of U-net is poor. In addition, the common heuristic based pre-processing methods do not help improving the

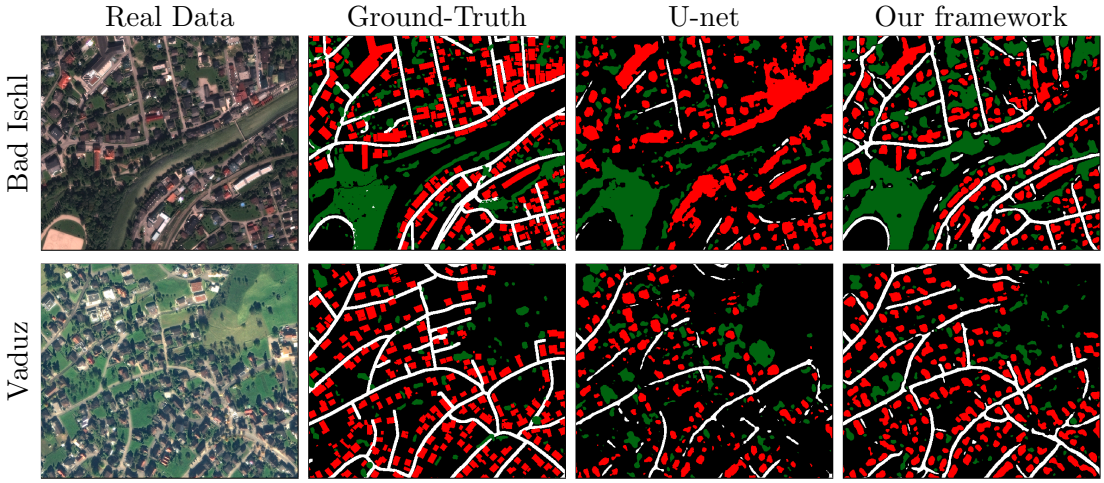


Figure 4.10: Comparison between the traditional U-net and our framework. Red, green, and white pixels represent building, road, and tree classes, respectively. The pixels in black do not belong to any class.

results. However, the StandardGAN better allow the classifier to generalize completely different geographic locations. Fig. 4.10 illustrates the improvement of our framework against the naive U-net in terms of predicted maps.

4.7 Concluding Remarks

The recent technological developments in satellite images have enabled us to collect huge volume of data. The availability of such massive data motivates us to propose novel solutions that can learn from multiple satellite images and generate high quality maps for unlabeled data, even when there exists large distribution differences between images.

In remote sensing, while there is a rich literature in terms of domain adaptation methods aiming at adapting the classifiers from one domain to another, these approaches are not feasible for large-scale semantic segmentation problem. Although multi-source domain adaptation problem has many practical real-world applications, it has not been investigated in detail until recently. To address this issue, in this chapter, we presented novel StandardGAN, which is a new pre-processing approach proposed with the purpose of standardizing multiple satellite images. In our experiments, we verified that the standardized data generated by StandardGAN enable the classifier to significantly better generalize to new Pléiades data. Note that StandardGAN has only one encoder, one discriminator, one decoder, and n style encoders. Although there are multiple style encoders, their architecture is fairly simple. Thus, it is feasible to use StandardGAN to standardize larger number of domains than the number of cities in our experiments.

We believe that StandardGAN is not only relevant for domain adaptation, it is suitable for other real-world applications as well. As mentioned Sec. 4.2, satellite images

collected in different times usually have different data distributions. Because of this reason, oftentimes, we observe radiometric differences between satellite images taken from neighboring locations. Image mosaicking aims at fixing such radiometric distortions to obtain one continuous satellite image [2]. As future work, one can consider applying StandardGAN to this problem. Another potential application is change detection, where similarities and differences between satellite images acquired over the same location are analyzed [108]. However, distribution difference between the images makes the methods proposed for this problem likely to fail. StandardGAN can be used as a pre-processing step for change detection problem as well.

As in our segmentation framework described in the previous chapter, the segmentation approach explained in this chapter cannot be trained end-to-end as well. One needs to standardize training and test images via StandardGAN as a pre-processing step to train a classifier in the second stage. Another drawback of StandardGAN is that every time when a new image is added, it is required to insert an additional style encoder. Although the style encoder is a shallow network, still the method is limited to a certain number of domains. In this next chapter, we present a method dealing with this issue.

Chapter 5

Multi-source, Multi-target, and Life-long Domain Adaptation

5.1 Problem Statement

In this chapter, we address unsupervised, multi-source, multi-target, and life-long domain adaptation problem. We aim at introducing a dense labeling method that can learn on multiple annotated source images and generate high quality maps for multiple target images under large distribution differences between the images. We also want the approach to be capable of adapting to progressively increasing data. In our problem definition, we assume that there might be one or more source and target images with greatly different data distributions. Moreover, we suppose that new labeled source and unlabeled target images can be added over the time. In such a setting, instead of training a domain adaptation method from scratch, we wish to devise an approach that can adapt the previously trained classifier to newly added source and target images, even when each image has a significantly different distribution.

5.2 Motivations

As mentioned in the previous chapter, the common single-source and single-target adaptation setting limits the proposed approaches to small-scale segmentation problems. The new generation of satellites with a short revisit time generate in routine massive amounts of remote sensing data. These data usually contain multiple domains with largely different data distributions and are relevant for many real-world applications. Therefore, to perform large-scale classification, an ideal unsupervised domain adaptation approach should be capable of learning from annotated multiple source domains and well segmenting unlabeled multiple target domains, even when the data distributions of all the domains are different. In addition, nowadays, the constellations of satellites collect images from the entire earth everyday. As a consequence, in many cases, one receives new annotated source domains and unlabeled target domains over time. Hence, it is

also crucial for the ideal approach to perform life-long adaptation over the continuously increasing data.

By performing unsupervised style transfer between two domains with GANs, it is possible to generate a fake source domain whose data distribution is as similar as possible to the distribution of a target domain [167, 168]. A possible solution for unsupervised, multi-source, multi target, and life-long domain adaptation problem would be learning to perform style transfer between all domain pairs. If this task can be accomplished, we can generate multiple fake domains from each domain, where each fake domain is representative for a different domain. We then can use the fake domains generated from the source domains to train a classifier. Training a classifier on such diversified data would allow the classifier to learn from the data that are representative for both the source and the target domains, instead of capturing the idiosyncrasies of each source domain.

However, the problem of style transfer between progressively growing multiple domains raises some challenges. Firstly, all of the fake domains generated from each source domain must be semantically consistent with the original domain. For example, if the method replaces some objects with others or adds artificial structures to the fake domains in the generation process, the fake domains and the ground-truth for the real domain would not match. Thus, we cannot train a classifier. Secondly, when we perform style transfer between domain pairs, the data distribution of the fake domain generated from one domain must be similar to the distribution of the other domain. Otherwise, the fake domain would not be representative for the other domain. Thirdly, the number of networks should not grow with the number of domains in order for the method to scale to many domains. For example, if the method requires using a style encoder for each domain as in MUNIT [78], DRIT [96], StarGAN [36], StarGAN v2 [37], and StandardGAN [79], it would be limited to a certain number of domains. In addition, it would be necessary to add new style encoders every time when we receive new data. Finally, once the training is completed, ability to perform style transfer between the domains on the fly is desired. Otherwise, it would be required to store all the fake domains on a disk.

5.3 Related Work

5.3.1 Single-Source and Multi-target Adaptation

As in multi-source adaptation problem, it is possible to solve multi-target adaptation problem by executing an I2I method to generate fake source domains with similar distributions to those of the target domains. These fake domains can then be used to train a classifier. However, such a solution is sub-optimal as it requires applying an I2I approach multiple times. Specifically for multi-target adaptation problem, an information theoretic approach [58] and a model parameter adaptation framework [188] have been proposed. However, in their problem definitions, these methods assume that there exists only one source domain. In our case, on the other hand, there might be more than one source domain. In addition, these methods have been proposed for image categorization

problem. Hence, it may not be possible to use them for semantic segmentation.

5.3.2 Multi-Source and Multi-target Adaptation

The methods based on data standardization described in the previous section, namely Z-score normalization [4], histogram equalization [60], color constancy algorithms such as gray-world [24] and gamut [53] are relevant for multi-source and multi-target adaptation. Our experiments in the previous chapter proved that StandardGAN outperforms these approaches. However, the main limitation of StandardGAN is that there needs to be a different style encoder for each satellite image. As a consequence, the method is limited to a certain number of images.

5.3.3 Life-long Adaptation

The remote sensing literature describes methods based on semi-supervised active learning, where an annotator labels some portions of the received domain to be included in the training data. In the approach proposed by Persello and Bruzzone [133], the classifier iteratively learns from a small number of newly added samples from the target domain and removes some of the source samples whose distribution does not fit with the distribution of the target domain. The SVM based method presented by Matasci *et al.* also follows a similar approach [118]. Another kernel based active learning approach is introduced by Deng *et al.* [46]. The usage of neural networks for active learning has been studied as well [57]. However, we seek for an unsupervised method instead, as automation is the key for practical real-world applications.

5.4 Contributions

In this chapter, we introduce a new approach dealing with the challenges mentioned in Sec. 5.2. Our contributions are as follows:

- We propose an approach that can perform style transfer between multiple domains with only one encoder, one decoder, and one discriminator unlike MUNIT [78], DRIT [96], StarGAN [36], StarGAN v2 [37], and StandardGAN [79] that rely upon a different style encoder for each domain. In addition, since the number of networks is constant irrespective of the number of domains, our approach can do style transfer across multiple domains even when new domains are progressively added.
- We introduce novel DAugNet consisting of a data augmentor and a classifier. The data augmentor, which is a shallow network, can stylize each domain as another one. Due to its simple architecture, it allows online data diversification rather than storing all the fake data on a disk and loading them when training a classifier. In each training iteration of DAugNet, the data augmentor diversifies a batch of training patches by stylizing each patch as a randomly selected domain before passing the batch to the classifier.

- We validate DAugNet on semantic segmentation of Pléiades images captured over multiple cities from three European countries. Our data comprise images with either red, green, blue bands or near-infrared, red, green bands. In three extensive experiments, we investigate how DAugNet performs on single-source and single-target, multi-source and multi-target, and life-long adaptation problems. We also conduct five ablation studies to further analyze the properties of our solution.

5.5 Method

Our training pipeline consists of two stages. In the first stage, we learn how to stylize each domain like any other using only one encoder, one decoder, and one discriminator. Note that this stage is completely unsupervised. Therefore, we use both annotated source and unlabeled target domains during training. A subset of the networks used in this stage forms a data augmentor. In the second stage, we train DAugNet that is composed of the data augmentor and a classifier. In each training iteration of DAugNet, the data augmentor is frozen and diversifies a batch of patches sampled from source domains by stylizing each patch in the batch as a randomly selected domain. The diversified batch is then given to the classifier. The randomly selected domain can be chosen among both source and target domains. Hence, when training DAugNet, the classifier learns from the data that are representative for all of the source and the target domains. As a result, it is expected that DAugNet would perform better than a classifier trained only on real source domains.

In the remainder of this section, we first describe how we concurrently switch the styles of two domains. We then detail how we perform style transfer between multiple domains. We finally explain the architecture of DAugNet and its overall training pipeline for multi-source, multi-target, and life-long domain adaptation.

5.5.1 Style Transfer Between Two Domains

To perform style transfer between two domains, we use one encoder, one decoder, and one multi-task discriminator. Throughout this sub-section, we refer to one of the domains as A and the other one as B for the sake of simplicity.

Before the training starts, we initially generate a unique and constant style code for each domain. We denote by $S_A = \{\gamma_A, \beta_A\}$ and $S_B = \{\gamma_B, \beta_B\}$ the style codes of A and B. We initialize each parameter of S_A and S_B (i.e., γ and β) with 128 values drawn from the uniform distribution that ranges between 0 and 1. To generate a fake domain stylized as the other one, we first extract an embedding from the domain using the encoder. We then normalize the embedding via adaptive instance normalization (AdaIN) [77] that is defined as:

$$\text{AdaIN}(x, \gamma, \beta) = \gamma \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta, \tag{5.1}$$

where x denotes the final activation of the encoder from one domain, and γ and β correspond to the style code parameters of the other domain. We finally decode the

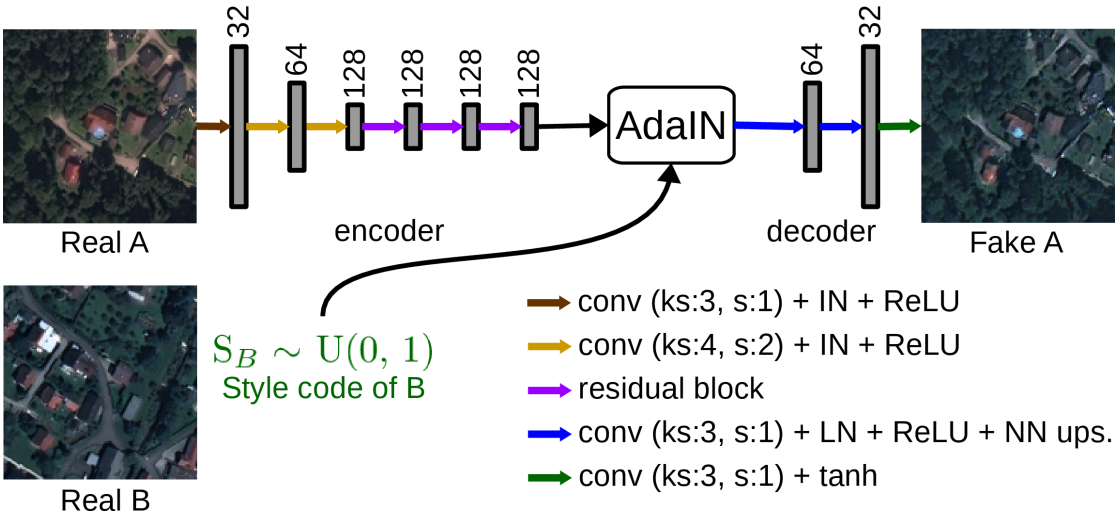


Figure 5.1: Combining the content of one domain with the style of another domain. ks and s correspond to kernel size and stride parameters of the convolution. IN, LN, and NN $ups.$ stand for instance normalization, layer normalization, and the nearest neighbor upsampling, respectively. The numbers above the gray rectangles denote the number of channels in each activation.

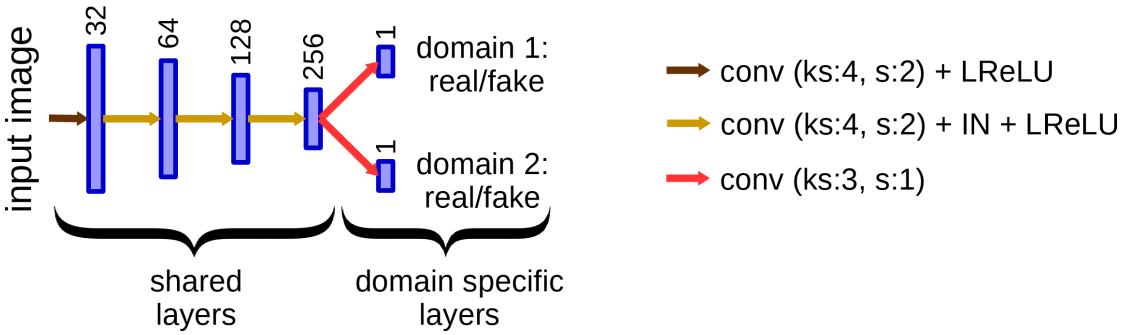


Figure 5.2: Multi-task discriminator. IN and $LReLU$ denote instance normalization and leaky rectified linear unit, respectively. The number of channels in each activation is indicated above the blue rectangles.

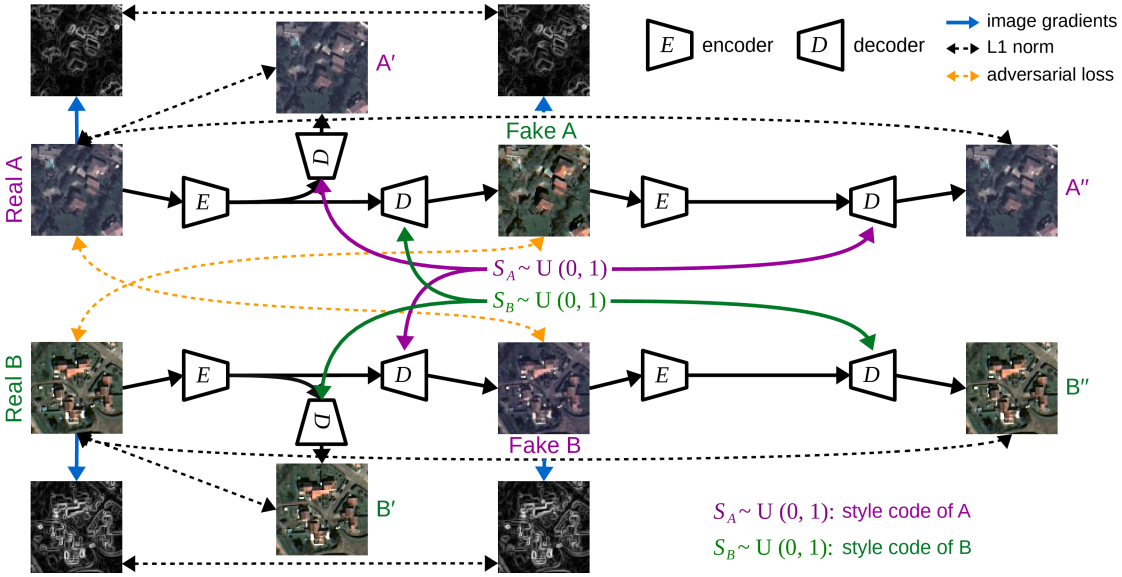


Figure 5.3: Style transfer between two domains. To generate fake A and fake B, we combine the embeddings of A and B with S_B and S_A, respectively. L1 norms enforce semantic consistency between A and fake A, and B and fake B. The adversarial losses force fake A and fake B to look like B and A, respectively.

normalized embedding via the decoder. As depicted in Fig. 5.1, the embedding extracted from one domain through the encoder consists of 128 channels, each of which is scaled and shifted by the style code parameters of the other domain with Eq. 5.1 before feeding the embedding to the decoder. Decoding the normalized embedding of A with S_B and the embedding of B with S_A results in generating fake A stylized as B and fake B with the style of A. StandardGAN introduced in the previous chapter aims at learning the parameters of S_A and S_B from A and B by domain specific so-called style encoders. Such approach requires using a different style encoder for each domain. However, we have discovered that it is possible to combine the content of A with the style B (or vice versa) using randomly initialized and unique S_A and S_B parameters instead of trying to learn them.

