Master thesis

Deep domain adaptation for semantic segmentation of remotely sensed images based on activation distribution matching

Yunshuang Yuan, 10007354

Supervisor:

Prof. Franz Rottensteiner(IPI)

M.Sc. Dennis Wittich(IPI)

M.Sc. Christian Poss(BMW)

Institute of Photogrammetry and Geoinformation

Leibniz Universität Hannover

Hannover, March 16, 2020

Institut für Photogrammetrie
und GeoInformation

Leibniz
Universität
Hannover

Institut für Photogrammetrie und GeoInformation,
Nienburger Straße 1, 30167 Hannover

Nienburger Straße 1, 30167 Hannover
Fakultät für Bauingenieurwesen
und Geodäsie

Institut für Photogrammetrie
und GeoInformation

Prof. Dr.-habil. Christian Heipke

apl. Prof. Dr. techn. Franz
Rottensteiner

Tel.+49 511 762-3893
Fax +49 511 762-2483
E-Mail: rottensteiner
@ipi.uni-hannover.de

15. September 2019

Topic of the M.Sc. Thesis of Ms. Yunshuang Yuan B.Sc.

Deep domain adaptation for semantic segmentation of remotely sensed images based on activation distribution matching
(working title)

The semantic segmentation of images is a well-known task in computer vision and remote sensing as a first step to detect objects and their precise delineation in the images. For the last years, Deep Neural Networks (DNN) have been adapted to that task, surpassing the results of classical machine learning approaches. However, DNN for segmentation, usually in form of Fully Convolutional Neural Networks (FCNN), require a large amount of training data in order to learn the complex mapping required to solve the respective tasks. One approach to counteract this problem is Domain Adaptation (DA). Here, unlabelled samples from a (new) target domain are used to adapt the model trained on existing (source domain) data before applying it to infer the labels in the target domain. Consequently, re-using the existing (source) data, no new labelled samples are required in the target domain. Although there are many approaches to use DA in combination of FCNN, it remains unclear which approaches are best suitable for special applications e.g. the classification of images in combination with depth or height data.

Consequently, it is the goal of the M.Sc. thesis of Ms. Yuan to compare the results of different existing approaches for DA when applied to multiple datasets with varying domain differences. In particular, Ms. Yuan should investigate whether approaches such as minimizing the maximum mean discrepancy between source and target domain activations or performing a correlation alignment can be used to adapt models between domains for both indoor and

Besucheradresse:
Nienburger Straße 1
30167 Hannover
www.ipi.uni-hannover.de

remote sensing applications. These approaches are based on matching the first and second order distributions, respectively, of source and target domain layer activations. Ideally this leads to a representation alignment resulting in higher quality of the semantic segmentation when applying the model on the target domain.

This task includes the implementation of a segmentation network as well as the training of this network on the respective source domain. Next, the approaches for DA mentioned above have to be implemented and possibly adapted to work with remotely sensed images and height data. Finally, Ms. Yuan has to perform an evaluation of the methods using reference data to assess and compare the applicability of these approaches to the field of remote sensing. For that purpose, she can use several available reference datasets, an indoor dataset provided by BWM and various remote sensing datasets. The remote sensing datasets can be used to create multiple DA scenarios, where data from one city are used to train a model (source domain) and the data from another city act as target domain.

Based on these datasets, Ms. Yuan shall also investigate how large the domain differences may be for the implemented approaches still to work, and she shall document under which circumstances, e.g. different seasons, different building densities or building sizes or a different label distribution, the methods fail.

apl. Prof. Dr. techn. Franz Rottensteiner / M.Sc. Dennis Wittich

Deep domain adaptation for semantic segmentation of remotely sensed images based on activation distribution matching

Yunshuang Yuan

Abstract

Domain adaptation(DA) is a very important topic in machine learning because labelling data manually is very expensive especially for semantic segmentation. DA is essential to leverage labelled data in the source domain for better performance of classification in the target domain. This thesis addresses DA for semantic segmentation which is rarely researched but very necessary in the field of remote sensing. A computational efficient fully convolutional neural network(FCN) is proposed to solve the classification task in this study. Two statistic-based methods, Maximum Mean Discrepancy(MMD) and *deep Coral*, are introduced to deal with deep DA and are the key points of this thesis. They were rarely used before for semantic segmentation. However, they are possible to decrease the distribution discrepancy between source and target representations. Experiments are conducted where activation distributions of an intermediate layer in the encoder of FCN are matched with both methods. The results show that *deep Coral* performs better than MMD. Based on the model after *deep Coral* adaptation on the intermediate layer, additional small accuracy gain can be obtained by further aligning the output logits layer with *deep Coral*. At last, appearance adaptations between synthetic and real dataset and bewteen two remote sensing datasets are performed using Cycle GAN. It shows that it can not guarantee the mitigation of the distribution discrepancy problem of feature representations between two remote sensing datasets while it can dramatically decrease the discrepancy between synthetic and real dataset.

Keywords

Domain adaptation, Semantic segmentation, Remote Sensing, Maximum Mean Discrepancy, Deep Coral, Cycle GAN, Convolutional Neural Network.

Statement

Contents

1. Introduction

With the development of Deep Neural Networks(DNN), research on computer vision has achieved immense successes based on this technique. DNN can dramatically surpass the accuracy limit of traditional methods like SVM[22] and Random Forest[30]. Since Imagenet[18], a large visual database with more than 14 million images designed for visual object recognition, was launched, the research direction of deep learning on computer vision began to thrive. From the famous AlexNet [38] to ResNet[29], ImageNet classification error(top 5) has been decreased from 16.4% to 3.6%. Along with DNN based image classification, other important computer vision tasks like object detection and semantic segmentation have also drawn great attention. In addition to the rapid development of the hardware for deep learning, another important factor for the great progress made in this field is the availability of numerous datasets like COCO[43], Pascal VOC[21], Cityscapes[15] and KITTI[24] with a huge amount of images and the corresponding labels for different tasks. However, in real applications, annotating images is very expensive, especially for semantic segmentation. The task of semantic segmentation is to assign a semantic label to each pixel in the images. Thus, the images for this task should be pixel-wise labeled according to a pre-defined class structure. In order to save labeling effort, it is better to increase the performance of the model from the perspective of training methods rather than increasing the amount of labeled data.

Moreover, pixel-wise classification for semantic segmentation is much more difficult than image classification or object detection, since it is very sensitive to the changes of marginal distributions of pixel-features and the joint distributions of pixel-features and pixel-labels. These two distributions of training data of a DNN are assumed to be the same as that of the test data. This prerequisite is usually not fulfilled. Because in reality, the images might be collected by different devices or under different environments. For instance, in the field of bioinformatics, biomedical engineers help radiologists doing tissue segmentation or pathology detection using DNN based semantic segmentation. The trained model would always fail if a new scanner or even new configuration or calibration for the same scanner is used for taking medical images[57]. In Remote Sensing(RS), aerial images of city 1 taken by one camera in winter and the images of city 2 taken by another camera in summer may have a big feature difference. One application of semantic segmentation of remotely sensed images in this work is to use the DNN trained on one source dataset(e.g. images of city 1) and then do predictions on a target dataset(e.g. images of city 2). If the statistical distributions of the data in city 1 and city 2 are very different, DNN trained on city 1 can hardly perform well on city 2. Besides, the class label distributions of RS datasets may also differ between cities. For example, one city may have more densely distributed buildings and fewer trees than another. The difference of the area and the shape of each building between different domains can also be problematic for classification during inference. All these differences may lead to an accuracy drop when the network trained on one city is used to predict labels of images of another city. It is arduous and not reasonable to create new dataset for each new scenario or new domain. Thus, it is necessary to utilize DA to decrease the influence of the probability distribution discrepancy of image features between source and target dataset during inference.

DA is a subfield of transfer learning, which aims to narrow the marginal distribution gap of features and the joint distribution gap of features and labels between source and target datasets on the condition that the input space as well as the label space should be the same in both source and target domain, namely the task in both domain should be the same. The main task of this work is to use statistical DA methods to improve the classification accuracy on the target domain on the assumption that no labels are available in the target domain. Specifically, a model trained on a labeled source dataset(source domain) and an unlabeled target dataset(target domain) should perform as well as possible on an unlabeled target domain.

There are many approaches for DA in the field of computer vision. In recent years, many research findings have revealed that DA using deep features extracted by DNN outperforms the shallow features extracted by traditional methods[16]. So this study focus on DNN-based DA, which is notated as deep DA. Deep DA introduces source-target distance constrains or source-target discriminators on deep features to minimize the feature distribution difference between source and target. According to the training methods, deep DA can be classified into two main categories: one-stage training and two-stage training. One-stage training directly trains the whole network with a combination of a classification loss and a discrepancy loss or an adversarial loss. Two-stage training trains both encoder and classifier on source data and then adapts the parameters of the network using target unlabeled data to make the adapted encoder and the classifier work better on target images. No matter for one-stage training or two-stage training, most state-of-the-art approaches use adversarial discriminative methods to solve the domain discrepancy problem of semantic segmentation. In contrast, the rarely researched non-adversarial deep DA methods are the emphasis of this thesis. Maximum Mean Discrepancy(MMD) and deep Coral are two efficient and statistic-based methods that were first introduced by [71] and [65], respectively, to the deep DA of image classification. In this work, these two methods are used on the problem of deep DA for semantic segmentation. Experiments with MMD and deep Coral are conducted based on the RS datasets. In order to compare the performance of the two methods, DA is achieved by aligning the feature activations of an intermediate layer of the encoder with MMD and deep Coral, respectively, as the distance metric between source features and target features. According to [65], deep Coral can be used on any layer. However, it is hard to use MMD on a layer like the output logits layer that has too many pixel samples due to the computational complexity of MMD.

So the performance of the same method on aligning feature distribution of different layers is evaluated only by experiments with deep Coral. An intermediate layer of the encoder and the output logits layer of the decoder are chosen for these experiments. At last, one more experiment is performed to see if the joint adaptation of different layers can further optimize the result. In this experiment, the adaptation is achieved by aligning the intermediate layer of encoder followed by the alignment of the output logits layer.