After combining A with S_B and B with S_A to generate fake A and fake B, we need to ensure that the data distributions of A and fake B, and B and fake A are as similar as possible. To overcome this issue, one could use adversarial learning to force the distributions of fake A and fake B to be similar to those of B and A, respectively. To perform style transfer between two domains, the current state-of-the-art I2I approaches such as SemI2I [168], CycleGAN [198], UNIT [104] use two discriminators, where one discriminates between A and fake B, and the other one tries to distinguish B from fake A. However, as discussed in Sec. 5.2, one of the challenges is to use as small number of networks as possible to easily extend the method to multi-domain adaptation problem. Hence, instead of including multiple discriminators, we propose to use only one multi-

task discriminator. As shown in Fig. 5.2, our discriminator comprises several shared layers and multiple domain specific layers. Each domain specific layer tries to understand whether the given data are fake or belong to that domain. Let us also remark that each domain specific layer of our discriminator outputs a two dimensional matrix. Each element of this matrix determines whether different parts of the input image are real or fake. We then take average of all the elements to generate a scalar quantifying the realness of the input. Our generator is depicted in Fig. 5.3. Let us denote by D_A and D_B the discriminator's outputs for A and B, and by G the generator. The adversarial loss for the discriminator is defined as:

$$\mathcal{L}_{D_{adv}}(X,Y) = -\big[\mathbb{E}_x[\log(D_X(x))] + \mathbb{E}_y[\log(1 - D_X(G(y)))]\big], \tag{5.2}$$

where X and Y denote two domains, and x and y correspond to patches sampled from these domains. The adversarial loss for the generator is described as:

$$\mathcal{L}_{G_{adv}}(X,Y) = -\big[\log(D_X(G(y)))\big]. \tag{5.3}$$

We compute the overall adversarial losses for the discriminator and the generator as:

$$\mathcal{L}_{D_{adv}} = \mathcal{L}_{D_{adv}}(\mathrm{A},\, \mathrm{B}) + \mathcal{L}_{D_{adv}}(\mathrm{B},\, \mathrm{A}) \tag{5.4}$$

and

$$\mathcal{L}_{G_{adv}} = \mathcal{L}_{G_{adv}}(\mathrm{A},\, \mathrm{B}) + \mathcal{L}_{G_{adv}}(\mathrm{B},\, \mathrm{A}). \tag{5.5}$$

Another challenge is to keep A and fake A, and B and fake B semantically consistent. Otherwise fake A and fake B would not match with the ground-truth of A and B, and could not be used to train a classifier. To enforce the semantic consistency, we define several constraints. Firstly, after we generate fake A and fake B, we combine their contents with the style codes of B and A (i.e., S_B and S_A) to generate A'' and B''. By switching the styles of fake A and fake B, the original domains need to be reconstructed. Hence, we minimize the cross reconstruction loss $\mathcal{L}_{cross}$ that is defined as:

$$\mathcal{L}_{cross} = |\mathrm{A} - \mathrm{A}''| + |\mathrm{B} - \mathrm{B}''|. \tag{5.6}$$

Secondly, when we combine A with S_A and B with S_B, we obtain A' and B'. Here, since we combine the content of each domain with its own style, A' and B' must be the same as A and B. Therefore, we minimize the self reconstruction loss that is computed as:

$$\mathcal{L}_{self} = |\mathrm{A} - \mathrm{A}'| + |\mathrm{B} - \mathrm{B}'|. \tag{5.7}$$

Finally, the textural features of A and fake A, and B and fake B must be very close. Let us assume that $Gr(\cdot,\cdot)$ is a function that takes two three-band images as inputs, converts them to gray-scale images, computes their horizontal and vertical gradients by Sobel filter, and sums the L1 norm between the horizontal and the L1 norm between the vertical gradients. We minimize the edge loss $\mathcal{L}_{edge}$ as:

$$\mathcal{L}_{edge} = Gr(\mathrm{A}, \text{fake A}) + Gr(\mathrm{B}, \text{fake B}). \tag{5.8}$$

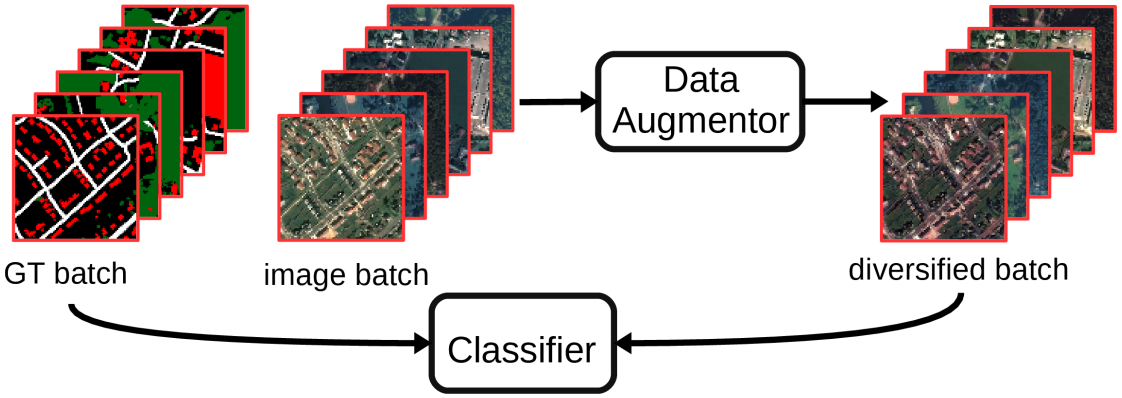


Figure 5.4: Training procedure of DAugNet that comprises a data augmentor and a classifier. In each training iteration, the classifier learns from the diversified batch generated by the data augmentor.

Note that other textural features such as Haralick features [65] can be considered as well. However, we prefer Sobel operator mainly because of its short execution time.

The final objective for the generator is computed as:

$$\mathcal{L}_G = \lambda_1 \mathcal{L}_{adv\_G} + \lambda_2 \mathcal{L}_{cross} + \lambda_3 \mathcal{L}_{self} + \lambda_4 \mathcal{L}_{edge}, \tag{5.9}$$

where $\lambda_1, \lambda_2, \lambda_3$, and λ_4 adjust the relative importance of each loss. The discriminator loss is defined as:

$$\mathcal{L}_D = \lambda_1 \mathcal{L}_{adv\_D}. \tag{5.10}$$

In the training stage, we minimize both $\mathcal{L}_G$ and $\mathcal{L}_D$ simultaneously.

5.5.2 DAugNet for Multi-source, Multi-Target, and Life-long Domain Adaptation

Let us assume that we have N domains out of which N_s of them are annotated source domains and N_t of them are unlabeled target domains. We also suppose that all of the domains have considerably different data distributions. In such a multi-source and multi-target segmentation setting, our goal is to generate maps for N_t unlabeled target domains. In the pre-processing step, we first split all of the domains into smaller patches and generate a unique style code for each domain. We then perform multi-domain style transfer by sampling a patch from randomly selected two domains among N and minimizing Eqs. 5.9 and 5.10 in each training iteration. After a sufficiently long training process, we are able to generate fake $N-1$ domains from each one. Our constraints explained in Sec. 5.5.1 enforce each fake domain to representative for a different domain and to be semantically consistent with the original domain.

Once the training process of the initial unsupervised and multi-domain style transfer is completed, we move on to training DAugNet. As illustrated in Fig. 5.4, the main components of DAugNet is the data augmentor and the classifier. In this stage, the data

augmentor is frozen, only the classifier is updated. The combination of the encoder and the decoder forms a data augmentor. Both the encoder and the decoder are shallow networks, where the former consists of six layers and the latter is composed of only two layers (see Fig. 5.1). In each training iteration of DAugNet, we randomly sample a batch of training patches from N_s source domains as well as their corresponding ground-truth. With 0.9 probability, the data augmentor diversifies the batch by extracting its embedding via the encoder, scaling and shifting the embedding for each patch with the style code parameters of a randomly selected domain, and decoding the normalized embedding. We then use the diversified batch and the ground-truth for the original batch to update the classifier by minimizing the classification loss $\mathcal{L}_{class}$ defined as:

$$\mathcal{L}_{class} = \lambda_5 \mathcal{L}_{sigmCE}(y, \hat{y}) + \lambda_6 \mathcal{L}_{soft\_IoU}(y, \hat{y}), \tag{5.11}$$

where $\mathcal{L}_{sigmCE}$, $\mathcal{L}_{soft\_IoU}$, y, and $\hat{y}$ denote sigmoid cross entropy loss, soft IoU loss [119], class labels, and the predictions, respectively. Diversifying the original batch with the proposed shallow data augmentor has two advantages. Firstly, such diversification approach allows the classifier to learn from the data that are representative for all N domains. Therefore, for target domains, we can expect DAugNet to achieve a better performance than naively training a classifier on the real data. Secondly, the diversification can be carried out in an online fashion, rather than storing all the fake domains in the pre-processing step and loading them when training the classifier. In each iteration, we deactivate the data augmentor with 10% chance for the classifier to learn from the real data as well.

To better explain life-long, multi-source, and multi-task adaptation problem, let us assume that we are provided additional M domains comprising M_s annotated source and M_t unlabeled target domains with different data distributions. The problem now becomes segmenting $N_t + M_t$ domains. To solve this problem, we first generate unique style codes for the newly added M domains. We load the style codes of the initial N domains and the pre-trained weights for the encoder, the decoder, and the classifier of DAugNet. We also initialize the parameters of the discriminator's shared layers as well as its domain specific layers for the previous domains with the pre-trained weights. For the current M domains, we add additional M domain specific layers to the discriminator with random initialization. We then conduct the initial training to perform style transfer between $N + M$ domains. This time, when selecting two random domains, we randomly choose one domain among the current M domains and another one from all $M + N$ domains we have at hand. With this sampling strategy, we guarantee that at least one of the domains is selected from the newly added ones. We finally fine-tune DAugNet with the same diversification method as explained above. Let us remark that every time we receive a new domain, we do not increase the number of networks. We only add a new layer to the discriminator. Considering that the networks with hundreds of layers can be fit into the current GPUs as in ResNet-152 [68], it is feasible to add many layers to the discriminator.

Table 5.1: The data set.

City (Country)	Class percentages (%)			256×256 patches
	Building	Road	Tree	
Villach (AT)	9.26	10.63	19.91	749
Sankt Pölten (AT)	6.68	6.39	25.13	1558
Bad Ischl (AT)	5.51	6.0	35.38	457
Salzburg Stadt (AT)	9.44	8.69	23.88	2496
Leibnitz (AT)	7.00	7.37	16.78	572
Bourges (FR)	9.81	10.52	14.83	1188
Lille (FR)	18.36	12.71	15.40	2181
Béziers (FR)	19.09	17.62	10.91	407
Albi (FR)	17.20	14.48	15.19	413
Vaduz (LI)	3.57	4.30	33.69	1612

5.6 Experiments

5.6.1 Data Set

Our data set consists of Pléiades images collected over five cities in Austria, four cities in France, and one city in Liechtenstein. We are given the annotations for building, road, and tree classes by the data providers. Table 5.1 records city names and the percentage of pixels belonging to each class. The image from Leibnitz is composed of near-infrared, red, green bands, and its spatial resolution is 0.5 m. The rest of the cities comprise red, green, and blue channels, and their resolution is 1 m. We reduce the resolution of the image from Leibnitz to 1 m so as to make our data set homogeneous in terms of spatial resolution. The images in our data set cover the total area of 663.02 km^2.

In the pre-processing step, we split all the images into 256×256 smaller patches with an overlap of 32 pixels. We use these patches in every stage of our training pipeline (i.e., both the multi-domain style transfer part and the process of training DAugNet). We indicate the number of patches belonging to each city in Table 5.1.

5.6.2 City-to-city Adaptation

In our first experimental setup, we perform city-to-city adaptation between Villach and Bad Ischl. A close-up from each city is depicted in Figs. 5.5a and 5.6a. We first use Villach as the source city and Bad Ischl as the target. We then switch the source and the target cities. In both cases, we assume that the source image is annotated and the target image is unlabeled. We compare our method with thirteen different approaches. The compared methods can be categorized into four groups: naive, adapting the classifier, data standardization, and image-to-image translation.

The approach belonging to the first category corresponds to training a U-net [144] on the source image and segmenting the target city without doing any domain adaptation. As confirmed by Tables 5.2 and 5.3, when there exists a large data distribution difference

Table 5.2: IoU scores for Villach in the first experiment (City-to-city adaptation).

Method			**Source: Bad Ischl ⇒ Target: Villach**			
Name		**Type**	**Building**	**Road**	**Tree**	**Overall**
U-net [144]		naive	29.00	3.78	58.52	30.43
AdaptSegNet Single [173]		adapting	6.01	4.37	10.43	6.94
AdaptSegNet Multi [173]		classifier	24.59	9.02	56.08	29.86
U-net trained on data generated by	Gray-world [24]	data standard.	11.98	25.66	60.91	32.85
	Hist. Equalization [60]		23.28	23.36	64.31	36.98
	Z-score norm. [4]		17.86	20.53	61.64	33.34
	Hist. Matching [60]	image to image trans.	8.20	20.18	54.81	27.73
	UNIT [104]		1.96	0.51	66.68	23.05
	MUNIT [78]		6.72	0.00	37.32	14.68
	DRIT [96]		0.00	0.00	15.14	5.05
	CycleGAN [198]		25.80	25.20	66.07	39.03
	ColorMapGAN [167]		42.28	**40.25**	65.28	49.27
	SemI2I [168]		40.18	32.66	66.83	46.56
DAugNet (ours)		divers.	**49.68**	35.32	**68.14**	**51.05**

Table 5.3: IoU scores for Bad Ischl in the first experiment (City-to-city adaptation).

Method			**Source: Villach ⇒ Target: Bad Ischl**			
Name		**Type**	**Building**	**Road**	**Tree**	**Overall**
U-net [144]		naive	5.12	0.08	0.00	1.73
AdaptSegNet Single [173]		adapting	3.06	2.71	10.23	5.33
AdaptSegNet Multi [173]		classifier	14.26	4.46	24.66	14.46
U-net trained on data generated by	Gray-world [24]	data standard.	34.30	29.08	58.12	40.50
	Hist. Equalization [60]		34.08	22.81	64.93	40.60
	Z-score norm. [4]		23.20	34.22	52.00	36.47
	Hist. Matching [60]	image to image trans.	2.27	0.13	0.00	0.80
	UNIT [104]		32.51	4.39	41.77	26.22
	MUNIT [78]		2.41	0.00	1.22	1.21
	DRIT [96]		0.00	0.30	0.00	0.10
	CycleGAN [198]		42.01	12.15	76.39	43.52
	ColorMapGAN [167]		52.00	**42.29**	47.28	47.19
	SemI2I [168]		49.32	41.06	70.18	53.52
DAugNet (ours)		divers.	**53.19**	41.84	**80.07**	**58.37**

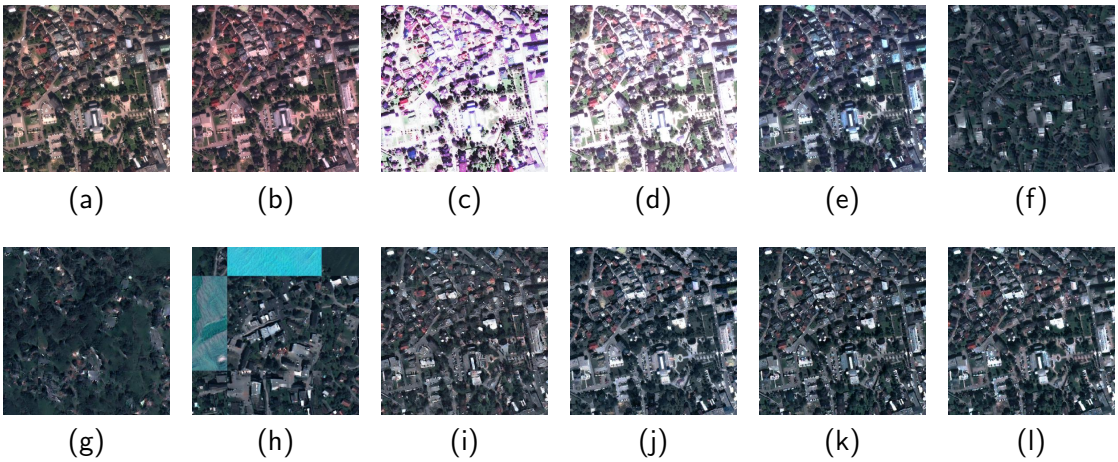


Figure 5.5: Bad Ischl and its modified versions to segment Villach. (a) Bad Ischl, (b) gray-world [24], (c) histogram equalization [60], (d) z-score normalization [4], (e) histogram matching [60], (f) UNIT [104], (g) MUNIT [78], (h) DRIT [96], (i) CycleGAN [198], (j) ColorMapGAN [167], (k) SemI2I [168], (l) DAugNet (ours).

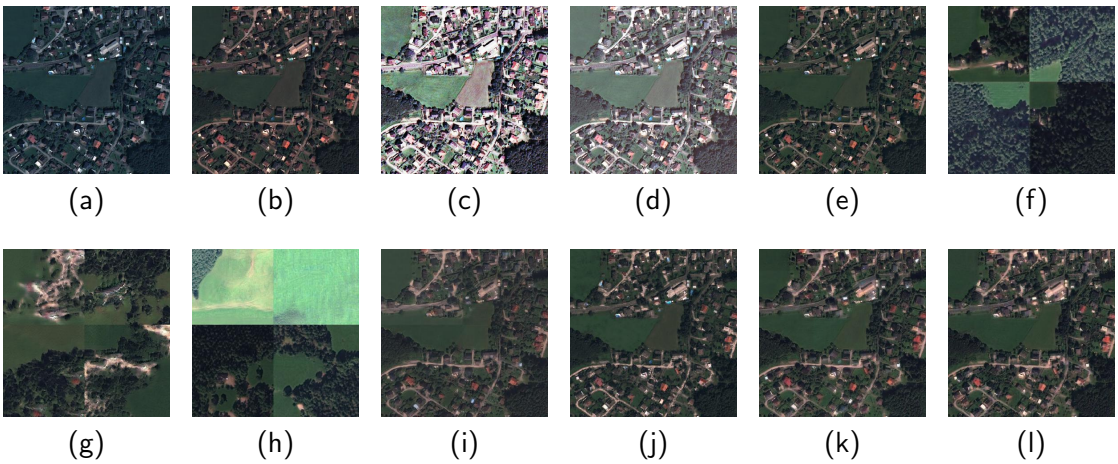


Figure 5.6: Villach and its modified versions to segment Bad Ischl. (a) Villach, (b) gray-world [24], (c) histogram equalization [60], (d) z-score normalization [4], (e) histogram matching [60], (f) UNIT [104], (g) MUNIT [78], (h) DRIT [96], (i) CycleGAN [198], (j) ColorMapGAN [167], (k) SemI2I [168], (l) DAugNet (ours).

between the source and the target cities, U-net fails to generate high quality maps. Especially for Bad Ischl, the quantitative results are extremely poor.

We also apply AdaptSegNet [173], which is an approach belonging to the second group, to our data set. This network architecture simultaneously learns from the source city and aligns the features extracted from both cities to train a domain agnostic classifier. The authors use DeepLab v2 [32] as the classifier. The feature alignment can be executed in one or multiple layers. In Tables 5.2 and 5.3, AdaptSegNet Single and AdaptSegNet Multi respectively correspond to the proposed method when the alignment is done in the last and in the last two layers before upsampling. However, the quantitative results prove that this method does not exhibit a good performance on domain adaptation of satellite images.

Among the methods based on data standardization, we compare our approach with gray-world algorithm [24], histogram equalization [60], and z-score normalization [4]. We first standardize images from both cities. We then train a U-net on the standardized source image and segment the standardized target data. Although these standardization methods enable U-net to better generalize, the improvement is unsatisfactory. As shown in Figs. 5.5 and 5.6, while the data distributions of the images get close to each other with these approaches, we still observe some differences.

To evaluate image-to-image translation based methods, we first a generate target stylized fake source city. Afterwards, we train a U-net on the fake source city and evaluate it on the target city. We provide the results for histogram matching [60], UNIT [104], MUNIT [78], DRIT [96], CycleGAN [198], ColorMapGAN [167], and SemI2I [168]. As illustrated in Figs. 5.5 and 5.6, the fake images generated by UNIT [104], MUNIT [78], and DRIT [96] are semantically inconsistent with the real source images. Therefore, U-net learns from data, where image and the ground-truth do not match. As a consequence, the quantitative results for these approaches are poor. Histogram matching does not take into account the contextual information. For example, fake image generated by histogram matching from Bad Ischl contains a lot of buildings with violet or cyan rooftops. However, such buildings do not exist in Villach. Since the fake source image is not representative for the target image, the performance of this approach is unsatisfactory. CycleGAN generates blurry fake source images, which negatively affects the performance of U-net. ColorMapGAN and SemI2I are better performers than the other ones. More detailed comparisons between city-to-city adaptation methods can be found in the third chapter.

When we train the multi-domain style transfer part of our solution, we set λ_1, λ_2, λ_3, and λ_4 parameters in Eqs. 5.9 and 5.10 to 1, 10, 10, and 100, respectively. We have found these values empirically. We train our style transfer method for 25 epochs. The learning rate for the initial 15 epochs is 0.0001. We reduce it gradually in the rest of the epochs as:

$$\mathrm{LR} = 0.0001 \times \frac{\mathrm{num\_epochs} - \mathrm{epoch\_no}}{\mathrm{num\_epochs} - \mathrm{decay\_epoch}}, \tag{5.12}$$

where LR, num_epochs, epoch_no, and decay_epoch denote the current learning rate, total number of epochs, current epoch no, and the epoch no where we start reducing

Table 5.4: IoU scores for Bad Ischl in the second experiment (multi-source and multi-target adaptation).

Method		IoUs for Bad Ischl			
		Building	**Road**	**Tree**	**Overall**
U-net [144]		35.48	26.36	51.24	37.69
U-net trained on data generated by	Gray-world [24]	50.30	43.00	70.41	54.57
	Histogram Equalization [60]	42.32	38.21	75.16	51.90
	Z-score normalization [4]	39.77	41.86	78.53	53.39
DAugNet		**59.70**	**49.05**	**83.39**	**64.05**

Table 5.5: IoU scores for Vaduz in the second experiment (multi-source and multi-target adaptation).

Method		IoUs for Vaduz			
		Building	**Road**	**Tree**	**Overall**
U-net [144]		12.49	18.68	58.34	29.84
U-net trained on data generated by	Gray-world [24]	19.45	19.59	18.85	19.29
	Histogram Equalization [60]	19.32	20.64	68.50	36.15
	Z-score normalization [4]	15.40	24.16	58.94	32.83
DAugNet		**51.83**	**34.41**	**69.74**	**51.99**

the learning rate. As the optimizer, we use Adam algorithm [90]. We set the beta coefficients of Adam to 0.5 and 0.999, respectively. When we train DAugNet, our data augmentor generates only target stylized image batches, since there are only two cities in this experimental setup. Note that any network architecture can be used as the classifier of DAugNet. We prefer to use U-net. We train DAugNet for 35 epochs with 0.0001 learning rate. Except for AdapSegNet, we train U-net for all the compared methods for 35 epochs with the same learning rate. When training DAugNet and U-net for the compared methods, we set λ_5 and λ_6 in Eq. 5.11 to 0.25 and 0.75, respectively. We sample a batch of 32 training patches and perform online data augmentation with random flips and rotations. Tables 5.2 and 5.3 attest that DAugNet performs the best in most cases.

5.6.3 Multi-source and Multi-target Domain Adaptation

To evaluate the performance of our approach on Multi-source and Multi-target adaptation setting, we choose Villach, Sankt Pölten, Bourges, and Lille as the source, and Bad Ischl and Vaduz as the target cities. In this setup, we train our initial multi-domain style transfer approach for 200 epochs. We use Adam optimizer with initial learning of 0.0001 and the same beta coefficients used in the previous experiment. Starting from the 100^{th} epoch, we reduce the learning rate via Eq. 5.12. We train DAugNet as described in Sec. 5.5.2 for 100 epochs. We compare our framework with the same standardization approaches explained in the previous experimental setup. Once all the data are stan-

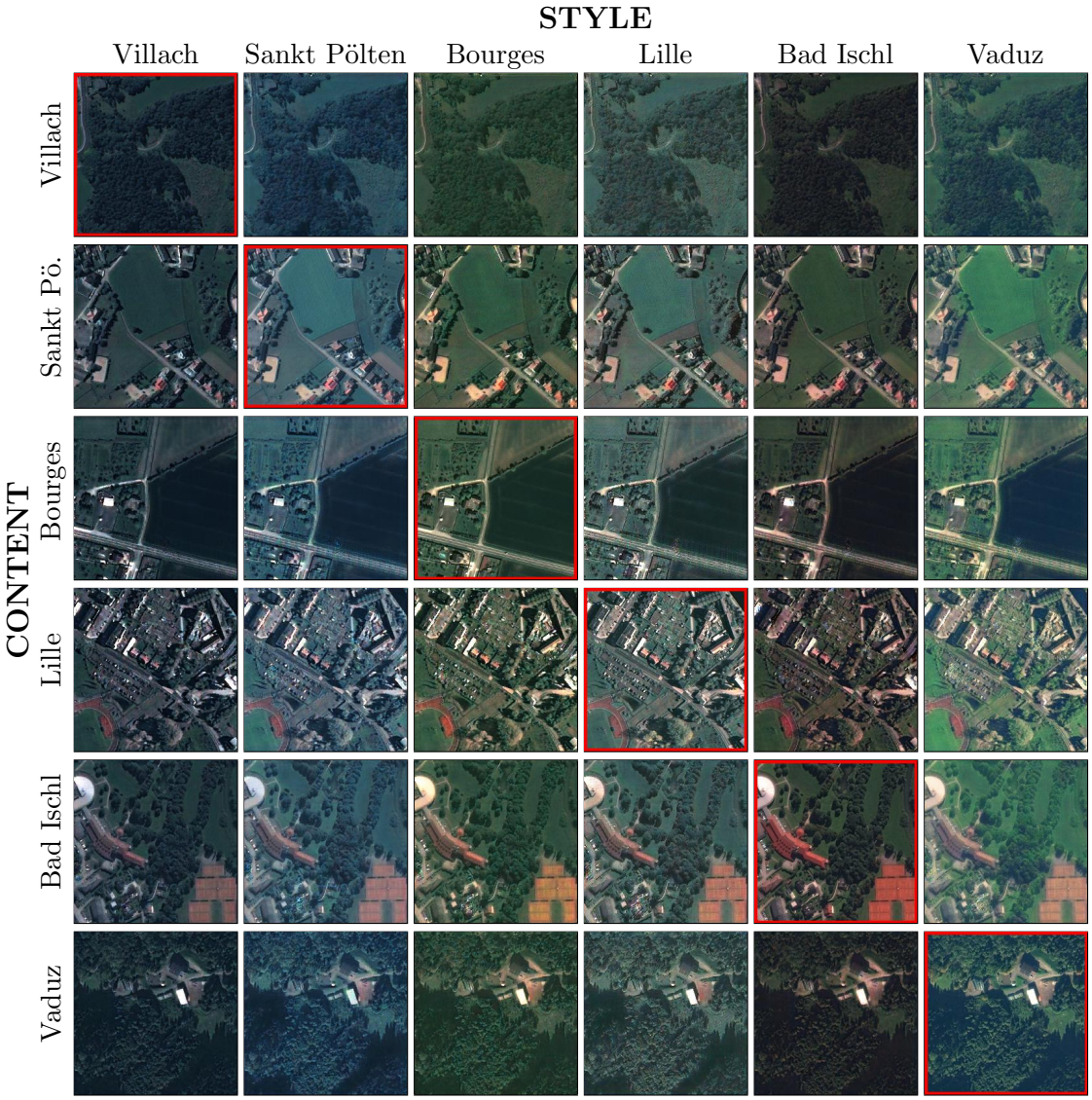


Figure 5.7: Style transfer between the images used in the second experiment. The cells with red bounding boxes are real data. The rest of the cells represent the fake data generated by our multi-domain style transfer approach.

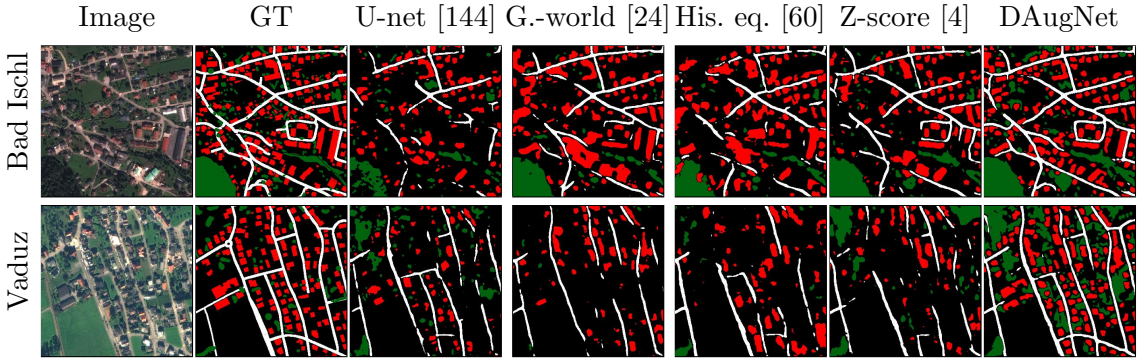


Figure 5.8: Target images used in the second experiment, their ground-truth, and the predictions. Red, white, and green pixels represent building, road, and tree classes, respectively. The pixels in black do not belong to any class.

Table 5.6: IoU scores for Bad Ischl in the third experiment (multi-source, multi-target, and life-long adaptation).

Method		**IoUs for Bad Ischl**			
		Build.	**Road**	**Tree**	**Overall**
U-net [144]		51.86	39.33	79.80	56.99
U-net on data generated by	Gray-world [24]	59.59	47.95	68.72	58.75
	Hist. Eq. [60]	44.03	39.94	78.56	54.18
	Z-score norm. [4]	46.58	44.62	75.25	55.48
DAugNet		**60.52**	**53.08**	**82.87**	**65.49**

dardized, we train a U-net on the standardized source images for 100 epochs and report the results on the standardized target cities.

Tables 5.4 and 5.5 report the IoU scores for multi-source and multi-target experimental setup. Our first observation is that adding more source cities with different data distributions bridges the domain gap between the source and the target cities. Thus, the classifier better generalizes to new geographic locations. For instance, although we use Bad Ischl as the target city both in this experiment and in the previous one, the quantitative results of each method for this city are significantly higher in this setting. Secondly, in some cases, the performances of the standardization algorithms are unstable. All of these algorithms allow the classifier to exhibit a considerably better performance on Bad Ischl, whereas the quantitative results for Vaduz are either roughly on par with or even worse than naive U-net. As depicted in Fig. 5.7, our multi-domain style transfer method is able to stylize each city as another one. As mentioned earlier, diversifying a batch of image patches when training DAugNet with the help of the data augmentor enables DAugNet to learn from data that are not only representative for the source cities but also representative for the target cities. Hence, DAugNet achieves a better performance than the others. Fig 5.8 shows a close-up from each city, its ground-truth, and the predictions.

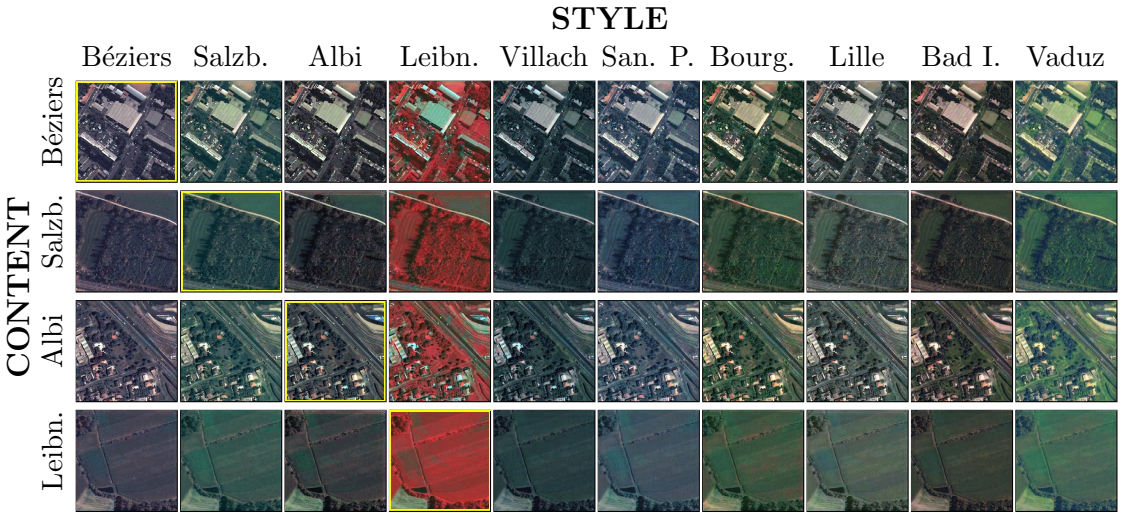


Figure 5.9: Style transfer between newly added cities in the third experiment and all the cities. The cells with yellow bounding boxes represent the additional real images. The rest of the cells show the fake images generated by our multi-domain style transfer approach.

Table 5.7: IoU scores for Vaduz in the third experiment (multi-source, multi-target, and life-long adaptation).

Method		**IoUs for Vaduz**			
		Build.	**Road**	**Tree**	**Overall**
U-net [144]		38.04	34.33	**75.88**	49.42
U-net on data generated by	Gray-world [24]	26.04	30.98	45.37	34.13
	Hist. Eq. [60]	19.19	23.64	71.11	37.98
	Z-score norm. [4]	29.08	28.97	48.38	35.48
DAugNet		**53.03**	**39.44**	71.77	**54.75**

Table 5.8: IoU scores for Leibnitz in the third experiment (multi-source, multi-target, and life-long adaptation).

Method		**IoUs for Leibnitz**			
		Build.	**Road**	**Tree**	**Overall**
U-net [144]		2.29	1.13	0.04	1.15
U-net on data generated by	Gray-world [24]	0.01	0.10	21.59	7.23
	Hist. Eq. [60]	1.23	0.42	55.93	19.19
	Z-score norm. [4]	0.10	1.14	11.80	4.35
DAugNet		**43.31**	**34.28**	**73.18**	**50.26**

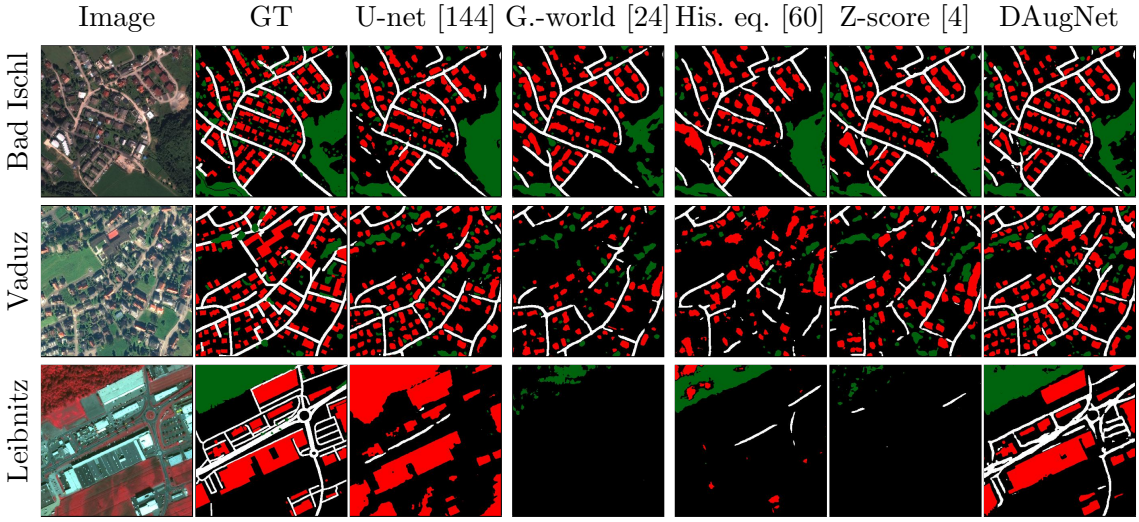


Figure 5.10: Target images used in the third experiment, their ground-truth, and the predictions. Red, white, and green pixels represent building, road, and tree classes, respectively. The pixels in black do not belong to any class.

5.6.4 Life-long Domain Adaptation

In the third experiment, we assume that in addition to the ones in the second experiment, we receive new annotated source images from Béziers, Salzburg Stadt, Albi, and unlabeled target data from Leibnitz. The image from Leibnitz consists of near-infrared, red, and green channels, whereas the other nine cities are composed of red, green, and blue spectral bands. This experimental setup has two goals. Firstly, we wish to verify whether our method can perform style transfer between ten images having different data distributions even when the type of spectral bands are different. Secondly, we want to observe the relevance of our solution for the life-long adaptation setting, where new source and target images are added. In this experimental setup, we have seven source and three target cities in total.