Another task addressed in this work is automated recognition and segmentation of car components in BMW dataset. In a typical scenario, there are hundreds of different components for a car. It is very expensive to label all the images for different parts of a car. However, the 3d CAD models of all car components are available, they can be used to simulate the scenario of the real application and create synthetic images and the corresponding labels using software like Unity or Blender. Ideally, these synthetic images can be directly used to train the semantic segmentation network. However, the problem is that the synthetic images are very different from real images in appearance. One way to overcome this is image-to-image translation which is a problem of learning a function of mapping an input image to an output image from the sufficient training data[36]. It can change the appearance(style) of the images. However, the training data of input and output images should be paired. Namely, the input and output image in an image pair should contain the same content. However, in most applications, the paired image dataset is not available. Cycle GAN[82] introduces cycle consistency loss to tackle this problem and makes it possible to learn mapping functions only with the unpaired images from two domains. Thus, Cycle GAN is used in this work to mitigate the appearance difference between synthetic images and real images. To check if this technique can also perform well and stable on RS datasets so that it can be integrated into the method with activation distribution matching to further improve the target classification quality, one more experiment is conducted on appearance adaptation by utilizing Cycle GAN[82] both on BMW dataset and RS dataset.

The rest of the paper is organized as follows. Section 2 will first give an overview of the related works. In section 3, the concept of semantic segmentation will be defined, then the components and the overall architecture of DNN used in this work will be discussed. In section 4, a formal description of DA will be defined, and it is followed by the definition of MMD and deep Coral and by the explanation of Cycle GAN. In the end of this section, the methodology of applying MMD, deep Coral and Cycle GAN for DA will be explained. In section 5, the datasets used in this work will be introduced and the experiments made are described. Afterwards, the results of all experiments will be visualized graphically and evaluated. In the last section, the conclusion is drawn based on the results obtained and an outlook for future work will be given.

2. Related Works

Both semantic segmentation and deep DA techniques are two extensive research fields. The related works of semantic segmentation and deep DA will be discussed separately. Then the state-of-the-art works that aim to solve DA problems for semantic segmentation will be discussed.

2.1. Semantic segmentation

Semantic Segmentation describes the pixel-wise classification of images. It is regarded as a much more challenging task compared to image classification. Instead of assigning a single label to each image, semantic segmentation gives every single pixel a semantic label. Before DNN is applied for this task, hand-crafted features which are obtained by using predefined algorithms like SIFT[47] and HOG[17], and traditional machine learning techniques like SVM[22], random forest[30] are used to do the classification. In recent years, tremendous successes in semantic segmentation have been achieved by using DNN.

Fully Convolution Network(FCN)[63] can be regarded as a milestone for semantic segmentation. Contrarily to regular Convolutional Neural Networks(CNN) for image classification, a FCN doesn't contain any fully connected layers but can adapt the basic feature encoding structure of these CNNs like VGG, AlexNet, and ResNet to take an input image of arbitrary size and then generate the output image with the same spatial dimension as input. In general, a FCN first encodes the images into feature maps in a latent feature space and then decode/upsample the feature maps with interpolation or transposed convolution to scale the feature maps back to the original input size and construct a label map.

Based on FCN, many networks have been developed. These networks follow the basic structure of standard CNN to build the encoder, but they use different strategies to upsample the encoded feature maps. ICNet[79] uses bilinear interpolation for upsampling to make the model as light as possible. U-Net[61], a convolutional network originally developed for semantic segmentation of biomedical images, utilizes up-convolution(transposed convolution) to perform upsampling in order to learn more complex and precise assembling of feature information. It has the same number of convolutional layers and transposed convolutional layers for downsampling and upsampling respectively. Based on this symmetric structure, skip connections between downsampling and upsampling layers are applied to combine high-level semantic information with low-level information of higher resolution to further refine the semantic segmentation result. Segnet[3] also has a symmetric encoder-decoder structure, but instead of transferring the entire feature maps to the corresponding decoder layer by skip connections, it stores the indices of maximum activations during max-pooling, and then utilizes these indices to perform upsampling in the decoder and produce sparse feature maps. Each upsampling layer is followed by several convolution layers to generate dense feature maps. DeconvNet[56] uses the same up-pooling strategy, but it introduces deconvolution to generate dense feature maps. According to the results of the works mentioned above, decoding feature maps with convolutions or transposed convolutions perform better than interpolation. In this thesis, transposed convolution is used for upsampling to simply construct the symmetric encoder-decoder structure like U-Net. This structure is also utilized by [75] and [74] to classify pixels of aerial images in RS. However, these two U-Net variants both have two separate encoder branches for multispectral(MS) images and Digital Surface Model(DSM) images because [75] shows that separate encoders for MS and DSM images perform slightly better than a single encoder for the fused input of MS and DSM images. [74] further improved the structure of [75] by using zero-mean convolution for height data to make the model invariant to local terrain height changes, and by replacing padded convolution with unpadded convolution not only because of the positive effect on accuracy but also because of the computational efficiency. Besides, [74] also uses convolution and transposed convolution for downsampling and upsampling respectively because they can preserve small details better than max-pooling and bilinear interpolation. In this thesis, the same RS datasets are used as in [74]. The basic network structure for semantic segmentation in [74] is adopted.

In addition to the basic encoder-decoder structure mentioned above, many techniques can be integrated into any semantic segmentation networks to further improve the performance. For example, Conditional Random Field(CRF) as a post-processing method can refine the final semantic segmentation result. It is integrated into DeepLabV1[9] and DeepLabV2[8]. Instead of regarding CRF as a post-processing step, CRF-RNN[81] formulates CRF as Recurrent Neural Network(RNN) and integrate this into FCN, and train the network in an end-to-end manner. These techniques can be used as an extra aid for semantic segmentation and will not be discussed in this thesis which mainly focuses on DA. Moreover, networks like PSPNet[80] which uses pyramid pooling can reduce the errors caused by different sizes or resolutions of the same object. Because pyramid pooling combines the information from different convolution layers that convey information of different semantic levels and different sizes of the reception field. This network structure is useful when the input images have different resolutions. In this work, images for each DA experiment are converted to the same resolution. Thus, it is not necessary to use pyramid pooling to increase the complexity and capacity of the network.

2.2. Deep Domain Adaptation

Deep DA aims to leverage available labeled data from the source domain to make the semantic segmentation model perform better on the target data, where no labels are available. Deep DA methods can be classified into discrepancy-metric-based methods([71], [46], [66], [65]), and adversarial-discriminative-model-based methods([70], [23]) based on how the features from source and target are differentiated.

Deep Domain Confusion(DDC)[71] is one of the early works of solving DA for image classification that is based on Maximum Mean Discrepancy(MMD). DDC introduced MMD loss to the outputs of the fully connected layer of AlexNet[38] between source and target activations to mitigate the feature distribution discrepancy. Deep Adaptation Network(DAN)[46] is published one year after DDC, it modified DDC based on two considerations. First, the domain shift may occur in multiple layers. Second, kernel functions are critical for MMD estimation. So DAN uses multi-kernel MMD(MK-MMD) to adapt all fully connected layers. CORrelation ALignment(CORAL)[66] is proposed to align second-order statistics of source and target features to minimize domain shift. It computes a linear transformation based on the covariance of source and target features and applies this transformation on the source features to transform these features from the source domain to the target domain. This transformation can be split into two steps. First, whitening the source feature by removing the covariance of the source features. Then, re-coloring the whitened source feature by applying the covariance of the target features to the source features. Deep Coral[65] replaces the linear transformation in the original work CORAL by an implicit nonlinear transformation in DNN that is learned by backpropagation and gradient descent.

Adversarial Discriminative Domain Adaptation(ADDA)[70] is the first generalized framework for adversarial discriminative DA. It uses 2-stage training. At first, it trains the whole network including encoder and classifier on source dataset. Then, it copies weights from the source encoder to the target encoder and shares the same classifier in both source and target domain. Only the weights in the target encoder are updated in an adversarial manner, namely, it trains the target encoder and discriminator jointly. Domain-Adversarial Neural Network(DANN)[23] proposed a similar architecture, but it trains encoder, classifier and discriminator simultaneously by inserting gradient reversal layer between encoder and discriminator, so that the network can be trained by minimizing both classification loss and discrimination loss. According to the authors' announcement, the new proposed Geometry-Aware Domain Adaptation Network(GA-DAN)[77] can model domain shifts that are caused by large perspective changes of objects in the images and can realistically convert images across domains that have very different geometric appearance characteristics. This might be very beneficial for augmenting datasets like COCO, Cityskypes, in which one specific object might be observed in different directions. However, aerial images are taken remotely with a specific height and orientation, namely the camera has certain pose, and the objects on the ground are static, so the domain shift caused by perspective changes can hardly influence RS images.

The technologies for solving DA problems are very diverse. Based on the works mentioned above and many other works, DA techniques have evolved a lot and been applied to different scenarios on DNN applications. Many of these methods are not originally designed for semantic segmentation. However, the basic ideas of these methods can be inspiring for semantic segmentation. In this work, MMD and deep Coral will be explored based on the task of semantic segmentation.

2.3. Deep Domain Adaptation for Semantic Segmentation(DDASS)

In the recent two years, deep DA for semantic segmentation has been researched a lot make use of the huge street-view datasets like GTA5, SYNTHIS, and CITYSCAPES. Most papers in this research direction are published in these 2 years and the networks developed are mostly based on these 3 datasets. In contrast, the topic DDASS in remote sensing community has not been researched too much.