We first generate unique style codes for the new cities and train our style transfer method as described in Sec. 5.5.2 for 50 epochs. As in the first two experiments, we use Adam optimizer with the initial learning rate of 0.0001. We reduce the learning rate after the 25^{th} epoch via Eq. 5.12. Before training DAugNet, we initialize the parameters of U-net in DAugNet with the pre-trained weights from the second experiment. We then fine-tune DAugNet for 100 epochs. We compare DAugNet with the same data standardization approaches as the previous experiment. For each compared standardization method, we fine-tune the U-net trained in the second experiment for 100 epochs.

As confirmed by Tables 5.6, 5.7, 5.8, because of the new three source cities, the IoU scores of all the methods for Bad Ischl and Vaduz are higher than in the second experiment. For example, the performance of U-net especially for tree class significantly improves. It is probably because trees in the new cities are more representative for

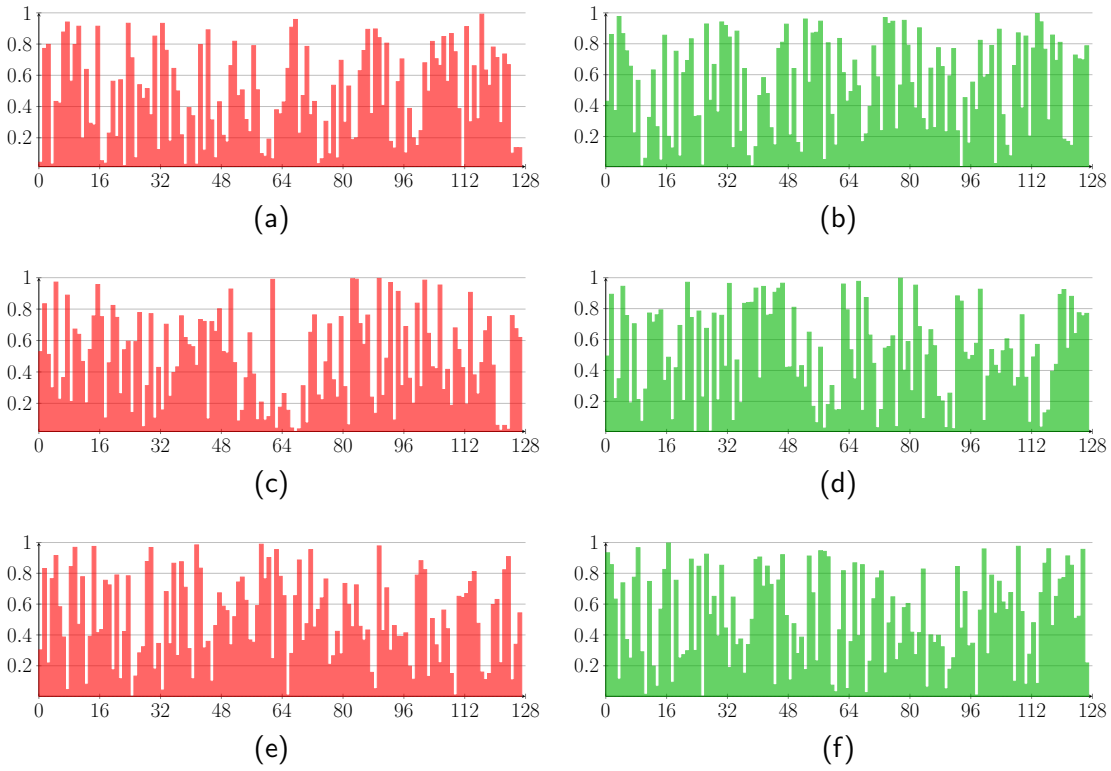


Figure 5.11: Random style codes of Villach in the second experimental setup and in the second ablation study. (a-b) γ and β values of Villach in the second experiment. (c-d) γ and β values of Villach in the first run of the second ablation study. (e-f) γ and β values of Villach in the second run of the same study.

the target cities. For instance, as can be seen in Fig. 5.9, the data distributions of trees in Salzburg stadt and Vaduz, and Albi and Bad Ischl seem close. Naive U-net slightly better detects trees in Vaduz than DAugNet. This slight performance difference probably stems from some artifacts added by our GAN architecture in the process of generating fake cities. However, DAugNet outperforms the compared approaches for all the other classes. When it comes to segmenting Leibnitz, the performances of the compared methods are extremely poor, mainly because the spectral bands of this image do not comprise red, green, and blue channels. On the other hand, as shown in Fig. 5.9, our style transfer method can easily convert red, green, blue images to near infrared, red, green ones. As a consequence, DAugNet significantly outperforms the others on Leibnitz, since it learns from data consisting of both red, green, blue and near infrared, red, green bands. Fig. 5.10 depicts a close-up from each city used in this experiment, its ground-truth, and the predictions.

Table 5.9: Quantitative results for the first ablation study.

City	Result type	Building	Road	Tree	Overall
Bad Ischl	IoUs for run 2	59.91	50.00	82.70	64.20
	IoUs for run 3	60.50	50.44	83.64	64.86
	standard dev.	0.34	0.58	0.40	0.35
Vaduz	IoUs for run 2	51.65	35.42	62.56	49.88
	IoUs for run 3	51.72	32.76	66.98	50.49
	standard dev.	0.07	1.10	2.96	0.89

Table 5.10: Quantitative results for the second ablation study.

City	Result type	Building	Road	Tree	Overall
Bad Ischl	IoUs for run 2	59.10	49.45	81.71	63.42
	IoUs for run 3	59.95	52.27	83.16	65.12
	standard dev.	0.36	1.43	0.74	0.70
Vaduz	IoUs for run 2	50.07	37.31	64.80	50.73
	IoUs for run 3	46.03	34.14	65.86	48.68
	standard dev.	2.43	1.43	2.12	1.36

(a) (b) (c)

Figure 5.12: Real Vaduz and Fake Vaduz in Bad Ischl style. (a) Vaduz, fake Vaduz by our approach with (b) edge loss and (c) without edge loss.

Table 5.11: Drop in IoU scores when edge loss is deactivated.

City	Building	Road	Tree	Overall
Bad Ischl	2.04	2.25	2.59	2.30
Vaduz	5.88	2.00	14.81	7.56

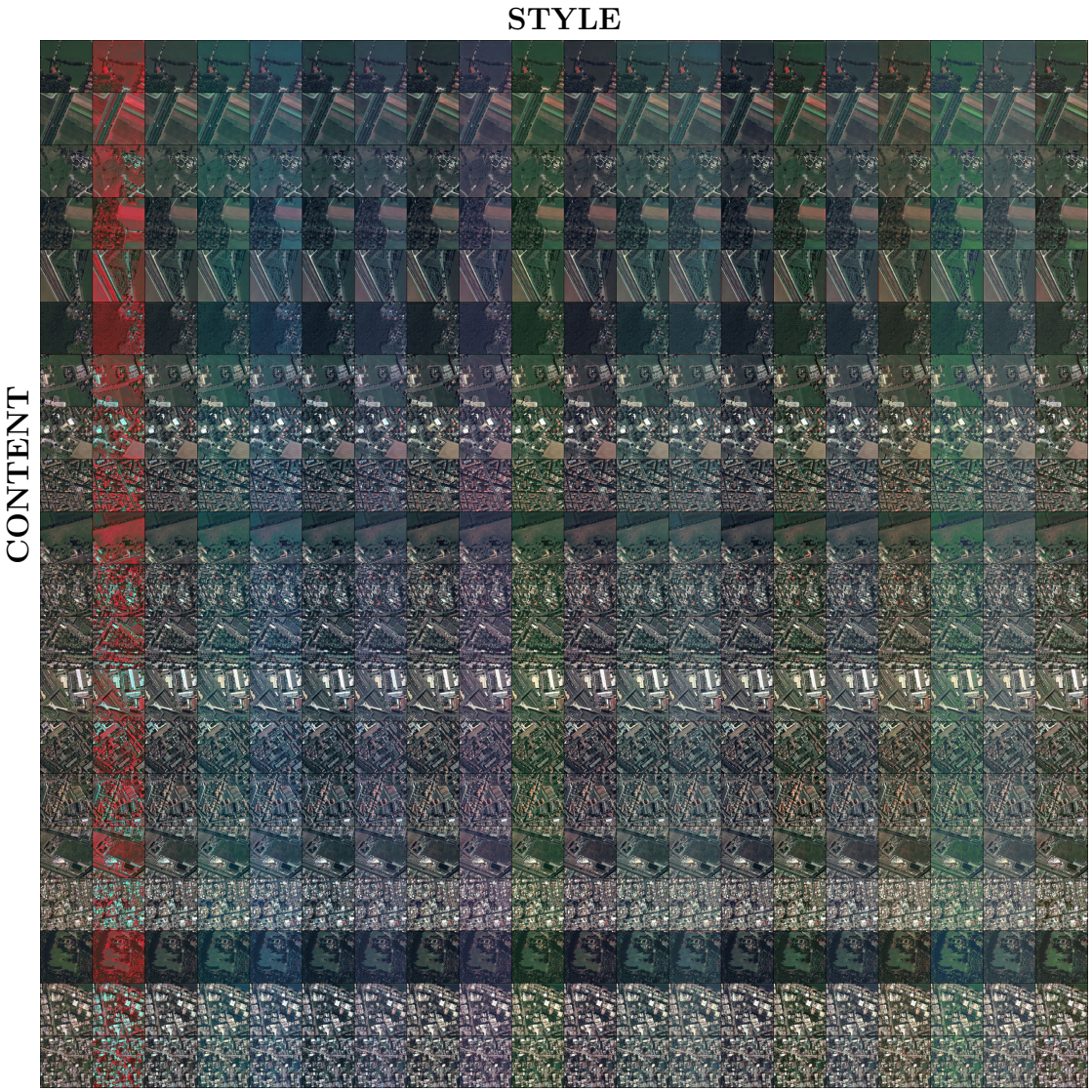


Figure 5.13: Style transfer between 20 satellite images, each of which has been collected from a different city. Rows and columns of this matrix correspond to content and style information, respectively. The cells on the diagonal from top-left to bottom-right represent real images. Each of the rest of the cells shows a fake city with the style of another one.

Table 5.12: DAugNet vs. U-net training time comparison.

Training Data (# cities)	# patches	Tr. time (h:m:s)	
		U-net	DAugNet
Source data in Exp. 2 (4)	5676	1:47:21	2:05:25
Source data in Exp. 3 (7)	8992	2:50:29	3:18:05

Table 5.13: Training time of multi-domain style transfer.

Tr. Data (# cities)	# patches	# epochs	Tr. time (h:m:s)
Cities in Exp. 2 (6)	7745	200	2:25:47
Cities in Exp. 3 (10)	11633	50	0:34:57

5.6.5 Ablation Studies

Effect of random diversification. As explained earlier, the data augmentor in DAugNet randomly diversifies the training batch before passing it to the classifier. In the first ablation study, we analyze the robustness of DAugNet to random diversification. We train DAugNet two more times from scratch on the source cities in the second experimental setup and segment Bad Ischl and Vaduz. Note that we do not train the initial multi-domain style transfer part of our pipeline again to evaluate the effect of only random diversification. Table 5.9 reports IoU scores of the two runs. In the table, we also indicate the standard deviation of IoUs in the second experiment and IoUs of these two runs. Because the standard deviations are quite small, we conclude that our framework is robust to random diversification.

Effect of random style codes and diversification. We repeat the whole training pipeline of our approach in the second experiment two times. Fig. 5.11 depicts the randomly initialized 128 γ and β values for Villach in the second experiment and in the two runs of this ablation study. In Table 5.10, we report IoU scores of these two runs and the standard deviations. Although the style codes (i.e., γ and β) of the cities are different in each run, the standard deviations for IoUs are very small. In conclusion, our solution is robust to random style codes and random diversification.

Effect of edge loss. We deactivate the edge loss by setting λ_4 in Eq. 5.9 to zero and repeat the second experiment to evaluate the effect of the edge loss. As can be seen in Fig. 5.12, when λ_4 is zero, our style transfer approach generates slightly blurred fake images, which causes DAugNet to exhibit a worse performance. The drop in IoU scores is reported in Table 5.11. Note that the other terms in Eq. 5.9 are mandatory; therefore, λ_1, λ_2, and λ_3 cannot be set to zero.

U-net vs. DAugNet training time comparison. Because the data augmentor needs to diversify the batch in each training iteration, the training time of DAugNet is

longer than that of U-net. Table 5.12 reports the time required to train DAugNet and U-net on the source data in the second and in the third experiments on an NVidia GeForce GTX 1080 Ti GPU. Due to its simple architecture, the data augmentor prolongs training time of the classifier for only 18 minutes in the second experiment, and for 28 minutes in the third experiment. Table 5.13 reports the time needed to train the multi-domain style transfer part of our framework in the second and in the third experiments.

Scalability of the proposed method. As mentioned in Sec. 5.2, significant technological advancements in satellite sensors and a large number of satellite missions have made massive volume of remote sensing data available. Therefore, it is of paramount importance to propose highly scalable methods that can efficiently process a lot of satellite images. To demonstrate that our method is capable of dealing with many satellite images, we perform style transfer between twenty images collected from different cities in Europe. Spectral bands of one image consists of near-infrared, red, green bands, whereas the other images comprise red, green, and blue channels. As depicted in Fig. 5.13, our style transfer method successfully stylizes each image like any other. This ablation study proves that our method is suitable to large-scale segmentation problems.

5.7 Concluding Remarks

The satellite missions launched in the last decade have enabled us to collect huge volume of remote sensing data over the entire Earth every day. Because of various atmospheric conditions, sensor characteristics, and differences in acquisition, remote sensing images collected from separate geographic locations in distinct times tend to have largely different data distributions. The data distribution difference between source and target images prevents the machine learning models from generating precise maps. Moreover, scarcity of annotated data motivates us to propose methods that are robust to such distribution difference. In addition, it is crucial to introduce new approaches that can adapt to continually growing data.

In this context, we presented a new approach for multi-source, multi-target, and life-long domain adaptation problem. In the pre-processing stage of our method, we learn how to perform style transfer between multiple source and target domains using only one encoder, one decoder, and one discriminator. A subset of the networks used in this stage constitutes a data augmentor. We then train the proposed DAugNet comprising the data augmentor and a classifier. In each training iteration, the data augmentor randomly diversifies the training batch before passing it to the classifier. As a consequence, the classifier is more robust to the data distribution difference between the images, since it learns from the data that are representative for all source and target cities.

The state-of-the-art methods in the literature either aim at solving single-source and single target adaptation problem or use multiple networks to deal with multi-source domain adaptation problem. However, because the number of networks in multi-domain style transfer stage (pre-processing step) of our method is constant regardless of the number of cities, our approach enabled us to perform adaptation between many cities.

For the same reason, we could extend our solution to life-long adaptation setting. In three extensive experiments, we verified effectiveness of our solution. Moreover, in five ablation studies, we analyzed the properties of our approach in great detail, and we demonstrated that our method is scalable.

One limitation of our approach is that the training time of DAugNet is slightly longer than that of U-net. However, as confirmed by Table 5.12, the difference is kept small due to the simple architectural design of the data augmentor. Another limitation is that our method matches the distribution of one image with the global distribution of another one when we perform style transfer. Matching the global distributions does not guarantee similarity between class specific distributions. For example, let us assume that we are interested only in road segmentation. When we stylize a source city as the target city, in some cases, roads in the fake training and the target cities may not look similar even though their global distributions are similar. In the literature, there is a recent research activity on attention guided image-to-image translation [35,120,155,185]. These methods aim at performing translation between only the desired objects via the proposed attention mechanisms. Our approach could be further improved by adopting such attention methods.

Chapter 6

Vectorization of Buildings via Mesh Approximation

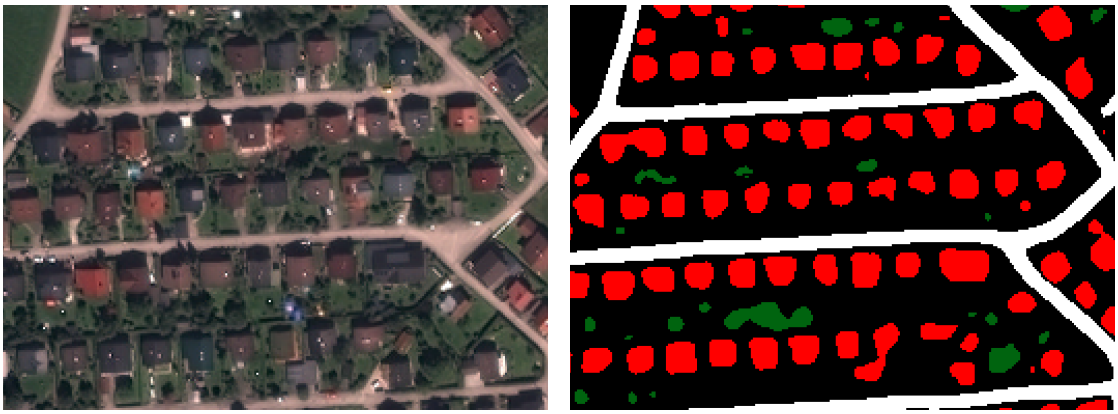
6.1 Problem Definition

In this chapter, we introduce a method based on mesh approximation to vectorize the raster maps. Here, we are particularly interested in the building class. The input for the proposed approach is a binary segmentation mask generated by a machine learning model. The desired output is a vector map in which building contours are precisely delineated.

6.2 Motivations

The maps that are used in Geographic Information System (GIS) applications can be split into two categories: raster maps and vector maps. Raster maps comprise a grid of pixels. They are generated by assigning a class label to each pixel in the remotely sensed image. Vector maps, on the other hand, consist of features. The features in GIS applications usually correspond to points, lines, or polygons [157]. In the previous chapters, we devised efficient approaches allowing to generate raster maps from satellite or aerial images. However, vector maps are more desired in many real-world GIS applications, because they have several advantages over raster maps.