There are many different techniques for DDASS based on matching the input or output distribution of any possible layer in a DNN from source and target domain. The start point of solving this problem is to think "which" distribution should be aligned and "where" the chosen distribution should be aligned. "Which" refers to marginal distribution or joint distribution. The marginal distribution of a high-dimensional variable doesn't relate to any pre-defined classes. So the alignment of marginal distribution is notated as global matching. In contrast, joint distribution relates to two variables, the feature variable, and the label variable. The label variable is related to pre-defined label categories. So category-wise matching is then used to refer to joint distribution alignment. "Where" refers to the location of the input or output data of a layer. These locations of layers are sorted into 3 levels, appearance(input data of the first layer), feature(activations of any intermediate layer of a network) and output(logits output before the softmax layer, probability output after softmax, or predictions). The overview of this taxonomy is summarized in figure 1. The arrows on the first level of the taxonomy tree tell "where" the distribution should be matched. The text notated on the arrow shows "which" distribution of the data at the chosen location can be matched. Each full path from left to right refers to a complete method. For example, the unique path to image translation means this method aligns the global marginal distribution of input images(appearance adaptation). Based on this taxonomy, the methods for DDASS will be listed firstly according to "which"(global or category-wise) and then

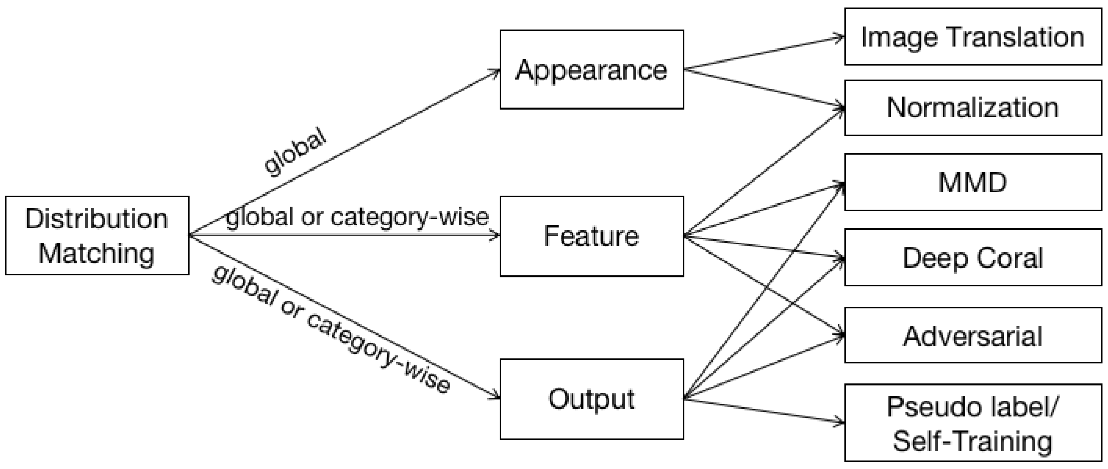


Figure 1: A taxonomy for deep domain adaptation methods of semantic segmentation

according to "where"(appearance, feature or output).

All methods do the **global** alignment. On top of this, some methods also tried **category-wise** alignment to decrease the joint distribution discrepancy of features and outputs. For example, [7] achieved this by using multiple domain classifiers, [32] re-weights the gradients by transferring the image-level label distribution from source to target domain. [49] also uses a re-weight strategy, but it re-weights the adversarial loss instead of gradients. [83] uses different proportion parameters to filter pseudo labels similar as [35], which utilizes a category-balanced pseudo-label-selection strategy.

Appearance alignment is one class of methods that makes the style of the images from source and target domains look similar. Normalization can be regarded as a standard pre-processing step, it is used nearly in all image processing DNNs. To further alleviate the over-fitting problem between source and target domain, [74] uses online augmentation by applying random scale and shift during training. Image translation is another prevalent method for translating image styles from one domain to another. Most works use CycleGAN([82]) architecture or the variants of it to do the image translation to decrease the appearance discrepancy between source and target images([54],[31],[4], [41],[11],[25],[10]). There are also other techniques to translate image appearance like [6] which uses multiple encoders to disentangle domain-invariant and domain-variant features and then combine these two of different domains to obtain new stylized images. Different from CycleGAN, [12] introduces a cycle-free architecture to change the image appearance. Since this class of method can be regarded as pre-processing of the images, it can be applied additionally to any other DA method. In this work, online augmentation will be used for training the source models to make the trained source model more robust both in the source and target domain. Image translation with CycleGAN will also be tested both on RS data and BMW data to prove if it can generally help to decrease the domain discrepancy for all applications.

Feature in figure 1 refers to the activation of any intermediate layer in a semantic segmentation network. There are loads of works aligning feature distributions and mostly in a adversarial manner([7],[32],[31],[33],[11],[48]). DAN[60] instead uses normalization to align feature distributions. And different from DA for image classification, there is no work applying MMD or deep Coral to solve the DDASS problem, no matter in feature space or output space. So in this work, the potentiality of MMD and deep Coral will be explored.

Output in the taxonomy can be the predicted label map, the probability output after the softmax layer or the logits output before the softmax layer as mentioned above. The two main methods operating on output layer are the adversarial method and pseudo label/self-training. Adversarial methods align the output distributions by using discriminators to distinguish the output from the source and the target domain. Similar to a standard GAN, it updates the weights of semantic segmentation model and discriminators jointly by maximizing the discriminative loss and minimizing the generative loss. [68] aligns the output logits, [6] instead aligns the source prediction and the target prediction while [73] aligns the fused features of the predicted depth map and the entropy of logits. Instead of matching the output distribution of source and target model, [53] matches the source ground truth labels and the predicted labels of both the source and the target image. Self-training, on the other hand, is the process of minimizing the cross-entropy between the predictions and the generated/selected pseudo labels. Pseudo labels are the predicted labels. The most reliable pseudo labels are "picked" and then used to supervise the training of semantic segmentation network. There are different ways to choose reliable labels. [83] and [19] choose the labels in the whole prediction map that have a higher probability than a pre-defined threshold probability. [35] uses category-balanced pseudo-label selection. The number of the selected pixels for each class depends on the occurrence frequency of this class in the source images. Besides, there are also other forms of pseudo labels. For example, [7] "softens" the label by using grid-wise label distributions(proportion of each class in each grid). [78] utilizes superpixels to generate pseudo labels and [42] replaces superpixel in [78] with overlapped squares. Since it is hard to find any patterns in the label maps that are invariant in both source and target domain when solving DA problem of RS domains in this work, methods like [7], [42] and [42] would all fail. However, one experiment will be conducted in this work by aligning the output logits with deep Coral instead of adversarial methods mentioned above.

3. Semantic Segmentation

Semantic Segmentation is a task of assigning a label to every single pixel in an image, in other words, the trained DNN of semantic segmentation can "understand" images on pixel level. Figure 2 gives an overview of semantic segmentation. For this task, semantic labels on the right side are supposed to be generated by passing the input image on the left side of the figure to the semantic segmentation DNN. Each label has a semantic meaning. For example, label 3 means plants or grass and is marked in green.

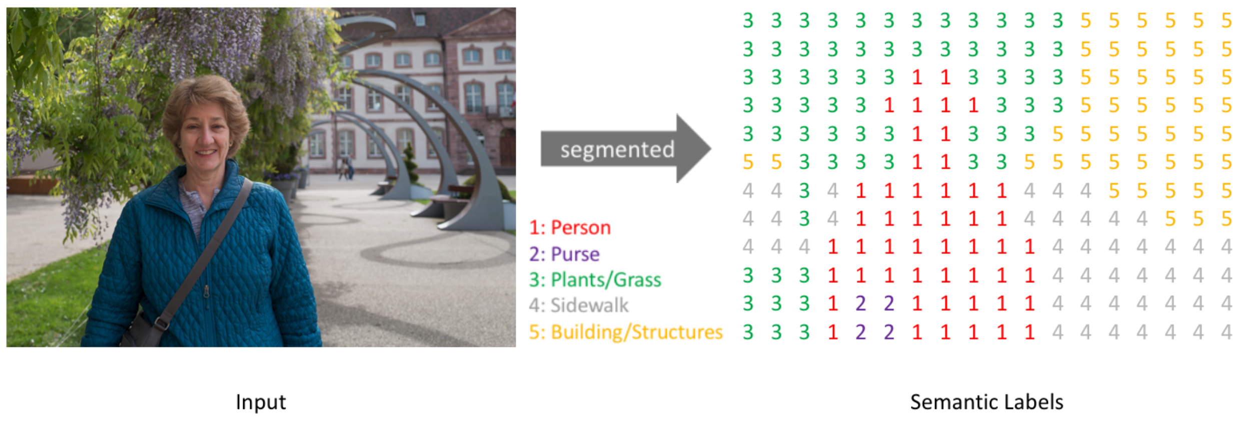
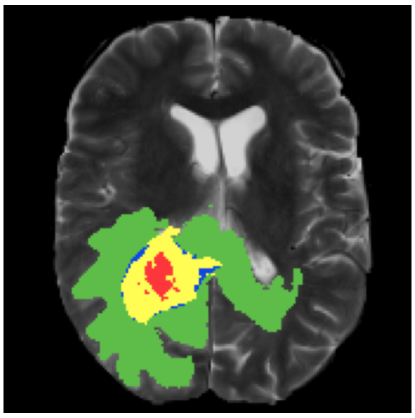
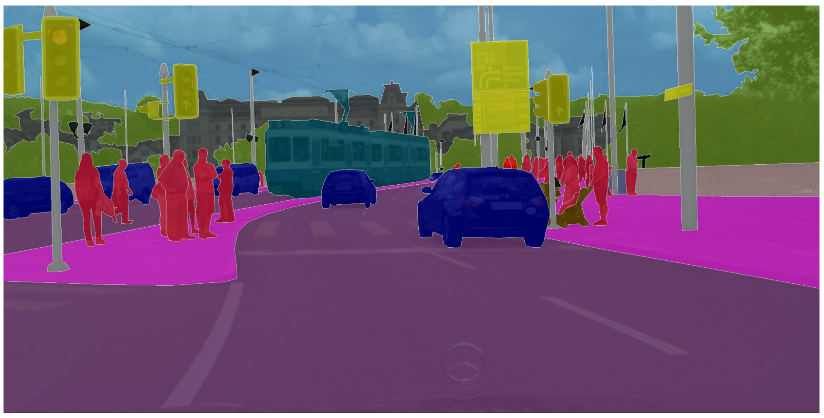


Figure 2: An example of semantic segmentation[37]

Compared to plain object detection, semantic segmentation can extract richer semantic information from images and locate the objects more precisely. Classifiers of semantic segmentation are more powerful and straight forward than that of object detection in many applications, such as brain tumor segmentation(fig:3a) for MRI images of the brain using BRATS[52] dataset, it can not only detect the tumor but also locate where exactly they are and tell what shape they have. For autonomous driving, it's also very beneficial to classify every pixel in the street scene so that the vehicle can make the best decision on where to go with this rich information of semantic segmentation(fig:3b). In RS, manually labeling the aerial images to generate land use or land cover map is very time-consuming(fig:3c). If one semantic segmentation model is trained on a small amount of labeled dataset and then used to segment and classify the pixels in aerial images of other datasets, it can save a lot of time.