First of all, they allow a more compact representation that enables fast processing and easy management. For instance, if we would like to create a polygon feature that delineates borders of a building, it is sufficient to store only the points located on the building corners. Secondly, vector maps enable users to easily query the data. With vector maps, it is feasible to answer the queries such as "How many buildings are located in this area?", "What are the areas of the selected objects?", "How many objects intersect with the selected road?", "What is the shortest path between two geographic locations?", etc. When enriched with semantic, maps can answer many more application-dependent usages. Thirdly, vector maps represent the objects more accurately. For instance, the



(a) (b)

Figure 6.1: Close-ups from a satellite image and its corresponding raster map generated by U-net [144]. (a) Satellite image, (b) predictions by U-net. Red, white, and green colors represent building, road, and tree classes, respectively. As can be seen, building corners are overly rounded.

user can zoom in/out unlimited number of times without observing any pixel effect unlike in raster maps. Finally, in vector maps, geometric properties such as topology can easily be described and maintained.

When vectorizing the raster maps, we encounter with several challenges. For example, it is often the case that building corners in the raster maps predicted by a machine learning model are overly rounded (see Fig. 6.1). In such cases, the vectorization algorithm should correct the raster maps so that corners of the buildings meet at right angle. The correction can be performed either by introducing geometric prior information such as right angle regularity of building corners, parallelism of curves or edges delineating roads, etc., or by using satellite images in addition to the raster maps, as the images carry spectral and spatial information. Another challenge is generating vector maps that efficiently represent all the objects. Otherwise, the maps may not meet the needs of real-world applications because of their long processing times. Therefore, we must generate maps with as small number of vertices, edges, etc. as possible.

The advantages described here have motivated us to generate efficient vector maps from remote sensing data.

6.3 Related Work

6.3.1 Polygon Generalization

In order to generate a digitized representation, the most straightforward approach is to vectorize the classification map, and simplify the polylines of the complex vectorized output, which is referred to as polygon generalization [56]. Among the polygon gener-

alization methods, Radial distance [157] and Reumann-Witkam [140] are quite similar. The former removes vertices located inside tolerance circles centered at vertices of interest, and the latter computes the line passing through two consecutive vertices and removes the vertices that are closer to this line than a tolerance value. The Valingam-Whyatt approach [178] ranks all the vertices in accordance to their significance derived from their effective area, and iteratively removes the less significant vertices if their effective areas are lower than a tolerance value. The common Douglas-Peucker [49] approach computes the edge joining the first and last vertices, and finds the farthest vertex from this edge. If the distance between this vertex and the edge is larger than a threshold, the algorithm is repeated recursively with the first and farthest, then farthest and last vertices. If the distance between the farthest vertex and the edge is smaller than a threshold, all intermediate vertices are removed.

All of these methods are available in most GIS packages such as GRASS and QGIS. Douglas-Peucker is the most commonly used method in the community. It has been extended to preserve topology and to generate outputs that do not contain self-intersecting polygons [149, 179].

6.3.2 Mesh Approximation

A polygon mesh is defined as a collection of vertices, edges, and faces to represent the shape of a polyhedral object. One of the most commonly used meshes is triangle mesh that divides the space into non-overlapping triangles [16, 44].

Mesh approximation has been a long-standing problem in geometry processing, where a set of operators are defined and applied to modify the mesh [16]. A mesh can be modified by both discrete operators such as edge flips and collapses [16] and continuous operators like vertex relocation [16]. A combination of different operators can be used for mesh approximation [14]. In the context of generating vector maps from raster maps, labeled triangle meshes can be used [113, 169].

When approximating a mesh via a set of certain operators, we may confront with several challenges. One of the common issues is ending up having a mesh that contains degenerate triangles. The degenerate triangle is a triangle where all three vertices form almost a line. This problem can be overcome by adding some constraints to the operators. For instance, we can force the smallest degree in each triangle to be larger than a certain threshold. On the other hand, such constraints may limit the flexibility of the operators. After approximating the mesh, a post-processing can be another way to avoid degenerate triangles [15]. Another challenge is topology preservation. Especially collapse operators (e.g., half-edge collapse) may cause distorting the topology. The topology preservation can be enforced via Euler characteristic [141].

6.3.3 Learning Based Approaches

These approaches aim at training a machine learning model that can generate polygonal segmentation for a given image.

To this end, inspired by interactive segmentation tools [145], some semi-supervised learning approaches have been proposed. PolygonRNN [27] and its improved version PolygonRNN++ [1] are examples of such semi-supervised methods. Both approaches require the user to draw a bounding box around the object of interest. PolygonRNN first predicts location of the first vertex inside the bounding box. It then takes the last two vertices and the first vertex as inputs in the current time step and tries to predict location of the next vertex using a combination of CNNs and recurrent neural networks (RNNs) [156]. PolygonRNN++ extends PolygonRNN by introducing the use of graph neural networks (GNNs) [100, 152]. Since one must draw a bounding box around the object for which a polygonal segmentation would be generated, these approaches are not feasible for vectorizing objects in satellite images. To avoid human interaction, PolyMapper [102] proposed to use feature pyramid networks [103] to extract bounding boxes around buildings. This extension made the approach end-to-end trainable.

Active contour models (ACMs) [88] historically have been used for many segmentation problems. These models constrain the polygonal segmentation to a set of curves and optimize them by minimizing a cost function consisting of several terms such as continuity and smoothness. To further improve their performance, additional terms (e.g., balloon [39]) have been proposed as well. Recently, a few deep learning based methods that learn parameters of the ACMs have been proposed [66, 117]. In the papers, it has been proven that these methods can be trained end-to-end and can be used for building segmentation. However, the major limitation of these approaches is that building borders are delineated by curves, which results in overly rounded building corners. In remote sensing data, buildings usually have regular shapes; their corners oftentimes meet at a factor of 90 degrees.

6.4 Contributions

The contributions of this chapter are as follows:

- We present a mesh approximation based approach, where a dense initial mesh is decimated and optimized using local edge and vertex-based operators in order to minimize an objective function that models a balance between fidelity to the classification map in ℓ_1 norm sense, right angle regularity for polygonized buildings, and final mesh complexity.
- Our method yields a better performance with a more compact representation. In our experiments, we compare our approach with common polygon generalization methods available in GIS software. Our experiments prove that our approach generates vector maps that significantly better represent buildings with less number of vertices.

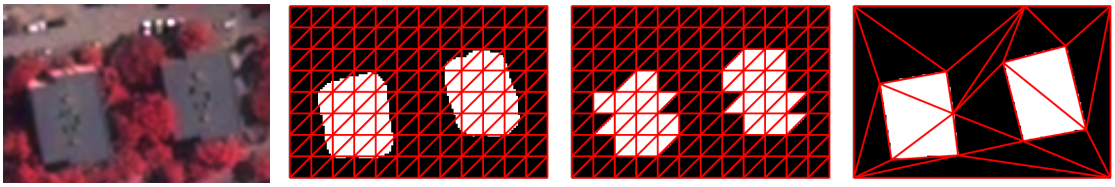


Figure 6.2: Input image and example labeled meshes. (a) Input image, (b) Initial fine lattice, (c) Initial and (d) Optimized labeled triangle meshes. The triangles labeled as building are indicated by white.

6.5 Method

6.5.1 Objective Function

In this chapter, we extend a recent method [113], which uses a binary labeled triangle mesh to approximate the input classification map. In the recent approach, the objective function is designed to trade mesh complexity for fidelity to the classification map in ℓ_1 norm sense. While deep neural network approaches yield classification maps with high accuracy, a closer visual inspection reveals that such maps do not delineate the building contours perfectly, in particular near building corners that are overly rounded. Using only the classification map yields artifacts in the vectorized outputs. Based on a prior knowledge that most building edges meet at right angles, we add a novel geometric regularity term to the objective function, which favors angles of building corners being a factor of $\frac{\pi}{2}$ radians.

We denote by $\mathcal{L}$ a set of binary labels, where $l \in \mathcal{L}$ is 1 for building and 0 for non-building class. We denote by T the labeled triangle mesh consisting of triangles $\{t_i\}$, $A(t)$ the area of triangle t, l_t its label, and V_t its vertices. Θ_i denotes the wedge of a vertex $v_i \in V_t$, defined by summing the angles between the edges origination from v_i, of the consecutive faces with label 1 (building), starting from t and incident to v_i. Fig.6.3 depicts three wedges of a triangle t.

Our goal is to minimize the following objective function:

$$E(T) = \sum_{t \in T} \Bigg[(1 + \delta(l_t)) \left(\min_{l_t \in \mathcal{L}} \iint\limits_{x,y \in t} C_{prob}(l_t, x, y)\, dx\, dy \right) + \delta(1 - l_t) \left(\sum_{v_i \in V_t} C_{reg}(\Theta_i) \right) \frac{A(t)}{3} + \lambda \Bigg], \tag{6.1}$$

where $C_{prob}(l_t, x, y)$ denotes the cost of assigning label l_t to the pixel located at x, y, $C_{reg}(\Theta_i)$ is the regularity cost for wedge Θ_i, and λ provides a means to trade mesh complexity for fidelity.

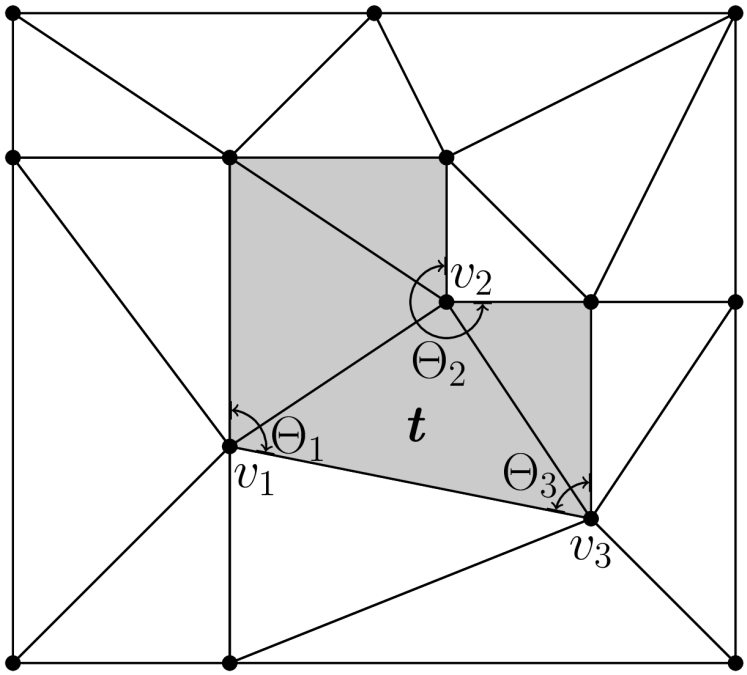


Figure 6.3: Three wedges of triangle t. Θ_1, Θ_2 and Θ_3 correspond to wedges of vertices v_1, v_2 and v_3, respectively. Triangles with label 1 are depicted in gray.

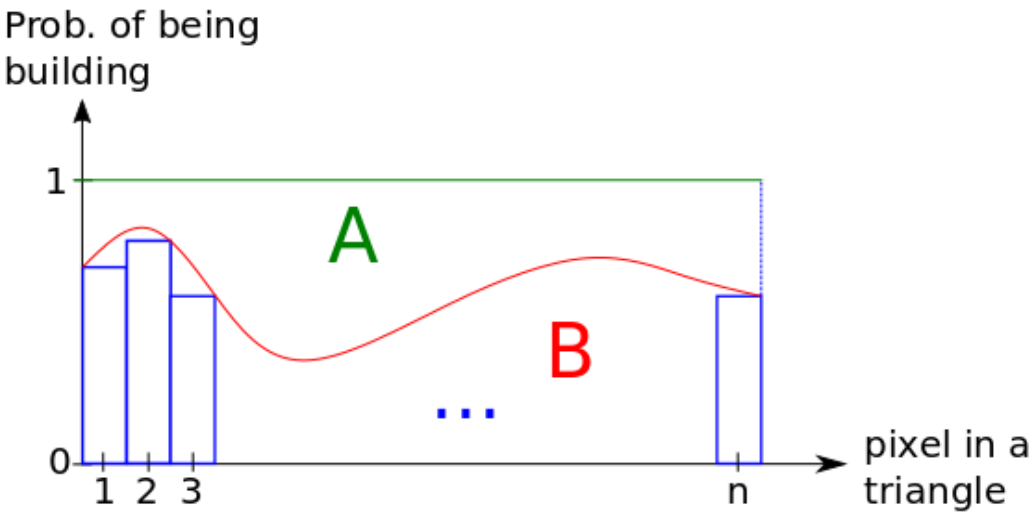


Figure 6.4: An illustration for C_{prob}. Blue rectanges represent probability of each pixel in a triangle to be building. In this figure, C_{prob} can be considered as minimum of A and B.

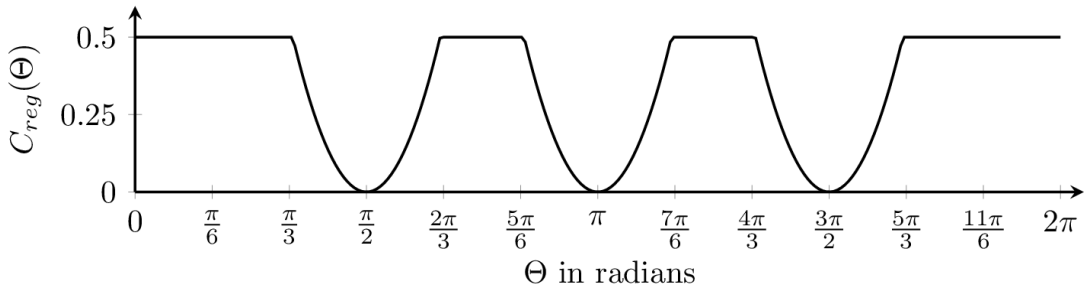


Figure 6.5: $C_{reg}(\Theta)$ when *scale* is 0.5 and *skewness* is 2.

$P(l, x, y)$ denotes the probability, estimated by a classifier, of assigning a label $l \in \mathcal{L}$ to a pixel located at (x, y) in the image. We define the probability cost in Eq. 6.1 as:

$$C_{prob}(l, x, y) = \|1 - P(l, x, y)\|_1, \tag{6.2}$$

which may be seen as the volume contained between the classifier's probability surface and the approximation surface delineated by the labeled mesh, as can be seen in Fig.6.4.

The regularity cost function $C_{reg}(\Theta)$ is designed to favor that edges of building corners meet at right angles (i.e., factor of $\frac{\pi}{2}$ radians) and is defined as:

$$C_{reg}\left(\Theta\right) = \min_{k=\{1,2,3\}} \left\{ scale, skewness \left(\Theta - \frac{k\pi}{2} \right)^2 \right\}, \tag{6.3}$$

where *scale* is the maximum cost for a wedge, and *skewness* adjusts how tolerant the system would be to the distance from the closest factor of $\frac{\pi}{2}$ for a wedge. A large value for the *skewness* parameter reduces the tolerance to the distance from the closest factor of $\frac{\pi}{2}$, and vice-versa. We set *scale* parameter to 0.5 to ensure that probability and regularity costs range between 0 and $0.5A(t)$. The *skewness* parameter is set to 2 by default. A plot for $C_{reg}(\Theta)$ with default parameters is shown in Fig.6.5.

The last term λ in Eq. 6.1, also summed over the mesh triangles, provides a means to control the balance with the final mesh complexity (a large value for λ decreases the number of triangles, and vice-versa). $\delta(\cdot)$ denotes the Dirac delta function, with value one at zero and zero elsewhere. The regularity cost for a triangle is computed only if it has been classified as building (i.e., $l_t = 1$) and if at least one of its vertices is located on building borders. Otherwise, its regularity cost is ignored and the probability cost in Eq. 6.1 is multiplied by 2 so that the total cost for each triangle consistently and ranges between 0 and $A(t)$.

6.5.2 Operators

We now detail the local mesh-based operators utilized to minimize the objective function.

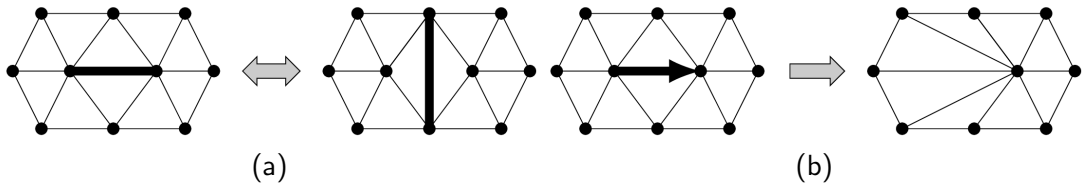


Figure 6.6: Edge based operators. (a) Edge flip, (b) Half-edge collapse.

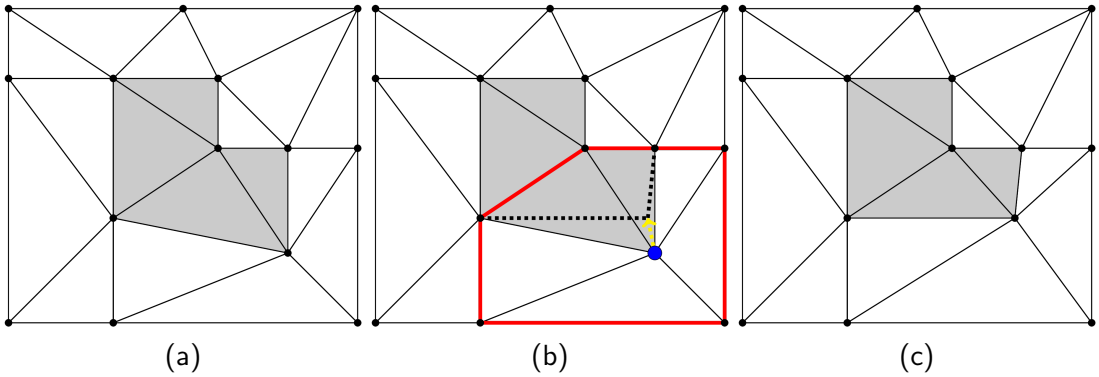


Figure 6.7: Vertex relocation. Mesh (a) before relocation, (b) during relocation, (c) after relocation. The triangles with label 1 are depicted in gray, and the others in white.

Discrete edge-based operators. Among the existing discrete operators in the literature, we use edge flips and half-edge collapses. The flip operator is valid for an edge only if the edge is between two adjacent triangles. This operator is used to flip the inner edge as illustrated in Fig. 6.6a. It is useful to improve edge alignment with building borders. Let us assume that the vertices of a line segment are denoted by A and B. Half-edge collapse operator collapses A to B or B to A as depicted in Fig. 6.6b. Note that collapsing A to B or B to A result in obtaining different meshes. Therefore, half-edge collapse can be applied to each line segment in two different directions. In order for this operator to be valid on a line segment, we need to verify that the resulting planar subdivision will be valid. For example, after applying the operator there should not be any intersecting triangle.

Continuous vertex-based operator. The limitation of edge-based operators is that they work on only fixed vertices that have already been placed. In order to compensate this limitation and increase expressiveness of the labeled triangle mesh we utilize another operator, which relocates the vertices in continuous space (see Fig. 6.7).