There are traditional segmentation algorithms like k-means, mean-shift, watershed, and spectral clustering that can be used to segment the images into patches. They can be combined with traditional classification algorithms like SVM, random forest, and adaboost to achieve the semantic segmentation. But nowadays, DNN based algorithms have become more popular and are the state of the arts in this task since they can achieve higher accuracy and can integrate the segmenter and the classifier into one single model and get the final


(a) Brain tumor segmentation


(b) Street scene segmentation

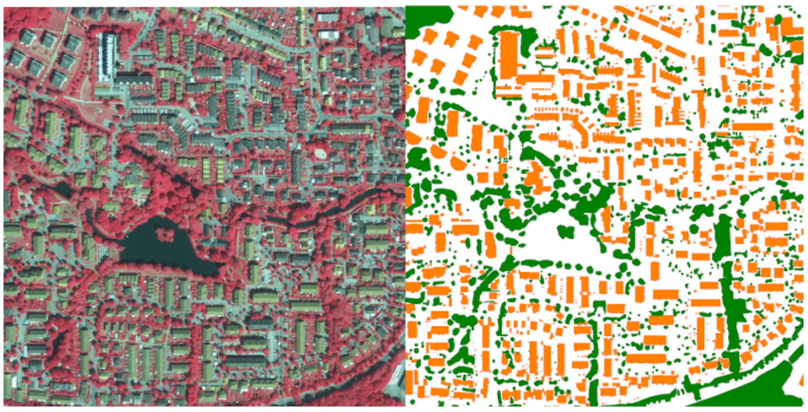

(c) Aerial image segmentation

Figure 3: Use cases of semantic segmentation

semantic segmentation result on the fly. So in this work, only the deep-learning-based semantic segmentation algorithms will be discussed.

In the following subsections of this chapter, the task of semantic segmentation will be defined formally. Afterwards, the network architecture used in this work will be explained. In the last part of this chapter, the evaluation metrics being used in this thesis will be discussed.

3.1. Notations and Definitions

To better describe the task of semantic segmentation, some notations are introduced. A dataset is defined as $D = \{x_i, y_i\}_{i=1}^{N}$, where each x_i is an independent sample or image drawn from variable space $\mathcal{X}$, y_i is the corresponding label mask for x_i and belongs to the label space $\mathcal{Y}$. X and Y refer to the samples drawn from variable space $\mathcal{X}$ and $\mathcal{Y}$ respectively. The variable space $\mathcal{X}$ is a real space $\mathbb{R}^{d \cdot h \cdot w}$, d, h, w are the number of channels, height and width of the input image respectively. The output label space is a discrete real space $\mathbb{R}^{h \cdot w}$ of C classes. Each pixel of the a label map has one of the label values from set {1,...,C}.

Domain is notated as $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}, P(X), P(X, Y)\}$, which contains 4 components, the input space $\mathcal{X}$, the marginal distribution of the input data $P(X)$, the output space $\mathcal{Y}$ and the joint distribution of X and Y, $P(X, Y)$. The DNN architecture of the task is defined as M, of which the parameters are θ. The task of semantic segmentation is defined as $\mathcal{T} = \{\mathcal{X}, \mathcal{Y}, f(\cdot)\}$, which has 3 components, input space $\mathcal{X}$, label space $\mathcal{Y}$ and objective prediction function $f(\cdot)$, which is the combination of M and θ and also represents the conditional distribution $P(Y|X)$. The predicted label mask for input image x_i is notated as $\hat{y}_i$. With all notations defined above, the task of semantic segmentation can be summarized and defined as equation

$$\min_{\theta} \hat{R}(\theta|D, M) = \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} l(f(x_i), y_i) = \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} l(M(x_i, \theta), y_i) \tag{1}$$

To obtain the optimal configuration of the parameters θ, the empirical risk $\hat{R}$ expressed in equation 1 should be minimized with the condition of training data D. N is the number of samples. l is a loss function, which will be further discussed in section 3.2.1.

3.2. Components and Architecture of the FCN

Before introducing the holistic view for the network architecture, a brief explanation of the common basic modules of FCNs for semantic segmentation is given in the first part. DNN is composed of many different layers, it is not possible to list and explain them all in this work, only the modules used in this work are introduced. These modules are then connected and depicted graphically in subsection 3.2.2. The network used in this work is a FCN network and it also has a symmetric encoder-decoder structure like U-Net. Different from most other works, which uses standard base networks like VGG, ResNet, etc. as feature extractor(encoder), a simpler structure for both feature extractor(encoder) and classifier(decoder) is built with convolutional layer, dropout, Rectified Linear Unit(ReLU), transposed convolutional layer, and cross-entropy for loss computation. Besides, skip-connections are not used across the whole work.

3.2.1. Basic modules for the Network Architecture

Convolutional Layer

Compared to the fully connected layer, the convolutional layer is more efficient when applied to structural data like images and point clouds. In a fully connected layer, each neuron is connected to every neuron in the previous layer, and each connection has its own weight. In contrast, each neuron in a convolutional layer is only connected to a few nearby neurons and each set of weights(weights in each filter) in this layer is shared across the whole image because convolutional layer assumes that the same feature is equally likely to occur anywhere in the image.

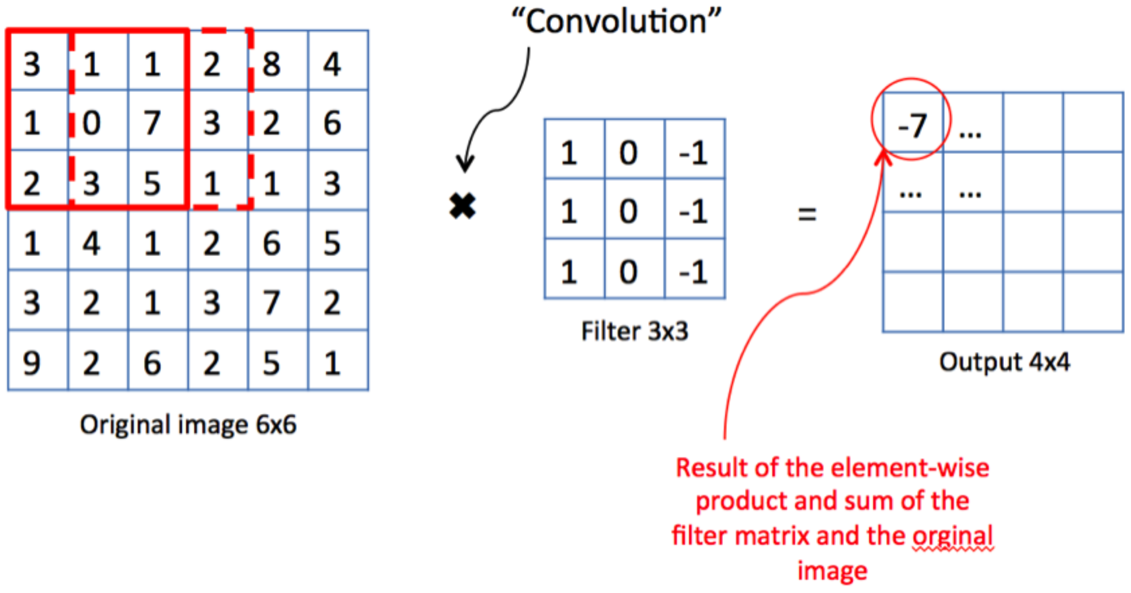


Figure 4: Convolutional layer[5]

Assume that there is an image with 1 channel and the size of 6x6 as input. Figure 4 gives an overview of how a convolutional layer with a filter size of 3x3 works. The parameters of the filter are also called weights, they weight the importance of the pixel values by multiplying the weights and the image values covered by the filter(red rectangle, this area is also called as receptive field) element-wise. The summation of these multiplications will be the value for the first element of output(red circle). Then the filter is shifted one step(pixel) to the right, and the multiplication and the summation are done again on this position. How many steps to move after each operation is defined as stride, in this case, stride=1. A convolutional layer is composed of many filters(kernels), which operates on input that has many channels. All feature channels of the input are convolved with each filter and one single channel of the new feature maps is generated for each filter by summing up all convolution results of this filter operated on all channels of the previous feature maps. For example, if an input of shape $3 \times 6 \times 6(depth \times height \times width)$ is given, and 10 filters are used for the convolutional layer, the output shape will be $10 \times 4 \times 4$. The output size is shrunk from 6 to 4 in figure 4. Mathematically, convolution operation can be expressed as equation

$$O_{i,m,n} = b_{i,m,n} + \sum_{c=0}^{C_{in}-1} \cdot \sum_{j,k} W_{i,j,k} \cdot I_{c,m-j,n-k} \tag{2}$$

$O_{i,m,n}$ is the value at the location of the m-th row and the n-th column of the i-th output feature map of the convolutional layer. C_{in} is the number of channels in the input feature maps. Index j and k indicate the location of weight in the filter. Namely, $W_{i,j,k}$ is the weight at the j-th row and the k-th column of the i-th filter. I is the input feature map, the subscripts of I indicate the channel, row, and column of the input respectively.

To overcome the edge effect caused by the size shrinking mentioned above, padding techniques like zero padding, reflection padding, replication padding, and partial convolution-based padding[44] can be used. They pad the image with a certain width of borders with different strategies before the image is passed to a convolutional layer. Zero padding fills the borders with zeros. Reflection and replication use the pixel values of edges in the original image to do the padding. Different from these 3 methods, partial convolution-based padding regards padded borders as holes and recovers the information of the holes by convolution operations. It can effectively reduce the bad predictions on the edges of the images. To decrease the training time, no padding is used in this work.