The kernel of a polygon is described as the locus of the points inside the polygon, from which all the vertices of the polygon are visible. The kernel of a polygon can be found by intersecting the half-planes determined by edges of the polygon [95]. Note that we must relocate each vertex only inside the polygon kernel in order to keep the

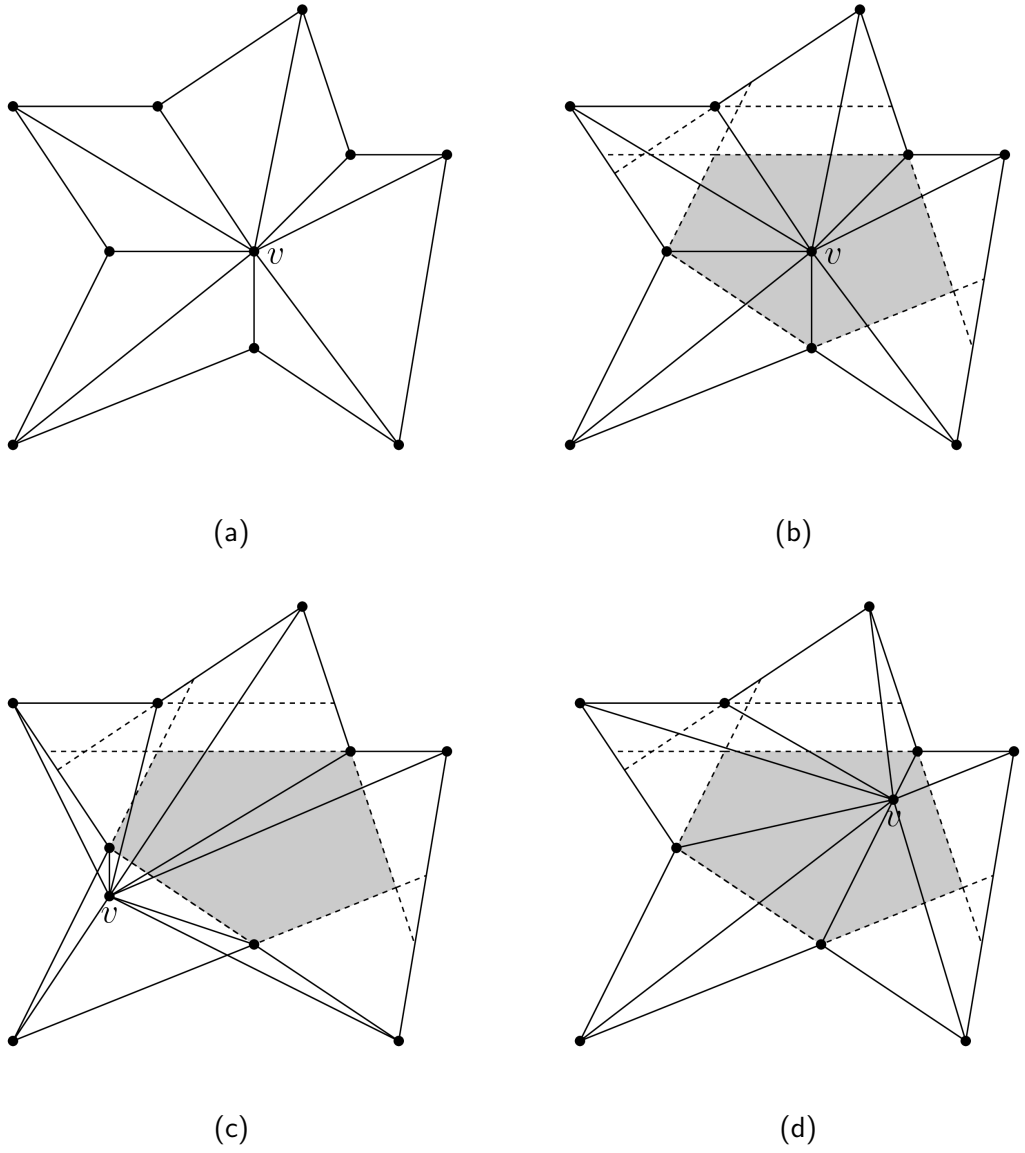


Figure 6.8: Illustration of polygon kernel, valid and invalid relocation of vertex v. (a) Example polygon, (b) its kernel, (c) invalid vertex relocation that spoils the triangulation by introducing self-intersecting triangles, (d) valid vertex relocation.

triangulation valid. If a vertex is relocated outside the kernel, the triangulation structure is spoiled, because some self-intersecting triangles are added. Fig. 6.8 depicts an example polygon, its kernel, valid and invalid vertex relocation operations.

When relocating a vertex, the main challenge is to determine the moving direction that reduces the objective defined in Eq. 6.1 for the vertex. Because the objective function is not differentiable w.r.t current location of the vertex, gradient based optimization approaches are not applicable to our problem. We adopt the simplex method [125] for function minimization. We denote the vertex that would be relocated by v. We generate two random vertices inside the kernel of the polygon (see Fig. 6.8b) that consists of the faces incident to v. The triangle formed by v and the two random vertices is moved to find the new location for v, by using three different movement types: reflection, expansion, and shrink. We define the best and the worst points of the triangle as the points, to which if v is relocated, would produce the lowest and the highest values for the objective defined in Eq. 6.1. We denote these points by P_b and P_w, cost of a point P_i of the triangle by y_i, and distance between P_i and P_j by $[P_iP_j]$. The reflection of P_w is denoted by P^* is and defined as:

$$P^* = (1+\alpha)\overline{P} - \alpha P_w, \tag{6.4}$$

where $\overline{P}$ is centroid of the triangle and α is the ratio of $[P^*\overline{P}]$ to $[P_w\overline{P}]$. If y^* is between y_w and y_b, P_w is replaced by P^* and the reflection process starts again. If $y^* < y_b$, the new minimum is found, P^* is expanded to P^{**} as:

$$P^{**} = \gamma P^* + (1-\gamma)\overline{P}, \tag{6.5}$$

where γ is the ratio of $[P^{**}\overline{P}]$ to $[P^*\overline{P}]$. If $y^{**} < y_b$, we replace P_w by P^{**} and restart the process. On the contrary, if $y^{**} > y_b$, that means it is a failed expansion, and P_w is replaced by P^* again. If $y^* > y_w$, P_w is either kept at its original location or replaced by P^* depending on which location gives the lower cost; then, the contraction operation is applied as:

$$P^{**} = \beta P_w + (1-\beta)\overline{P}, \tag{6.6}$$

where β is the contraction coefficient, which is the ratio of $[P^{**}\overline{P}]$ to $[P\overline{P}]$. We accept P^{**} for P_w and restart the process unless $y^{**} > min(y_w, y^*)$, which means the contracted point is worse than the better of P_w and P^*. For such a failed contraction movement, all the P_i's are replaced by $(P_i + P_b)/2$ and the process is started over.

Once the moving triangle includes the new optimum location for v, it keeps shrinking, i.e. its area is getting smaller and smaller at each iteration. We define three stopping criteria for the relocation algorithm. Firstly, we check if area of the triangle became smaller than a predefined threshold value. Secondly, we check whether the total number of iterations reached a threshold value. Finally, to avoid degenerate triangles, we check whether after the relocation, the minimum angle in each triangle would be smaller than 3 degrees. If one of these conditions is fulfilled, we stop the execution.

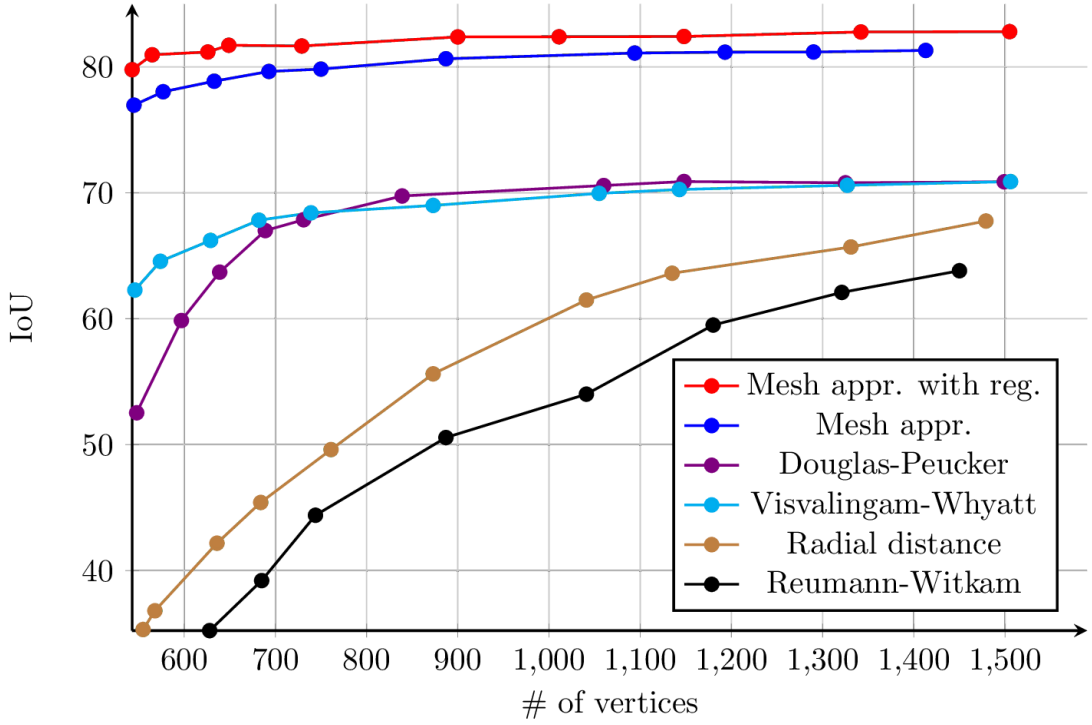


Figure 6.9: IoU vs. # of vertices plot.

6.5.3 Implementation

Starting from the initial lattice, we iteratively optimize the labeled triangle mesh. We simulate each change, caused by the operators, that transforms the mesh T to T' and calculate the objective difference $\Delta E = E(T') - E(T)$ incurred by the change. We push all the operators to a priority queue, where they are sorted according to their ΔE value in ascending order. The first element in the queue is popped first and applied if its associated ΔE is negative. In each iteration, we relabel all the affected triangles, recalculate their costs, and update the priority queue accordingly. The iterations continue until there is no operator left in the queue.

Although the labeled mesh is optimized by the operators according to Eq. 6.1, in some cases results may not be visually appealing because of the topology change. We observe the topology change clearly when neighboring buildings are very close to each other. In such a case, multiple objects are merged into one. To describe the topology, we use Euler characteristic $\chi = V - E + F$, in which V, E and F are number of vertices, edges and faces that are adjacent to triangles, labeled as building in the mesh. In order to preserve the topology, we compute difference $\Delta V, \Delta E$, and ΔF between in T and T', and transform T to T' only if $\chi = \Delta V - \Delta E + \Delta F = 0$.

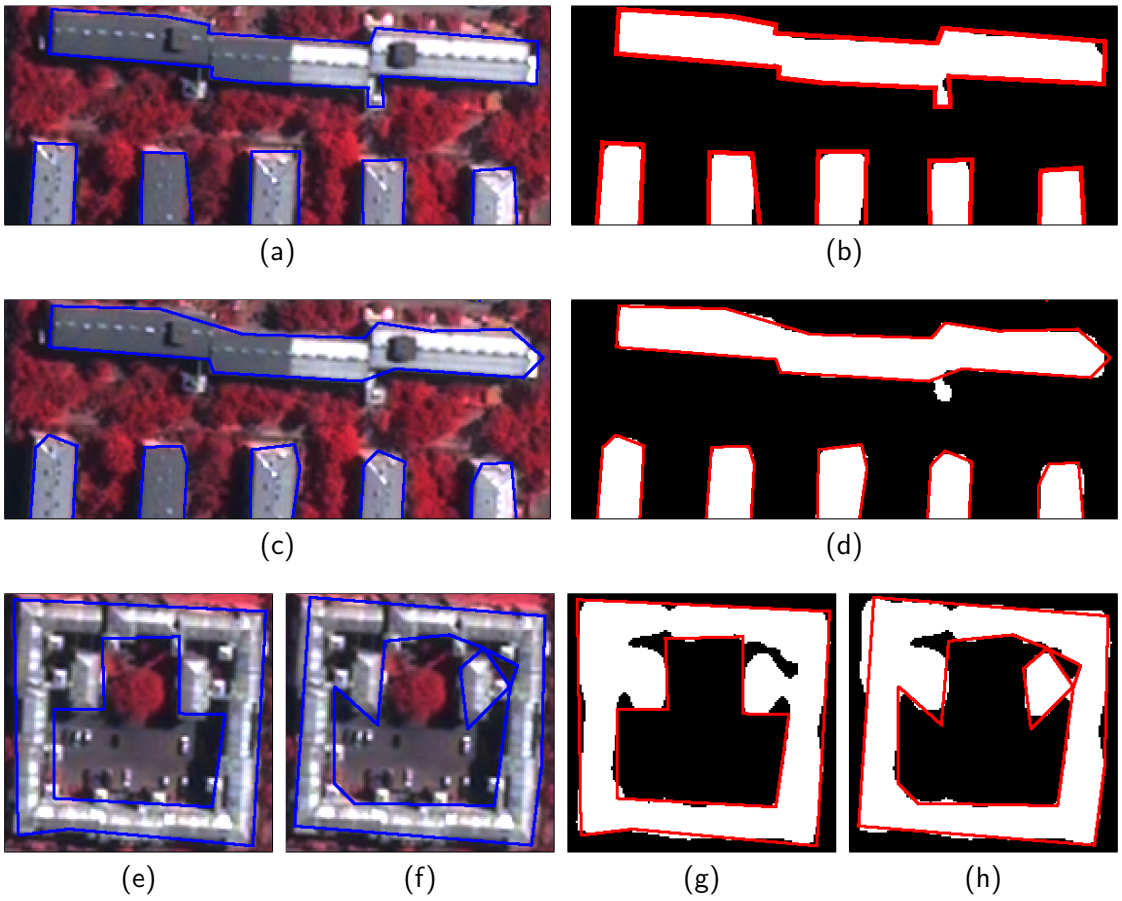


Figure 6.10: Visual comparison of mesh approximation with and without regularity when # of vertices are similar. Input image and classification map are depicted in the left and right column respectively. (a, b, e, g): with, (c, d, f, h) without regularity.

6.6 Experiments

We use a 1180 $\times$ 1030 Pléiades image that has been captured over Santiago and has near infrared, red, and green bands. A ground-truth, in which each pixel is labeled as either building or non-building has been manually prepared. We generate the classification map using the MLP network [112].

We first create a fine lattice by generating a vertex at every 10 pixels. We then apply mesh operators in batches, where each batch fills the queue with one type of operator and applies all the operators in the queue to the mesh. We modify the initial labeled triangle mesh by proceeding with this sequence of batches: edge flip, vertex relocation, half-edge collapse, and vertex relocation.

We set the parameters for reflection, expansion and contraction movements in vertex relocation operator to the following values; $\alpha = 1, \gamma = 2$, and $\beta = 0.5$. We also set # of iterations and area thresholds for the simplex optimization method to 100 and 0.1 respectively. In addition, we ignore relocation movements when magnitude of displacement is below 0.01 pixels.

We compare our approach with commonly used polygon generalization algorithms in GIS applications, namely Radial distance [157], Reumann-Witkam [140], Valingam-Whyatt [178], and Douglas-Peucker [49]. We also provide the results for our approach with and without right angle regularity term. For the generalization algorithms, we vectorize the classification map by using *gdal_polygonize* function of the GDAL library, and simplify the complex vectorized output by the generalization algorithms. We use intersection over union (IoU) between the vectorized classifications and ground-truth for the building class as the performance measurement. We compare the results for different number of vertices by changing the value of λ parameter in Eq. 6.1, and the value of tolerance threshold in generalization methods. IoU vs. number of vertices plots are shown in Fig. 6.9. The plots show that our approach significantly outperforms the others. We also observe that mesh approximation with right angle regularity yields better results than mesh approximation without the regularity term. Example building contours generated by our method with and without regularity term are shown in Fig 6.10. These qualitative results prove that even if building corners are overly rounded in the raster maps generated by the classifier, regular building contours can be delineated with the help of the regularity term defined in Eq. 6.1.

6.7 Concluding Remarks

Especially after convolutional neural networks (CNNs) had revolutionized the computer vision community, many approaches generating high quality pixel-wise maps have been proposed. However, in real-world GIS applications, digitized representations are more commonly used because of their certain advantages over raster maps such as fast processing and easy querying. Hence, it is essential to propose methods that can convert raster maps to vector ones.

In this context, we presented a novel mesh approximation based method. We defined

an objective function that takes into account fidelity to the classification map, right angle regularity, and mesh complexity. Furthermore, we defined two edge based and one vertex based operators that minimizes the objective function to approximate building contours. In our experiments, we demonstrated that the proposed approach outperforms the polygon generalization algorithms commonly used in GIS applications. We also showed that the regularity term allows the algorithm to correct imprecise building contours in the raster map.

The main limitation of our work is that it takes as input only an imperfect raster map. Although we showed that some errors in the raster maps could be eliminated with the help of the regularity term in our objective function, if the input raster map is severely imprecise, our approach may not generate high quality polygons. To overcome this limitation, one can think of using the input image itself in addition to the segmentation to benefit from spectral information. Our objective function is quite flexible, additional terms can easily be added. For instance, we can add another term enforcing standard deviation of the spectral values of the pixels in each triangle to be as small as possible.

Another future direction can be adding a richer set of regularities. For example, other regularities such as parallelism, symmetries, and orbits can be included as well. We must also extend our approach to polygonize other classes. To vectorize classes such as roads, water surfaces, trees, and low vegetation areas, working with higher order of geometric elements like Béziers curves might be useful. The recent curved optimal Delaunay triangulation approach [51] is relevant for this directiosn.

Chapter 7

Conclusions and Perspectives

7.1 Conclusions

With the continuous proliferation and improvement of satellite sensors, numerous new generation satellite missions have been created, which has made it possible to collect huge amounts of data. However, a significant portion of these massive volume of data is unstructured and stored as raw files. The main objective of this thesis was to generate efficient representations for large-scale remote sensing images. Such representations have several potential impacts in many application domains such as monitoring natural disasters, urban planning, autonomous driving, navigation and precise agriculture. To reach this goal, we split the overall problem into two sub-stages. The former task aimed to generate raster maps by performing pixel-wise classification via advanced machine learning methods, and the latter stage consisted in vectorizing the raster maps by computational geometry techniques. In both stages of the overall procedure, we have encountered with various scientific challenges.