Another technique for convolution is dilated convolution[76]. It skips points during convolution and can enlarge the receptive field without increasing the numbers of weights. The normal convolution mentioned above can be regarded as a dilated convolution with a dilation factor of 1. In this work, the RS images are small enough so that the receptive field of the feature point can cover the whole image only with a few convolutional layers, so the dilation factor is set to 1 for all convolutional layers.

Transposed Convolutional layer

Once input images are contracted into feature maps with smaller size after encoding by convolutional layers, up-sampling is needed to translate the low-resolution feature maps into high-resolution output images. Semantic segmentation, as well as the generator in GAN[26], is the application of up-sampling. Bilinear interpolation and transposed convolution[20](also known as deconvolution or fractionally-strided convolution) can be used to perform up-sampling. Bilinear interpolation samples the nearby pixel values in the image and uses them to compute the value for the new interpolated pixel. It is easy to implement and has no parameters to be learned, so it can decrease the number of parameters and decrease the capacity of the model. Different from bilinear interpolation, transposed convolution bears learnable parameters, and can perform better in many tasks. So if the network is not supposed to work on devices like smartphone and other low costs embedded system that only have very limited computational power, it is preferred to use transposed convolution for up-sampling, because it can increase the accuracy dramatically compared to bilinear interpolation.

Figure 5 shows 2 examples of transposed convolution, which are used in the network architecture of this work. Assuming that the input shape is $1 \times 4 \times 4$, the first example(notated as tr. conv1) shows how transposed convolution with stride 2 and filter size 2 works. Similar

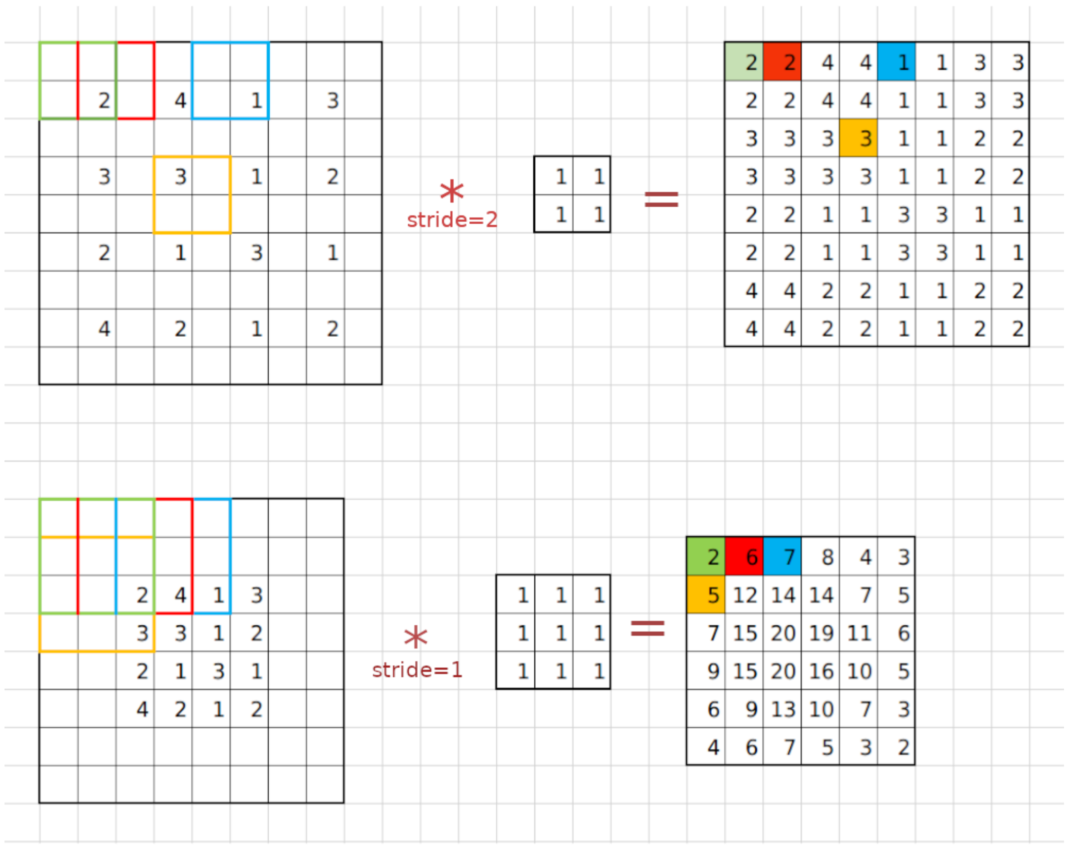


Figure 5: Transposed Convolution

to convolution, the filter of transposed convolution should also shift from left to right and from top to bottom until the final feature map is generated. For example, filter with parameters of 4 ones multiplies with the values in the green rectangle element-wise and then these values are summed up. As a result, the value $2(0 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 + 2 \cdot 1)$ is obtained as the first element, which is marked as green, for the feature map on the right. More examples of the same operation are marked with different colors. The size of the feature map is doubled after transposed convolution with the configuration of the first example and the values of the feature map are just copying values from the input, but if the weights are different, the values of the output feature map would be more diverse. In the second example(notated as tr. conv2), stride equals 1 and filter size is 3. Similar to convolution, no padding is used for transposed convolution to make the network symmetric so that the size of the input images can be recovered at the output.

Activation Functions

Convolutional layers only contain multiplication and summation. If many different convolutional layers are connected one by one directly, they would perform just like one layer only with different weights, because nested matrix multiplications can be replaced by a single multiplication. It is proven in equation

$$\begin{aligned}
O_{i_{l+1},m_{l+1},n_{l+1}} &= b_{i_{l+1},m_{l+1},n_{l+1}} + \sum_{c_{l+1}=0}^{C_{in_{l+1}}-1} \sum_{j_{l+1},k_{l+1}} W_{i_{l+1},j_{l+1},k_{l+1}} \cdot I_{c_{l+1},m_{l+1}-j_{l+1},n_{l+1}-k_{l+1}} \\
&= b_{i_{l+1},m_{l+1},n_{l+1}} + \sum_{c_{l+1}=0}^{C_{in_{l+1}}-1} \sum_{j_{l+1},k_{l+1}} W_{i_{l+1},j_{l+1},k_{l+1}} \cdot (b_{i_l,m_l,n_l} \\
&+ \sum_{c_l=0}^{C_{in_l}-1} \sum_{j_l,k_l} W_{i_l,j_l,k_l} \cdot I_{c_l,m_l-j_l,n_l-k_l}) \\
&= \boxed{b_{i_{l+1},m_{l+1},n_{l+1}} + \sum_{c_{l+1}=0}^{C_{in_{l+1}}-1} \sum_{j_{l+1},k_{l+1}} W_{i_{l+1},j_{l+1},k_{l+1}} \cdot b_{i_l,m_l,n_l}} \\
&+ \boxed{\sum_{c_{l+1}=0}^{C_{in_{l+1}}-1} \sum_{j_{l+1},k_{l+1}} \sum_{c_l=0}^{C_{in_l}-1} \sum_{j_l,k_l} W_{i_{l+1},j_{l+1},k_{l+1}} \cdot W_{i_l,j_l,k_l}} \cdot I_{c_l,m_l-j_l,n_l-k_l}
\end{aligned} \tag{3}$$

which utilizes equation 2 and takes two directly connected layers l and $l+1$ into consideration. As the equation derived above, the expression in the red rectangle can be regarded as a single bias and that in the blue rectangle as a single weight.

In order to increase the complexity and capacity of the model, non-linearity is introduced between convolutional layers. Some of these non-linear functions used in DNN are listed in Figure 6. They are also called activation function because they decide whether a neuron should be activated("fired") or not. Sigmoid is very computational expensive because of the exponential term. Besides, Sigmoid, as well as tanh, has the vanishing gradient problem because the output values will get saturated if the input is too large or too small. The gradients will get smaller and smaller along the backpropagation path from the last layer to the first layer. ReLU solves the vanishing gradient problem by using a linear function for all positive input values. It outputs zero values when the input is negative and makes the activations sparse. The linear function and the sparsity of activations make ReLU much easier to compute than Sigmoid and tanh. The drawback of ReLU is that a neuron can hardly recover once it gets negative value and outputs 0, this problem is called Dying ReLU. The variants of ReLU, Leaky ReLU, solves this problem by giving a small slope to the negative region as described in figure 6. ELU replaces the linear part of Leaky ReLU in the negative region with a curve so that the small slope works only on negative values that are near to zero. This can make a part of the activations saturated and speed up training[13]. Maxout is a learnable activation function that is a generalization of ReLU and leaky ReLU and doesn't have the dying ReLU problem. But it increases the number of parameters to be learned.

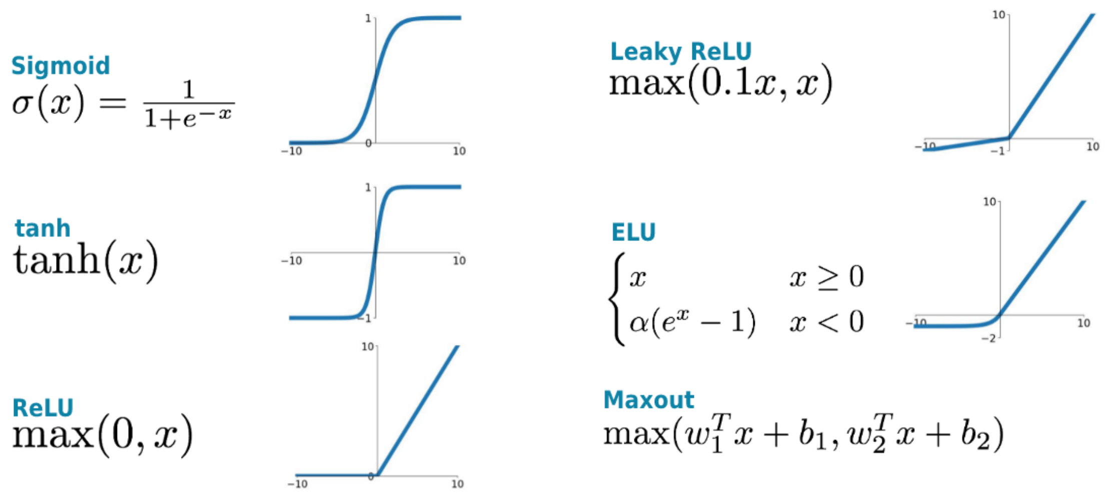


Figure 6: Activation functions for DNN[14]

In this work, easy and fast convergence of the network is the first criterion for selecting modules. ReLU and Leaky ReLU are easy to compute both in the forward and the backward path of training DNN. They are also saturation-free and parameter-free. During training, no strong signs of vanishing gradient problem are observed by using ReLU. So instead of Leaky ReLU, the standard ReLU is chosen as the activation function in this work.