The first two challenges were generalization and adaptability. Especially in the field of remote sensing, images collected in different times have significantly different data distributions. Such data distributions usually originate from atmospheric conditions, differences in acquisitions, etc. Although in many benchmarks convolutional neural networks (CNNs) have been proven to be excellent in semantic segmentation, in this thesis, we showed that their success crucially depends on the representativeness of the training data. In other words, they have quite limited generalization abilities. One can overcome this limitation by collecting more training data that would be representative for more diverse data. However, since manual annotations is too costly, it is quite challenging to have training data that are representative for the whole globe. Hence, it is essential to develop novel methods with high generalization capabilities although annotated data are scarce. In addition, oftentimes, we retrieve new annotated and unlabeled images from different locations of the world due to the continuous advancements. Therefore, we wanted to propose methods that can easily adapt to new images.

Another challenge was the heterogeneous annotations. It is often the case that different data set providers and practitioners are interested in maps for separate classes. For

this reason, different data sets obtained in various times tend to have annotations for different classes. Moreover, every time when we receive new data, it is not always possible to store enormous amounts of data. In such a setting, we must develop an incremental learning approach that can learn from the data acquired over the time with annotations for separate classes, without accessing to entire previous training data. Such approach would enable us to generate maps for all the classes that it has partially learned from different data sets.

The main challenge for the vectorization algorithm was to generate efficient vector maps with strong representation power. To this end, it was of paramount importance to obtain maps that precisely delineate object contours with a minimum number of geometric components. We also had to pay attention that both stages of our overall procedure are highly scalable and automated to efficiently process large-scale remote sensing data.

Given the problem definition and the aforementioned challenges, this thesis described several contributions for various machine learning and computational geometry problems. In Chapter 2, we argued that CNNs suffer from catastrophic forgetting: a significant performance drop for the already learned classes when new classes are added on the data having no annotations for the old classes. We proposed an incremental learning methodology enabling to learn segmenting new classes without hindering dense labeling abilities for the previous classes, although the entire previous data are not accessible. The key points of the proposed approach were adapting the network to learn new as well as old classes on new training data, and allowing it to remember the previously learned information for old classes. For adaptation, we kept a frozen copy of the previously trained network, which was used as a memory for the updated network in absence of annotations for the former classes. The updated network minimized a loss function, which balances the discrepancy between outputs for the previous classes from the memory and updated networks, and the mis-classification rate between outputs for the new classes from the updated network and the new ground-truth. For remembering, we regularly fed samples from the stored, little fraction of the previous data.

The major drawback of the incremental learning approach introduced in Chapter 2 is that the memory network generates imprecise maps for old classes on new data, when there exists a large data distribution difference between new and old data. This problem has motivated us to tackle domain adaptation problem. In Chapter 3, we proposed ColorMapGAN and SemI2I for city-to-city adaptation problem. The main objective of both approaches was to generate a target stylized fake source city that is semantically consistent with the original source city. In that chapter, we demonstrated that training a classifier using the fake source city and the ground-truth of the real source city yields a significantly better performance. On the other hand, we discussed that city-to-city adaptation has limited real-world applications, as usually we have many images collected from different geographic locations in the world.

To overcome this limitation, we had to deal with multiple images having largely different data distributions. In this context, we presented StandardGAN in Chapter 4. The core idea behind this approach is to standardize both source and target images by

taking average of their styles. This process was the pre-processing step prior to training a classifier. In the experiments, we proved that training a classifier on the standardized source data and segmenting the standardized target images allows a significant performance gain. Its main drawback is the use of a different style encoder for each image. Although the style encoder is a shallow network, the method is still limited to a certain number of images.

To address this issue, we introduced DAugNet in Chapter 5. It consists of a data augmentor and a classifier. In each training iteration, the data augmentor provides the classifier with diversified data, which makes the classifier robust to large data distribution difference between the domains. Here, the main advantage is that the proposed approach is able to perform style transfer between multiple images with only one encoder, one decoder, and one discriminator irrespective of how many images there are. In our experiments, we showed that mainly because of the data augmentor, DAugNet has a strong generalization ability. In our life-long experiment, we also proved that this approach can efficiently adapt to continuously growing data. In our last ablation study, where we performed style transfer between twenty images, we demonstrated that our solution is scalable.

In Chapter 6, we presented a mesh approximation based vectorization algorithm to vectorize binary classification maps for building class. In this approach, a dense initial mesh was decimated and optimized using local edge and vertex-based operators in order to minimize an objective function that models a balance between fidelity to the classification map, right angle regularity for polygonized buildings, and final mesh complexity. In our experiments, we showed that this approach outperforms common polygon generalization algorithms implemented in GIS applications.

7.2 Perspectives

We now discuss the possible future directions to further improve the methods presented in this thesis.

Although we verified the effectiveness of StandardGAN for domain adaptation, we believe that it is also relevant for other applications such as image mosaicking and change detection. Image mosaicking aims at generating a continuous image by combining adjacent images having different radiometries. The goal of change detection is to find similarities or differences between the images taken from the same locations at different times. As future work, it would be certainly interesting to explore whether StandardGAN is suitable for these problems. To efficiently process large-scale remote sensing images, one could extend StandardGAN by adopting the idea of using randomly initialized style codes as in DAugNet.

Domain generalization and domain adaptation are somewhat related but different problems. The main difference of domain generalization from domain adaptation is that it assumes that any target data are not accessible [98]. In other words, this research problem aims at training a model that is able to segment an arbitrary image given to the model. If this problem is solved, maps for any remote sensing image can be

automatically generated in a very short period of time. The recent domain randomization approach [189] showed that randomizing images using an auxiliary image set that is composed of ImageNet classes [147] is a valid option to solve this problem. As mentioned earlier, the data augmentor in DAugNet diversifies the training batch before passing it to the classifier. As a consequence, the classifier becomes robust to distribution difference between training and test data. As future work, it would be interesting to explore whether DAugNet is applicable for the domain generalization problem. For example, in order for the data augmentor to diversify the training data even more, one can collect all the publicly available images and perform style transfer between them. Another possible future direction might be introducing attention mechanisms [35,120,155,185] to perform style transfer between only the desired classes rather than matching the global distributions of two remote sensing images.

In this thesis, we presented a method for incremental learning and several approaches for various domain adaptation problems. However, we have not combined them. It would also be worth exploring whether our domain adaptation techniques can be incorporated into incremental learning approach. Another interesting direction would be investigating if crowd-source databases such as OpenStreetMap (OSM) could be used for better segmentation. The main challenge for this problem is that acquired remote sensing images and OSM maps usually do not match. Oftentimes, we observe a lot of unlabeled or incorrectly delineated objects. It is worth conducting research on the domain of weakly supervised learning [30] to benefit from such databases.

In the future, our vectorization approach definitely needs to be extended to other classes in addition to building class. Adding a set of richer geometric regularities such as parallelism, symmetries, and orbits can also be considered. To vectorize objects with irregular shapes such as trees and water surfaces, introducing higher order of geometric elements such as Béziers curves is relevant [51]. Besides, additional information such as the input satellite image itself would be certainly beneficial. For this direction, one can be inspired by the KIPPI approach that utilizes a kinetic computational geometry framework [9].

Bibliography

[1] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient Interactive Annotation of Segmentation Datasets with Polygon-RNN++. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–868, 2018.

[2] Yehuda Afek and Ariel Brand. Mosaicking of Orthorectified Aerial Images. *Photogrammetric Engineering and Remote Sensing*, 64(2):115–124, 1998.

[3] Vivek Agarwal, Besma R Abidi, Andreas Koschan, and Mongi A Abidi. An Overview of Color Constancy Algorithms. *Journal of Pattern Recognition Research*, 1(1):42–54, 2006.

[4] Selim Aksoy and Robert M Haralick. Feature Normalization and Likelihood-Based Similarity Measures for Image Retrieval. *Pattern Recognition Letters*, 22(5):563–582, 2001.

[5] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning What (not) to Forget. In *Proceedings of the European Conference on Computer Vision*, pages 139–154, 2018.

[6] Luciano Alparone, Lucien Wald, Jocelyn Chanussot, Claire Thomas, Paolo Gamba, and Lori Mann Bruce. Comparison of Pansharpening Algorithms: Outcome of the 2006 GRS-S Data-Fusion Contest. *IEEE Transactions on Geoscience and Remote Sensing*, 45(10):3012–3021, 2007.

[7] Israa Amro, Javier Mateos, Miguel Vega, Rafael Molina, and Aggelos K Katsaggelos. A Survey of Classical Methods and New Trends in Pansharpening of Multispectral Images. *EURASIP Journal on Advances in Signal Processing*, 2011(1):1–22, 2011.

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv*, 2017.

[9] Jean-Philippe Bauchet and Florent Lafarge. KIPPI: Kinetic Polygonal Partitioning of Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2018.

[10] Mariana Belgiu and Lucian Drăguţ. Comparing Supervised and Unsupervised Multiresolution Segmentation Approaches for Extracting Buildings from Very High Resolution Imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 96:67–75, 2014.

[11] Jon Atli Benediktsson, Martino Pesaresi, and Kolbeinn Amason. Classification and Feature Extraction For Remote Sensing Images from Urban Areas Based on Morphological Transformations. *IEEE Transactions on Geoscience and Remote Sensing*, 41(9):1940–1949, 2003.

[12] Jon Atli Benediktsson, Johannes R Sveinsson, Okan K Ersoy, and Philip H Swain. Parallel Consensual Neural Networks. *IEEE Transactions on Neural Networks*, 8(1):54–64, 1997.

[13] Bilel Benjdira, Yakoub Bazi, Anis Koubaa, and Kais Ouni. Unsupervised Domain Adaptation Using Generative Adversarial Networks for Semantic Segmentation of Aerial Images. *Remote Sensing*, 11(11):1369, 2019.

[14] Frank J Bossen and Paul S Heckbert. A Pliant Method for Anisotropic Mesh Generation. In *5th Intl. Meshing Roundtable*, pages 63–74. Citeseer, 1996.

[15] Mario Botsch and Leif Kobbelt. A Robust Procedure to Eliminate Degenerate Faces from Triangle Meshes. In *VMV*, pages 283–290, 2001.

[16] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing*. CRC press, 2010.

[17] Leo Breiman. Random Forests. *Machine learning*, 45(1):5–32, 2001.

[18] Lorenzo Bruzzone, Mingmin Chi, and Mattia Marconcini. A Novel Transductive SVM for Semisupervised Classification of Remote-sensing Images. *IEEE Transactions on Geoscience and Remote Sensing*, 44(11):3363–3373, 2006.

[19] Lorenzo Bruzzone and Roberto Cossu. A Multiple-Cascade-Classifier System for a Robust and Partially Unsupervised Updating of Land-Cover Maps. *IEEE Transactions on Geoscience and Remote Sensing*, 40(9):1984–1996, 2002.

[20] Lorenzo Bruzzone and Mattia Marconcini. Domain Adaptation Problems: A DASVM Classification Technique and a Circular Validation Strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):770–787, 2009.

[21] Lorenzo Bruzzone and D Fernandez Prieto. An Incremental-Learning Beural Network for the Classification of Remote-Sensing Images. *Pattern Recognition Letters*, 20(11-13):1241–1248, 1999.

[22] Lorenzo Bruzzone and Diego Fernàndez Prieto. Unsupervised Retraining of a Maximum Likelihood Classifier for the Analysis of Multitemporal Remote Sensing Images. *IEEE Transactions on Geoscience and Remote Sensing*, 39(2):456–460, 2001.

[23] Lorenzo Bruzzone and Diego Fernández Prieto. A Partially Unsupervised Cascade Classifier for the Analysis of Multitemporal Remote-Sensing Images. *Pattern Recognition Letters*, 23(9):1063–1071, 2002.

[24] Gershon Buchsbaum. A Spatial Processor Model for Object Colour Perception. *Journal of the Franklin Institute*, 1980.

[25] Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and Flexible Image Augmentations. *Information*, 11(2):125, 2020.

[26] Rich Caruana. Multitask Learning. *Machine learning*, 28(1):41–75, 1997.

[27] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating Object Instances with a Polygon-RNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5230–5238, 2017.

[28] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-End Incremental Learning. In *Proceedings of the European Conference on Computer Vision*, pages 233–248, 2018.

[29] Gert Cauwenberghs and Tomaso Poggio. Incremental and Decremental Support Vector Machine Learning. In *Advances In Neural Information Processing Systems*, pages 409–415, 2001.

[30] Lyndon Chan, Mahdi S Hosseini, and Konstantinos N Plataniotis. A Comprehensive Analysis of Weakly-Supervised Semantic Segmentation in Different Image Domains. *arXiv preprint arXiv:1912.11186*, 2019.

[31] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *arXiv preprint arXiv:1412.7062*, 2014.

[32] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2017.

[33] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[34] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[35] Xinyuan Chen, Chang Xu, Xiaokang Yang, and Dacheng Tao. Attention-GAN for Object Transfiguration in Wild Images. In *Proceedings of the European Conference on Computer Vision*, pages 164–180, 2018.

[36] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

[37] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse Image Synthesis for Multiple Domains. *arXiv preprint arXiv:1912.01865*, 2019.

[38] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column Deep Neural Networks for Image Classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.

[39] Laurent D Cohen. On Active Contour Models and Balloons. *CVGIP: Image understanding*, 53(2):211–218, 1991.

[40] René Roland Colditz. An Evaluation of Different Training Sample Allocation Schemes for Discrete and Continuous Land Cover Classification Using Decision Tree-based Algorithms. *Remote Sensing*, 7(8):9655–9681, 2015.

[41] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine learning*, 20(3):273–297, 1995.

[42] Gabriela Csurka, Diane Larlus, and Florent Perronnin. What is a Good Evaluation Measure for Semantic Segmentation?. In *The British Machine Vision Conference*, volume 27, page 2013. Citeseer, 2013.

[43] Haskell B Curry. The Method of Steepest Descent for Non-Linear Minimization Problems. *Quarterly of Applied Mathematics*, 2(3):258–261, 1944.

[44] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Schwarzkopf. Computational Geometry. In *Computational Geometry*, pages 1–17. Springer, 1997.

[45] Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raska. Deepglobe 2018: A Challenge to Parse the Earth Through Satellite Images. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 172–17209. IEEE, 2018.

[46] Cheng Deng, Xianglong Liu, Chao Li, and Dacheng Tao. Active Multi-Kernel Domain Adaptation for Hyperspectral Image Classification. *Pattern Recognition*, 77:306–315, 2018.

[47] Chengbin Deng and Changshan Wu. The Use of Single-date MODIS Imagery for Estimating Large-scale Urban Impervious Surface Fraction with Spectral Mixture Analysis and Machine Learning Techniques. *ISPRS Journal of Photogrammetry and Remote Sensing*, 86:100–110, 2013.

[48] Xueqing Deng, Hsiuhan Lexie Yang, Nikhil Makkar, and Dalton Lunga. Large Scale Unsupervised Domain Adaptation of Segmentation Networks with Adversarial Learning. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 4955–4958. IEEE, 2019.

[49] David H Douglas and Thomas K Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[50] Ahmed Elshamli, Graham W Taylor, and Shawki Areibi. Multisource Domain Adaptation for Remote Sensing Using Deep Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 2019.

[51] Leman Feng, Pierre Alliez, Laurent Busé, Hervé Delingette, and Mathieu Desbrun. Curved Optimal Delaunay Triangulation. *ACM Transactions on Graphics*, 37(4):61, 2018.

[52] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. *arXiv preprint arXiv:1701.08734*, 2017.

[53] David A. Forsyth. A Novel Algorithm for Color Constancy. *International Journal of Computer Vision*, 5(1):5–35, 1990.

[54] Robert M French. Catastrophic Forgetting in Connectionist Networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[55] Tommaso Furlanello, Jiaping Zhao, Andrew M Saxe, Laurent Itti, and Bosco S Tjan. Active Long Term Memory Networks. *arXiv preprint arXiv:1606.02355*, 2016.

[56] Martin Galanda. Automated Polygon Generalization in a Multi Agent System. *PhD. Dissertation, University of Zurich, Switzerland*, 2003.

[57] Sina Ghassemi, Attilio Fiandrotti, Gianluca Francini, and Enrico Magli. Learning and Adapting Robust Features for Satellite Image Segmentation on Heterogeneous Data Sets. *IEEE Transactions on Geoscience and Remote Sensing*, 57(9):6517–6529, 2019.

[58] Behnam Gholami, Pritish Sahu, Ognjen Rudovic, Konstantinos Bousmalis, and Vladimir Pavlovic. Unsupervised Multi-Target Domain Adaptation: An Information Theoretic Approach. *IEEE Transactions on Image Processing*, 29:3993–4002, 2020.

[59] Xavier Glorot and Yoshua Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[60] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[61] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[62] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv preprint arXiv:1312.6211*, 2013.

[63] Wolfgang Gross, Devis Tuia, Uwe Soergel, and Wolfgang Middelmann. Nonlinear Feature Normalization for Hyperspectral Domain Adaptation and Mitigation of Nonlinear Effects. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8):5975–5990, 2019.

[64] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved Training of Wasserstein GANS. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[65] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610–621, 1973.

[66] Ali Hatamizadeh, Debleena Sengupta, and Demetri Terzopoulos. End-to-End Deep Convolutional Active Contours for Image Segmentation. *arXiv preprint arXiv:1909.13359*, 2019.

[67] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[69] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*, 2015.

[70] Guilherme Hoefel and Charles Elkan. Learning a Two-stage SVM/CRF sequence classifier. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 271–278, 2008.

[71] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[72] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. FCNs in the wild: Pixel-Level Adversarial and Constraint-Based Adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

[73] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Lifelong Learning via Progressive Distillation and Retrospection. In *Proceedings of the European Conference on Computer Vision*, pages 437–452, 2018.

[74] George Hripcsak and Adam S Rothschild. Agreement, the F-measure, and Reliability in Information Retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298, 2005.

[75] Bohao Huang, Kangkang Lu, Nicolas Audeberr, Andrew Khalel, Yuliya Tarabalka, Jordan Malof, Alexandre Boulch, Bertr Le Saux, Leslie Collins, Kyle Bradbury, et al. Large-scale Semantic Classification: Outcome of the First Year of Inria Aerial Image Labeling Benchmark. In *International Geoscience and Remote Sensing Symposium*, pages 6947–6950. IEEE, 2018.

[76] Haoshuo Huang, Qixing Huang, and Philipp Krahenbuhl. Domain Transfer Through Deep Activation Matching. In *Proceedings of the European Conference on Computer Vision*, pages 590–605, 2018.