Dropout

DNNs are supposed to learn general concepts from specific examples. However, it is common that a DNN also learns specific concepts from the samples. This makes the DNN perform very well on the data it was trained on but much worse on the test dataset, which the model has never seen during training. This happens because the model is overly complex and has too many parameters. This problem is called overfitting. Neural networks are susceptible to overfitting because they use many layers and parameters to represent a complex function and it is hard to find the best-fitted networks for a specific task. Techniques like data augmentation, regularization, and dropout[64] are used to prevent overfitting while training neural networks. Image normalization and online augmentation are the two methods of data augmentation that will be used in this work. They are regarded as pre-processing steps rather than the module of DNN, so the detailed configuration of these two methods will be discussed in the experiments part in section 5.

In addition to data augmentation, regularization like L1 (Laplacian) or L2 (Gaussian) penalties on the weights of models can be used to decrease the capacity of the model and reduce overfitting. They minimize the absolute or the squared values of weights by adding the L1 or L2 regularization term to the loss function mentioned in equation 1. Dropout, on the other hand, modifies the network itself and will be used in the network of semantic segmentation in this work. Figure 7 shows how dropout works. It randomly drops neurons

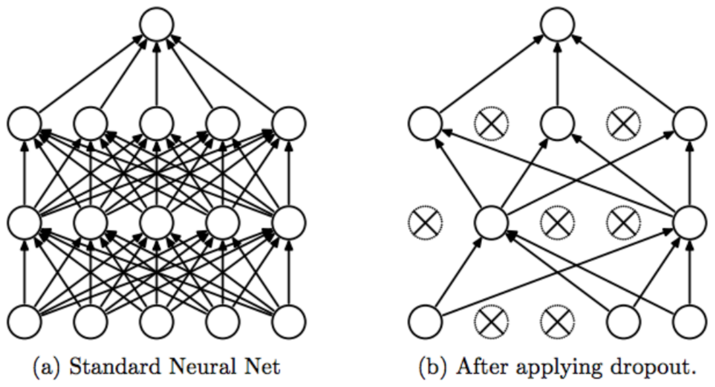


Figure 7: Dropout[64]

during training with a probability of p_{drop}. During test, it uses all neurons by multiplying a factor of $(1 - p_{drop})$ to the activations. Dropout aims to increase generalization of the neural networks by mitigating the correlation among neuron activations[67]. This assumption may not hold when dropping out neurons for output of 2d-convolutional layers since these neurons are strongly spatial correlated. To overcome this, [67] proposed spatial dropout, which randomly drops some channels of the features maps during training. In other words, if one channel is dropped out, all values of the feature map in this channel would be set to zero. Since the network used for semantic segmentation is FCN which only contains convolutional and transposed convolutional layers, the spatial dropout module will be used in each convolution block.

Softmax

Softmax is also an activation function that is often used in the last layer to produce an output vector per pixel that represents a probability distribution. The sum of all values in the vector is 1. Softmax function is defined in equation

$$\delta(z_k) = \frac{e^{z_k}}{\sum_{i=1}^{C} e^{z_i}} \tag{4}$$

C is the number of all possible classes, k is the current class, for which the probability is to be computed. z is the input vector.

Cross Entropy

Cross entropy is a concept in information theory. In information theory, entropy is originally defined by Shannon[62] to calculate the smallest average size of encoding for the transmission of messages. It is mathematically defined in as

$$H(P) = -\mathbb{E}_P[\log P] = -\sum_i P_i \log P_i = \sum_i P_i \log \frac{1}{P_i} \tag{5}$$

P is the distribution of all possible events, and P_i is the probability of the occurrence of the i-th event when considering a discrete case. For example, P_i can refer to the occurrence probability of "sunny" day when encoding different whether conditions. In the last term of the equation, $\log \frac{1}{P_i}$ tells the number of bits needed for encoding the message of the i-th event. In reality, the real distribution is always unknown, so cross-entropy is introduced to measure the average number of bits needed to encode data coming from a source with an unknown real distribution P by a model Q([55],p.57), which is shown in the following equation.

$$H(P, Q) = -\mathbb{E}_P[\log Q] = -\sum_i P_i \log Q_i \tag{6}$$

Since cross-entropy is always not the optimized encoding, it is bigger, or in the best case equal to the real entropy. The distance between the cross-entropy of the distribution Q of the sampled data and the entropy of the underlying real distribution P of these samples is called Kullback-Leibler Divergence(KL-Divergence) and is expressed as

$$D_{KL}(P\|Q) = H(P, Q) - H(P) = -\sum_i P_i \log Q_i + \sum_i P_i \log P_i = \sum_i P_i \log \frac{P_i}{Q_i} \tag{7}$$

It can be reformed to

$$H(P, Q) = H(P) + D_{KL}(P\|Q) \tag{8}$$

In equation 8, it is easy to find that the KL-divergence can be minimized by minimizing cross-entropy, because for a certain dataset, the entropy of it's real distribution is certain and can be regarded as a constant. In a machine learning problem, small KL-Divergence means the distribution learned from the samples is close to the real distribution of the variable in the space, from which the samples are drawn.

To train DNN, the loss function is needed as guidance to tell the current model, how good it is performing now so that it can backpropagate gradients correctly. Cross entropy is one of the most popular loss functions. In semantic segmentation, the output shape of the softmax layer is $C \times H \times W$. C is the number of classes, H and W are the height and width of the input image respectively. The output of the networks is notated as $\hat{\mathbf{p}}$, and $\hat{p}_{kij}$ means the probability that the pixel in the ith-row and jth-column belongs to class k. Based on this notation, cross-entropy can be expressed as

$$L_{CE} = -\sum_{kij} p_{kij} \log \hat{p}_{kij} \tag{9}$$

Inspired by cross-entropy, Negative Log-Likelihood(NLL) loss

$$L_{NLL} = -\sum_{kij} \delta(\hat{y}_{kij}) \cdot \log \hat{y}_{kij} \tag{10}$$

is often used after softmax layer. The function δ in NLL loss equation is

$$\delta(\hat{y}_{kij}) = \begin{cases} 1 & \text{if } \hat{y}_{kij} = y_{kij} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

which equals one only in the case that predicted label $\hat{y}_{kij}$ equals the ground truth label y_{kij}, otherwise it equals zero. Softmax and NLL loss can be combined into one module described as

$$\begin{aligned} L_{task} &= -\sum_{kij} \delta(\hat{y}_{kij}) \cdot \log \frac{e^{\hat{o}_{kij}}}{\sum_k e^{\hat{o}_{kij}}} \\ &= -\sum_{kij} \delta(\hat{y}_{kij}) \cdot (\hat{o}_{kij} - \log \sum_k e^{\hat{o}_{kij}}) \end{aligned} \tag{12}$$

to simplify the computation. The letter **o** refers to the output logits layer before the softmax layer.

3.2.2. Architecture of the FCN

The network architecture follows the basic structure of [74]. Instead of using the original UNet-like structure, the skip connections, without which the accuracy is only slightly dropped, is omitted. Since the focus of this work is DA, the simpler structure of semantic segmentation network for experiments of DA is preferred. Another difference from [74] is that the input image size is decreased from 640x640 to 320x320 based on the observation that the performance of the network is not highly affected by the image size if a robust network structure is used. Because the input size is halved, the size of the feature maps in the latent space, in other words, the output size of the feature extractor is also halved, which may lead to the loss of spatial information. Without skip connections, the lost spatial information can barely be recovered, so 2 convolutional layers and 2 transposed convolutional layers are removed to make the latent feature maps spatially larger. Figure 8 shows the whole structure of the network used for aerial image segmentation tasks. Since the shape of this network