[77] Xun Huang and Serge Belongie. Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[78] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision*, pages 172–189, 2018.

[79] IEEE. *StandardGAN: Multi-source Domain Adaptation for Semantic Segmentation of Very High Resolution Satellite Images by Data Standardization*, 2020.

[80] Vladimir Iglovikov, Selim S Seferbekov, Alexander Buslaev, and Alexey Shvets. TernausNetV2: Fully Convolutional Network for Instance Segmentation. In *CVPR Workshops*, volume 233, page 237, 2018.

[81] Vladimir Iglovikov and Alexey Shvets. TernausNet: U-net with vgg11 Encoder Pre-trained on ImageNet for Image Segmentation. *arXiv preprint arXiv:1801.05746*, 2018.

[82] Shilpa Inamdar, Francesca Bovolo, Lorenzo Bruzzone, and Subhasis Chaudhuri. Multidimensional Probability Density Function Matching for Preprocessing of Multitemporal Remote Sensing Images. *IEEE Transactions on Geoscience and Remote Sensing*, 46(4):1243–1252, 2008.

[83] Jordi Inglada, Arthur Vincent, Marcela Arias, Benjamin Tardy, David Morin, and Isabel Rodes. Operational High Resolution Land Cover Map Production at the Country Scale Using Satellite Image Time Series. *Remote Sensing*, 9(1):95, 2017.

[84] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-Image Translation with Conditional Adversarial Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[85] ISPRS. 2d semantic labeling contest. `http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html`.

[86] Klaus I Itten and Peter Meyer. Geometric and Radiometric Correction of TM Data of Mountainous Forested Areas. *IEEE Transactions on Geoscience and Remote Sensing*, 31(4):764–770, 1993.

[87] San Jiang and Wanshou Jiang. UAV-Based Oblique Photogrammetry for 3D Reconstruction of Transmission Line: Practices and Applications. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2019.

[88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active Contour Models. *International journal of computer vision*, 1(4):321–331, 1988.

[89] Andrew Khalel, Onur Tasar, Guillaume Charpiat, and Yuliya Tarabalka. Multi-Task Deep Learning for Satellite Image Pansharpening and Segmentation. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 4869–4872. IEEE, 2019.

[90] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[91] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[92] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[93] Tzu-Sheng Kuo, Keng-Sen Tseng, Jia-Wei Yan, Yen-Cheng Liu, and Yu-Chiang Frank Wang. Deep Aggregation Net for Land Cover Classification. In *CVPR Workshops*, pages 252–256, 2018.

[94] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[95] Der-Tsai Lee and Franco P Preparata. An Optimal Algorithm for Finding the Kernel of a Polygon. *Journal of the ACM*, 26(3):415–421, 1979.

[96] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European Conference on Computer Vision*, pages 35–51, 2018.

[97] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming Catastrophic Forgetting by Incremental Moment Matching. In *Advances in Neural Information Processing Systems*, pages 4652–4662, 2017.

[98] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain Generalization with Adversarial Feature Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[99] Yu Li, Zhongxiao Li, Lizhong Ding, Yijie Pan, Chao Huang, Yuhui Hu, Wei Chen, and Xin Gao. SupportNet: Solving Catastrophic Forgetting in Class Incremental Learning with Support Data. *arXiv preprint arXiv:1806.02942*, 2018.

[100] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated Graph Sequence Neural Networks. *arXiv preprint arXiv:1511.05493*, 2015.

[101] Zhizhong Li and Derek Hoiem. Learning Without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

[102] Zuoyue Li, Jan Dirk Wegner, and Aurélien Lucchi. Topological Map Extraction from Overhead Images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1715–1724, 2019.

[103] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[104] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[105] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate Your Networks: Better Weight Consolidation and Less Catastrophic Forgetting. In *24th International Conference on Pattern Recognition*, pages 2262–2268. IEEE, 2018.

[106] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[107] David Lopez-Paz et al. Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[108] Dengsheng Lu, Paul Mausel, Eduardo Brondizio, and Emilio Moran. Change Detection Techniques. *International Journal of Remote Sensing*, 25(12):2365–2401, 2004.

[109] Li Ma, Melba M Crawford, Lei Zhu, and Yong Liu. Centroid and Covariance Alignment-Based Domain Adaptation for Unsupervised Classification of Remote Sensing Images. *IEEE Transactions on Geoscience and Remote Sensing*, 57(4):2305–2323, 2018.

[110] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):645–657, 2016.

[111] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 3226–3229, 2017.

[112] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. High-Resolution Aerial Image Labeling with Convolutional Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(12):7092–7103, 2017.

[113] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Polygonization of Remote sensing Classification Maps by Mesh Approximation. In *IEEE International Conference on Image Processing*, pages 560–564. IEEE, 2017.

[114] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights. In *Proceedings of the European Conference on Computer Vision*, pages 67–82, 2018.

[115] Arun Mallya and Svetlana Lazebnik. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[116] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least Squares Generative Adversarial Networks. In *IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[117] Diego Marcos, Devis Tuia, Benjamin Kellenberger, Lisa Zhang, Min Bai, Renjie Liao, and Raquel Urtasun. Learning Deep Structured Active Contours End-to-End. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8877–8885, 2018.

[118] Giona Matasci, Devis Tuia, and Mikhail Kanevski. SVM-Based Boosting of Active Learning Strategies for Efficient Domain Adaptation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(5):1335–1343, 2012.

[119] Gellért Máttyus, Wenjie Luo, and Raquel Urtasun. DeepRoadMapper: Extracting Road Topology from Aerial Images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3438–3446, 2017.

[120] Youssef Alami Mejjati, Christian Richardt, James Tompkin, Darren Cosker, and Kwang In Kim. Unsupervised Attention-Guided Image-to-Image Translation. In *Advances in Neural Information Processing Systems*, pages 3693–3703, 2018.

[121] Farid Melgani and Lorenzo Bruzzone. Classification of Hyperspectral Remote Sensing Images with Support Vector Machines. *IEEE Transactions on Geoscience and Remote Sensing*, 42(8):1778–1790, 2004.

[122] Xiangchao Meng, Huanfeng Shen, Huifang Li, Liangpei Zhang, and Randi Fu. Review of the Pansharpening Methods for Remote Sensing Images Based on the Idea of Meta-Analysis: Practical Discussion and Challenges. *Information Fusion*, 46:102–113, 2019.

[123] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*, 2014.

[124] Volodymyr Mnih. *Machine Learning for Aerial Image Labeling*. Citeseer, 2013.

[125] John A Nelder and Roger Mead. A Simplex Method for Function Minimization. *The computer journal*, 1965.

[126] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational Continual Learning. *arXiv preprint arXiv:1710.10628*, 2017.

[127] Xuan Nie, Mengyang Duan, Haoxuan Ding, Bingliang Hu, and Edward K Wong. Attention Mask R-CNN for Ship Detection and Segmentation from Remote Sensing Images. *IEEE Access*, 8:9325–9334, 2020.

[128] Fabio Pacifici, Nathan Longbotham, and William J Emery. The Importance of Physical Quantities for the Analysis of Multitemporal and Multiangular Optical Very High Spatial Resolution Images. *IEEE Transactions on Geoscience and Remote Sensing*, 52(10):6241–6256, 2014.

[129] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at Once: Enhancing Learning and Generalization Capacities via IBN-Net. In *Proceedings of the European Conference on Computer Vision*, pages 464–479, 2018.

[130] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment Matching for Multi-Source Domain Adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.

[131] Massimiliano Pepe and Giuseppina Prezioso. Two Approaches for Dense DSM Generation from Aerial Digital Oblique Camera System. In *GISTAM*, pages 63–70, 2016.

[132] Luigi Perotti, Manuela Lasagna, Paolo Clemente, Giovanna Antonella Dino, and Domenico Antonio De Luca. Remote Sensing and Hydrogeological Methodologies for Irrigation Canal Water Losses Detection: the Naviglio di Bra Test Site (NW-Italy). In *Engineering Geology for Society and Territory-Volume 3*, pages 19–22. Springer, 2015.

[133] Claudio Persello and Lorenzo Bruzzone. Active Learning for Domain Adaptation in the Supervised Classification of Remote Sensing Images. *IEEE Transactions on Geoscience and Remote Sensing*, 50(11):4468–4483, 2012.

[134] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An Incremental Learning Algorithm for Supervised Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(4):497–508, 2001.

[135] Yao Qin, Lorenzo Bruzzone, and Biao Li. Tensor Alignment Based Domain Adaptation for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 57(11):9290–9307, 2019.

[136] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434*, 2015.

[137] Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder Based Lifelong Learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1320–1328, 2017.

[138] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental Classifier and Representation Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[139] Ievgen Redko, Nicolas Courty, Rémi Flamary, and Devis Tuia. Optimal Transport for Multi-Source Domain Adaptation Under Target Shift. *arXiv preprint arXiv:1803.04899*, 2018.

[140] Kurt Reumann and A. P. M. Witkam. Optimizing Curve Segmentation in Computer Graphics. In *Proceedings of the International Computing Symposium*, 1974.

[141] David S Richeson. *Euler's Gem: the Polyhedron Formula and the Birth of Topology*. Princeton University Press, 2012.

[142] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting. In *Advances in Neural Information Processing Systems*, pages 3738–3748, 2018.

[143] Rob Romijnders, Panagiotis Meletis, and Gijs Dubbelman. A Domain Agnostic Normalization Layer for Unsupervised Adversarial Domain Adaptation. In *Winter Conference on Applications of Computer Vision*, pages 1866–1875. IEEE, 2019.

[144] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[145] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut" Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004.

[146] Sebastian Ruder. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098*, 2017.

[147] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[148] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *arXiv preprint arXiv:1606.04671*, 2016.

[149] Alan Saalfeld. Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm. *Cartography and Geographic Information Science*, 26(1):7–18, 1999.

[150] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Adversarial Dropout Regularization. *arXiv preprint arXiv:1711.01575*, 2017.

[151] Syed Shakib Sarwar, Aayush Ankit, and Kaushik Roy. Incremental Learning in Deep Convolutional Neural Networks Using Partial Network Sharing. *IEEE Access*, 2019.

[152] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[153] Robert A. Schowengerdt. *Remote Sensing: Models and Methods for Image Processing*. Elsevier, 2006.

[154] Huanfeng Shen, Xiangchao Meng, and Liangpei Zhang. An Integrated Framework for the Spatio–Temporal–Spectral Fusion of Remote Sensing Images. *IEEE Transactions on Geoscience and Remote Sensing*, 54(12):7135–7148, 2016.

[155] Zhiqiang Shen, Mingyang Huang, Jianping Shi, Xiangyang Xue, and Thomas S Huang. Towards Instance-Level Image-to-Image Translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3683–3692, 2019.

[156] Alex Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *arXiv preprint arXiv:1808.03314*, 2018.

[157] Wenzhong Shi and ChuiKwan Cheung. Performance Evaluation of Line Simplification Algorithms for Vector Generalization. *The Cartographic Journal*, 43(1):27–44, 2006.

[158] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[159] Alexey A Shvets, Vladimir I Iglovikov, Alexander Rakhlin, and Alexandr A Kalinin. Angiodysplasia Detection and Localization Using Deep Convolutional Neural Networks. In *17th IEEE International Conference on Machine Learning and Applications*, pages 612–617. IEEE, 2018.

[160] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[161] Hao Su, Shunjun Wei, Min Yan, Chen Wang, Jun Shi, and Xiaoling Zhang. Object Detection and Instance Segmentation in Remote Sensing Imagery Based on Precise Mask R-CNN. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 1454–1457. IEEE, 2019.

[162] Héctor J Sussmann. Uniqueness of the Weights for Minimal Feedforward Nets with a Given Input-Output Map. *Neural networks*, 5(4):589–593, 1992.

[163] Yuliya Tarabalka, Mathieu Fauvel, Jocelyn Chanussot, and Jón Atli Benediktsson. SVM-and MRF-based Method for Accurate Classification of Hyperspectral Images. *IEEE Geoscience and Remote Sensing Letters*, 7(4):736–740, 2010.

[164] Yuliya Tarabalka and Aakanksha Rana. Graph-cut-based Model for Spectral-spatial Classification of Hyperspectral Images. In *2014 IEEE Geoscience and Remote Sensing Symposium*, pages 3418–3421. IEEE, 2014.

[165] Onur Taşar and Selim Aksoy. Object Detection Using Optical and Lidar Data Fusion. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 7204–7207. IEEE, 2016.

[166] Onur Tasar, Alain Giros, Yuliya Tarabalka, Pierre Alliez, and Sébastien Clerc. DAugNet: Unsupervised, Multi-source, Multi-target, and Life-long Domain Adaptation for Semantic Segmentation of Satellite Images. *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[167] Onur Tasar, S L Happy, Yuliya Tarabalka, and Pierre Alliez. ColorMapGAN: Unsupervised Domain Adaptation for Semantic Segmentation Using Color Mapping Generative Adversarial Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[168] Onur Tasar, S L Happy, Yuliya Tarabalka, and Pierre Alliez. SemI2I: Semantically Consistent Image-to-Image Translation for Domain Adaptation of Remote Sensing Data. *IEEE International Geoscience and Remote Sensing Symposium*, 2020.

[169] Onur Tasar, Emmanuel Maggiori, Pierre Alliez, and Yuliya Tarabalka. Polygonization of binary classification maps using mesh approximation with right angle regularity. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 6404–6407. IEEE, 2018.

[170] Onur Tasar, Yuliya Tarabalka, and Pierre Alliez. Continual Learning for Dense Labeling of Satellite Images. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 4943–4946. IEEE, 2019.

[171] Onur Tasar, Yuliya Tarabalka, and Pierre Alliez. Incremental Learning for Semantic Segmentation of Large-Scale Remote Sensing Data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(9):3524–3537, 2019.

[172] Sebastian Thrun. Is Learning the N-th Thing Any Easier Than Learning the First? In *Advances in Neural Information Processing Systems*, pages 640–646, 1996.

[173] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to Adapt Structured Output Space for Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7472–7481, 2018.

[174] Devis Tuia, Jordi Munoz-Mari, Luis Gómez-Chova, and Jesus Malo. Graph Matching for Adaptation in Remote Sensing. *IEEE Transactions on Geoscience and Remote Sensing*, 51(1):329–341, 2012.

[175] Devis Tuia, Claudio Persello, and Lorenzo Bruzzone. Domain Adaptation for the Classification of Remote Sensing Data: An Overview of Recent Advances. *IEEE Geoscience and Remote Sensing Magazine*, 4(2):41–57, 2016.

[176] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[177] Adam Van Etten, Dave Lindenbaum, and Todd M Bacastow. SpaceNet: A Remote Sensing Dataset and Challenge Series. *arXiv preprint arXiv:1807.01232*, 2018.

[178] Maheswari Visvalingam and James Duncan Whyatt. Line Generalisation by Repeated Elimination of the Smallest Area. *Cartographic Journal*, 30(1):46–51, 1992.

[179] Shin-Ting Wu and Mercedes Rocio Gonzales Marquez. A Non-Self-Intersection Douglas-Peucker Algorithm. In *16th Brazilian Symposium on Computer Graphics and Image Processing*, pages 60–66. IEEE, 2003.

[180] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, Zhengyou Zhang, and Yun Fu. Incremental Classifier Learning with Generative Adversarial Networks. *arXiv preprint arXiv:1802.00853*, 2018.

[181] Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, Yuxin Peng, and Zheng Zhang. Error-Driven Incremental Learning in Deep Convolutional Neural Network for Large-Scale Image Classification. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 177–186. ACM, 2014.

[182] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv preprint arXiv:1505.00853*, 2015.

[183] Ju Xu and Zhanxing Zhu. Reinforced Continual Learning. In *Advances in Neural Information Processing Systems*, pages 899–908, 2018.

[184] Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. Deep Cocktail Network: Multi-Source Unsupervised Domain Adaptation with Category Shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3964–3973, 2018.

[185] Chao Yang, Taehwan Kim, Ruizhe Wang, Hao Peng, and C-C Jay Kuo. Show, Attend, and Translate: Unsupervised Image Translation With Self-Regularization and Attention. *IEEE Transactions on Image Processing*, 28(10):4845–4856, 2019.

[186] Bayya Yegnanarayana. *Artificial Neural Networks*. PHI Learning Pvt. Ltd., 2009.

[187] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong Learning with Dynamically Expandable Networks. *arXiv preprint arXiv:1708.01547*, 2017.

[188] Huanhuan Yu, Menglei Hu, and Songcan Chen. Multi-Target Unsupervised Domain Adaptation Without Exactly Shared Categories. *arXiv preprint arXiv:1809.00852*, 2018.

[189] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain Randomization and Pyramid Consistency: Simulation-to-Real Generalization Without Accessing Target Domain Data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2100–2110, 2019.

[190] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *Proceedings of the European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[191] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org, 2017.

[192] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task Agnostic Continual Learning Using Online Variational Bayes. *arXiv preprint arXiv:1803.10123*, 2018.

[193] Junting Zhang, Chen Liang, and C-C Jay Kuo. A Fully Convolutional Tri-Branch Network (FCTN) for Domain Adaptation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3001–3005. IEEE, 2018.

[194] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial Multiple Source Domain Adaptation. In *Advances in Neural Information Processing Systems*, pages 8559–8570, 2018.

[195] Kang Zhao, Jungwon Kang, Jaewook Jung, and Gunho Sohn. Building Extraction From Satellite Images Using Mask R-CNN With Building Boundary Regularization. In *CVPR Workshops*, pages 247–251, 2018.

[196] Sicheng Zhao, Bo Li, Xiangyu Yue, Yang Gu, Pengfei Xu, Runbo Hu, Hua Chai, and Kurt Keutzer. Multi-Source Domain Adaptation for Semantic Segmentation. In *Advances in Neural Information Processing Systems*, pages 7285–7298, 2019.

[197] Lichen Zhou, Chuang Zhang, and Ming Wu. D-LinkNet: LinkNet With Pretrained Encoder and Dilated Convolution for High Resolution Satellite Imagery Road Extraction. In *CVPR Workshops*, pages 182–186, 2018.

[198] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.

[199] Xinge Zhu, Hui Zhou, Ceyuan Yang, Jianping Shi, and Dahua Lin. Penalizing Top Performers: Conservative Loss for Semantic Segmentation Adaptation. In *Proceedings of the European Conference on Computer Vision*, pages 568–583, 2018.

[200] Yang Zou, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang. Unsupervised Domain Adaptation for Semantic Segmentation via Class-Balanced Self-Training. In *Proceedings of the European Conference on Computer Vision*, pages 289–305, 2018.