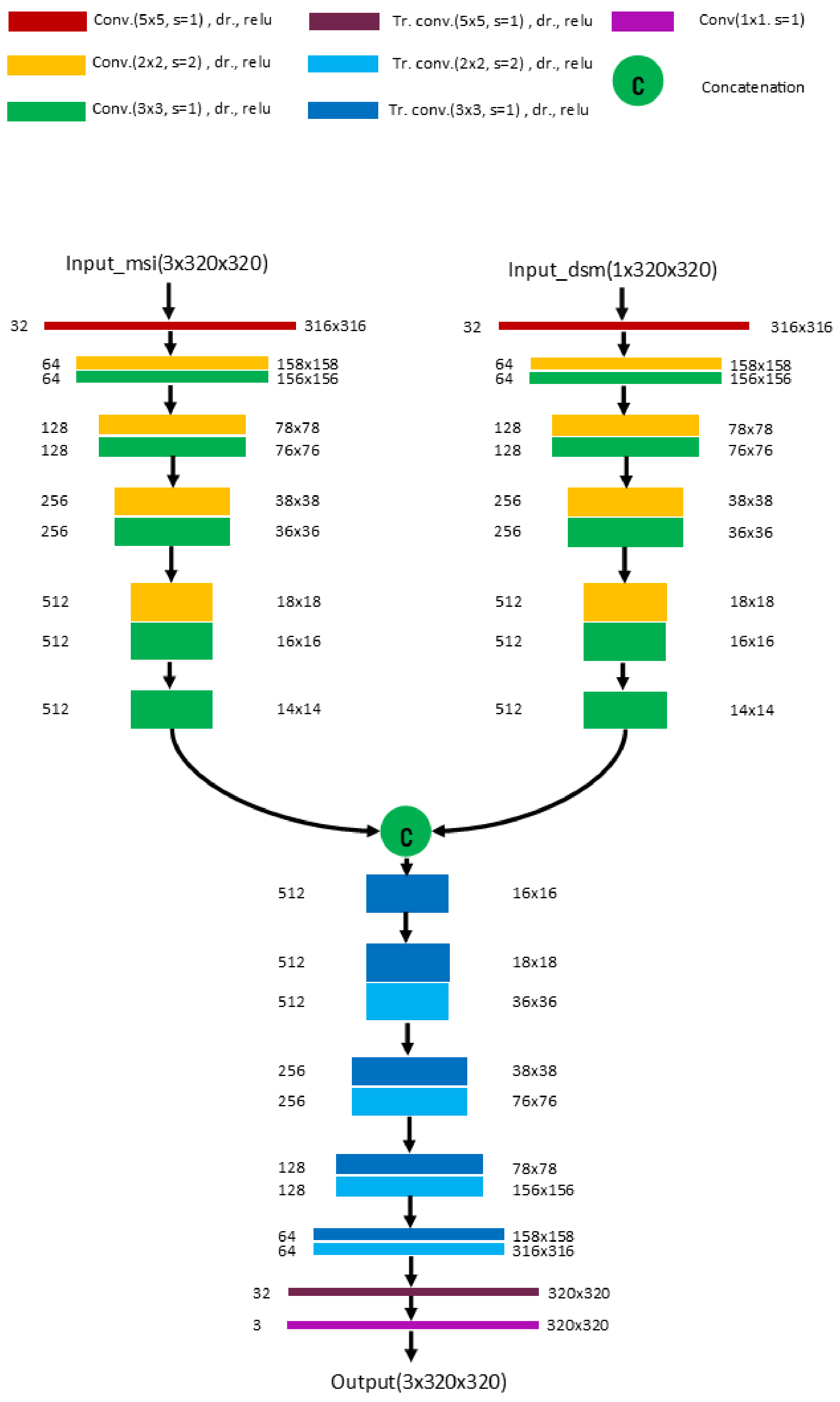


Figure 8: Network architecture

looks like the letter "Y", it is called Ynet. Similar to [74], separate paths for Multi-Spectral images(MSI) and the DSM images are used because [75] shows that a single path for MSI and DSM input performs worse than two paths.

On the top of figure 8 is the legend of different modules in the network. For example, the first red block refers to a sequence of modules, convolution(filter size is 5x5, stride is 1), dropout and ReLU successively, which are regarded as one layer. In the whole network, the probability 0.1 is used for dropout. The lower part of figure 8 shows how these layers are connected. In the encoders, down-sampling is achieved by convolution with a filter size of 2×2 and a stride of 2. The same filter size and stride are used for transposed convolution in the decoder to make the encoder and decoder symmetric. After each down-sampling(up-sampling) layer, a 3×3 convolution(transposed convolution) with a stride size of 1 is used to encode(decode) the features from the previous layer. Regarding the small size of the input image, only 4 down-sampling(up-sampling) layers are used. The numbers on the left of each module block indicate the output channels of the convolutional or transposed convolutional layers. On the right, the output feature map size for each layer is given. The output of the network has the shape of $3 \times 320 \times 320$ and will be used to compute the loss L_{task}(equation 12) during training. During inference, softmax is used instead to generate probabilities and the final label is determined by the maximum probability across class channels for each pixel.

3.3. Performance Evaluation Metrics

In order to evaluate how the model performs, appropriate metrics, which operate on predicted labels and ground truth labels should be used. Pixel Accuracy, Overall Accuracy, and mean Intersection Over Union(mIOU) are the metrics used in the work for semantic segmentation. In the following subsections, they will be explained one by one. Before introducing these metrics, several notations are defined as follows:

- TP: True Positive indicates the number of pixels that are classified as class k are truly from class k.
- FP: False Positive indicates the number of pixels that are classified as class k but labeled as other classes in the ground truth.
- TN: True Negative indicates the number of pixels that are from other classes are not classified as class k.
- FN: False Negative indicates the number of pixels that are truly from class k but classified as other classes.

3.3.1. Pixel Accuracy and Overall Accuracy

Pixel Accuracy(PA) is the percentage of pixels that are correctly classified. For class k, it is defined as

$$PA_k = \frac{TP_k + TN_k}{TP_k + FP_k + TN_k + FN_k} \tag{13}$$

Overall Accuracy(OA) is the PA-like metric for evaluating the accuracy of all classes. OA is calculated with the number of correctly classified pixels divided by the total number of pixels in the input image. Mathematically, it can be expressed as

$$OA = \frac{\sum_{i=1}^{N} \delta(\hat{y}_i = y_i)}{\|Y\|} \tag{14}$$

The function $\delta(\hat{y}_i)$ is similar to equation 11. $\hat{y}_i$ and y_i are the predicted label of the ith-pixel in the predicted label map $\hat{Y}$ and the label of the same location in the ground truth label map Y respectively. $\|Y\|$ represents the total number N of pixels in Y.

There is a drawback of this metric. If class labels are strongly unbalanced, this metric would be unreliable to represent how good the model is performing in some applications. For example, if only very small objects are within the image. These objects only compose 10% of the pixels in the image, and except these pixels, other pixels are background. The accuracy can still reach 90% if the whole image is classified as background. In this situation, mIOU is always preferred. However, for aerial images, different classes are mostly well distributed, so OA will still be used as the main evaluation metric for the semantic segmentation model in this work.

3.3.2. Mean Intersection Over Union

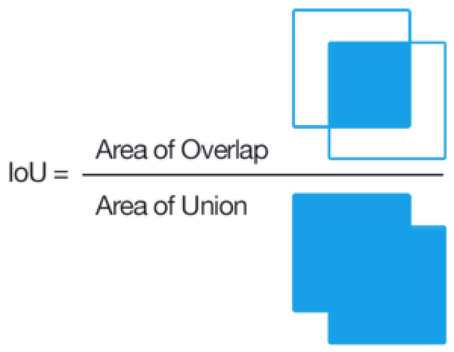


Figure 9: Intersection Over Union(source:Wikipedia)

Intersection Over Union(IOU) is an effective metric commonly used in semantic segmentation. It is also known as Jaccard Index. Figure 9 describes this metric graphically.

Mathematically, it can be expressed as the following function.

$$IOU_k = \frac{TP_k}{TP_k + FP_k + FN_k} \tag{15}$$

Mean Intersection Over Union(mIOU) is simply the mean value of IOUs over all classes, which is calculated with

$$mIOU = \frac{1}{C}\sum_{i=k}^{C} IOU_k \tag{16}$$

C is the total number of classes. This metric gives the evaluation of overall performance of the network.

4. Domain Adaptation(DA)

All machine models are driven by data. The performance of machine learning models is highly dependent on data, especially when using deep learning methods. One of the biggest challenges for machine learning is that the data distribution drawn for training is more or less divergent from the real distribution of the target test dataset. DA deals with this kind of scenarios, in which a model trained on a source dataset is to be used on a target dataset that has a different but related distribution compared to the source dataset. According to [58], DA is one sub-task of transfer learning and it assumpts that labeled data is only available for source domain.

In this work, DA on deep learning models will be discussed. DNNs aim to learn higher level of semantic features than traditional machine learning methods. This makes DA harder because it is challenging to make a DNN only learn the features that are discriminative for a specific task from the insufficient training data. As discussed in the related works, the state-of-the-art solutions for this problem can be categorized into appearance, feature and output alignment based on the "where to align" strategy. Most of these works use the adversarial method to do the alignment. In this work, two non-adversarial and statistic-based methods that are rarely researched for deep DA of semantic segmentation will be explored. Experiments based on these two methods, Maximum Mean Discrepancy[27] and deep Coral[65], will be conducted.

4.1. Notations and Definitions

According to the definition of domain in the section of semantic segmentation, a source domain is defined as $\mathcal{D}_S = \{\mathcal{X}_S, \mathcal{Y}_S, P(X_S), P(X_S, Y_S)\}$, and a target domain as $\mathcal{D}_T = \{\mathcal{X}_T, \mathcal{Y}_T, P(X_T), P(X_T, Y_T)\}$. Similarly, the semantic segmentation task for both domains are defined as $\mathcal{T}_S = \{\mathcal{X}_S, \mathcal{Y}_S, f_S(\cdot)\} = \{\mathcal{X}_S, \mathcal{Y}_S, \hat{P}(Y_S|X_S)\}$ and $\mathcal{T}_T = \{\mathcal{X}_T, \mathcal{Y}_T, f_T(\cdot)\} = \{\mathcal{X}_T, \mathcal{Y}_T, \hat{P}(Y_T|X_T)\}$ respectively. $\hat{P}$ means that this is the learned probability, which is an approximation of the real distribution. Then, the transfer learning and DA can be defined as follows:

Definition 1 ***(Transfer learning(TL))*** *Given a target learning task $\mathcal{T}_T$ in domain $\mathcal{D}_T$ and the auxiliary source learning task $\mathcal{T}_S$ in domain $\mathcal{D}_S$. Transfer learning is defined as $\langle \mathcal{T}_T, \mathcal{D}_T, \mathcal{T}_S, \mathcal{D}_S, f_T(\cdot)\rangle$ which aims to improve the performance of predictive function $f_T(\cdot)$ for learning task $\mathcal{T}_T$ by discover and transfer latent knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ because of the possible difference in any of the 4 components, $\mathcal{X}, \mathcal{Y}, P(X)$ and $P(X, Y)$, between source and target domains.*

Definition 2 ***(Domain Adaptation(DA))*** *Given a TL problem. Domain adaptation is a sub-task of the TL with the conditions of* $\mathcal{X}_S = \mathcal{X}_T$, $\mathcal{Y}_S = \mathcal{Y}_T$, *and* $P(X_S) \neq P(X_T)$, $P(X_S, Y_S) \neq P(X_T, Y_S)$ *for domains and the conditions of* $D_S = \{x_i^S, y_i^S\}_{i=1}^{N_S}$ *and* $D_T = \{x_i^T, -\}_{i=1}^{N_T}$ *for available datasets in both domains.*

Regarding the DA task for semantic segmentation, the conditions for domains in **Definition** 2 mean that the input space has the same dimension and the output space has the same number of classes, only the distributions of the two domains are different. This difference refers to the marginal distribution of the input variable as well as the joint distribution of input and output variables. In addition, a TL problem is called **deep TL**, when $f_T(\cdot)$ is a non-linear function which is represented by a DNN. As for the sub-task DA, it is called **deep DA**.

4.2. Maximum Mean Discrepancy (MMD)

MMD is first proposed to address the Two-Sample Test[27] problem in statistics. A two-sample test is a test performed on the data of two random samples, each independently obtained from a different given population and the purpose of the test is to determine whether the difference between these two populations is statistically significant[69]. To perform the two-sample test, a well behaved function $y = f(\mathbf{x})$ is used. This function can map sample point **x1** and **x2**, that are drawn from very different distributions p and q respectively, to very different values $y1$ and $y2$ so that the distance of the output values, $|y1 - y2|$, can be as large as possible in order to better distinguish the distribution p and q. On the other hand, if the two samples are drawn from the same distribution or very similar distributions, the distance $|y1 - y2|$ has to be as small as possible, and ideally 0. MMD is a metric for measuring difference or discrepancy of two distributions by utilizing a set of such well-behaved functions. According to [27], the definition of MMD is defined as

Definition 3 ***(MMD)*** *Given observations* $X := \{x_1, \ldots, x_m\}$ *and* $Y := \{y_1, \ldots, y_n\}$, *independently and identically distributed(i.i.d.) from distribution p and distribution q, respectively, and notation* $\mathbb{E}_{x\sim p}[f(x)]$ *and* $\mathbb{E}_{y\sim q}[f(y)]$ *as expectations with respect to p and q, respectively. Let* $\mathcal{F}$ *be a class of functions* $f : \mathcal{X} \to \mathbb{R}$, *MMD is defined as*

$$MMD[\mathcal{F}, p, q] := \sup_{f\in\mathcal{F}}(\mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)]) \tag{17}$$

The quality of the MMD depends on the set of smooth functions $\mathcal{F}$ that are used for computing MMD. $\mathcal{F}$ should be "rich enough" to uniquely identify $p = q$, and "restrictive enough" to provide useful finite sample estimates so that MMD can converge as the number of samples increases. The function classes in the unit balls[72] in the characteristic Reproducing Kernel Hilbert Spaces(RKHS)[59], notated as $\mathcal{H}$, are proven in [27] to satisfy both conditions. RKHS is a space of functions and can also be treated as an implicit feature space. To simplify the computation of MMD, kernel and mean kernel embedding are introduced. They are defined as

Definition 4 ***(Kernel)*** *Let $\mathcal{X}$ be a non-empty set. A function k: $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel if there exists a $\mathbb{R}$-Hilbert space and a map $\phi : \mathcal{X} \to \mathcal{H}$ such that $\forall x, x' \in \mathcal{X}$, $k(x, x') := \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$*

Definition 5 ***(Mean kernel embedding(MKE)****[27])* *Given distribution p, non-empty set $\mathcal{X}$ and kernel k: $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$,*

$$\mu_p = \int_{\mathcal{X}} k(x, \cdot) p(dx), \quad \mu_p \sim \mathcal{H}$$

is the KME of p if $\mathbb{E}_{x \sim p} f = \langle f, \mu_p \rangle_{\mathcal{H}}$ for all $f \sim \mathcal{H}$. The estimated KME on the given N samples is

$$\hat{\mu}_p = \frac{1}{N} \sum_{n=1}^{N} k(x_n, \cdot)$$

Based on definition 3 and 5, and assuming the existence of MKEs for p and q are satisfied, the functions $\mathcal{F}$ are in the unit balls of RKHS, the square of MMD distance can be derived into

$$MMD_k^2[\mathcal{F}, p, q] = \|\mu_p - \mu_q\|_{\mathcal{H}}^2 \tag{18}$$

μ_p, μ_q are the MKEs of distribution p and q respectively. Then "kernel trick" is used to avoid evaluation of mean embedding in $\mathcal{H}$ of an infinite number of dimensions.

$$MMD_k^2[\mathcal{F}, p, q] = \mathbb{E}_{x \sim p}[k(x, x')] - 2\mathbb{E}_{x \sim p, y \sim q}[k(x, y)] + \mathbb{E}_{y \sim q}[k(y, y')] \tag{19}$$

gives the kernel version of the squared MMD. x and x' are different samples from the same distribution, and similar for y and y'. k is the characteristic kernel in the universal RKHS, which are dense in the space of bounded continuous functions. Gaussian RKHS is universal and can approximate bounded and continuous functions arbitrarily well. In practice, equation

$$M\hat{M}D_k^2[X, Y] = \frac{1}{m(m-1)} \sum_{i \neq j} k(x_i, x_j) - \frac{2}{mn} \sum_{i \neq j} k(x_i, y_j) + \frac{1}{n(n-1)} \sum_{i \neq j} k(y_i, y_j) \tag{20}$$

is used for the discrete case, which only has a limited number of samples for estimating MMD. One of the most important property of MMD is $MMD_k^2[\mathcal{F}, p, q] = 0$ if and only if $p = q$. But because of sampling variance, $M\hat{M}D_k^2[X, Y]$ always do not equal zero even if $p = q$. The choices of kernels are also very critical for the test power and low test error of two-sample test[28]. So similar to [39] and [45], multiple characteristic kernels are used to estimate MMD distance of two distributions.

With the Gaussian kernel defined in

$$k(x,y) = exp(\frac{-\|x-y\|^2}{2\sigma^2}), \sigma > 0 \tag{21}$$

MMD loss is defined as biased MMD in the equation

$$\begin{aligned} L_{MMD} = \frac{1}{N^2}(&\sum_{i,j=1,i\neq j}^{N} \sum_{\sigma_k} exp(-\frac{1}{2\sigma_k^2}\|A_i^S - A_j^S\|_2^2) \\ -2 &\sum_{i,j=1,i\neq j}^{N} \sum_{\sigma_k} exp(-\frac{1}{2\sigma_k^2}\|A_i^S - A_j^T\|_2^2) \\ + &\sum_{i,j=1,i\neq j}^{N} \sum_{\sigma_k} exp(-\frac{1}{2\sigma_k^2}\|A_i^T - A_j^T\|_2^2)) \end{aligned} \tag{22}$$

based on equation 20. A_i^S and A_i^T are feature samples from source and target domain respectively. N is the number of samples and σ_k is the parameter for the k-th Gaussian kernel. For example, feature maps A^S and A^T from both domains are with shape $c \times h \times w$, then each sample point A_i^S or A_i^T has the feature dimension of c. Feature maps A^S, as well as A^T, has $N = w \times h$ samples.

4.3. Deep Correlation Alignment (Deep CORAL)

DNNs have many layers connected successively. Each layer can change the distribution of the activations from the previous layer. As DNN goes deeper, a small change in the bottom layers will cause a dramatic change in the distributions of top layers. [34] introduced Batch Normalization(BN) to solve this problem and call this phenomenon as Internal Covariate Shift(ICS). BN normalizes the data of each mini-batch to be zero-mean and unit-variance to compensate ICS. However, BN can not align the covariances of the source data and the target data.

Correlation Alignment (CORAL) is first introduced to DA by [66]. It minimizes the domain shift by aligning the second-order statistics of source and target distributions. Figure 10 demonstrates the main principle of CORAL. It describes how datasets with 3 features are aligned in 3d space. Red points are samples from the target domain and blue from the source domain. In the top left plot, features from source and target domain are normalized to zero mean and unit standard deviation, their distribution covariances are still different. CORAL

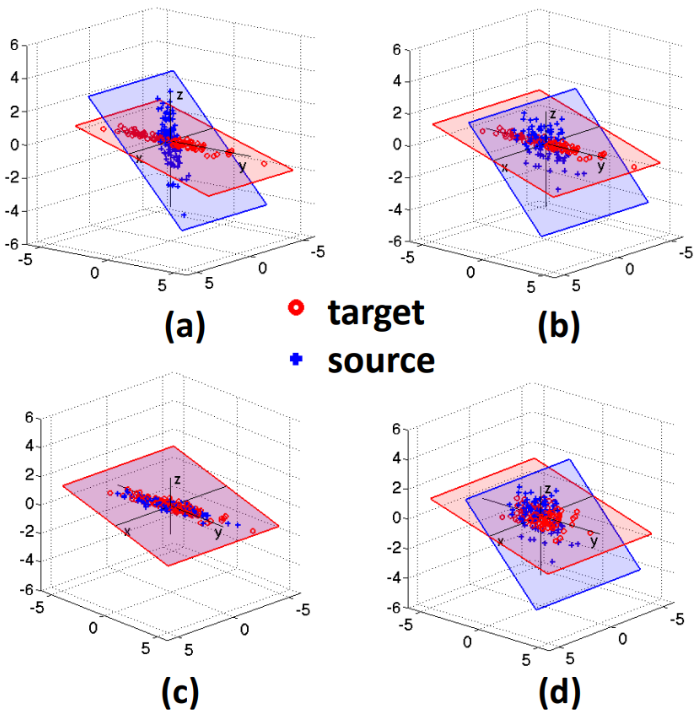


Figure 10: Correlation Alignment[66]

aims to align the covariances and get the result plotted in (c). Instead of de-correlating, in other words, whitening both source and target like plot (d), [66] re-correlates/re-colors the whitened source features with the target covariance.

CORAL is applied for object recognition in [66]. If $\mathbf{x}$ is the d-dimentional feature vector generated by representation $\phi(I)$ with I as the input image, all feature vectors of the n images in the dataset can be written as the matrix $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ in shape $d \times n$. Then the covariance matrix Q of X is

$$Q = \mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^T] = \frac{(X - \mu_X)(X - \mu_X)^T}{n-1} \tag{23}$$

μ_X is the mean of all feature vectors. Mathematically, CORAL can be described with the equation

$$\min_A \|Q_{\hat{S}} - Q_T\|_F^2 = \min_A \|A^T Q_S A - Q_T\|_F^2 \tag{24}$$

in which Q_S, Q_T are the covariance matrices of source and target feature vectors respectively. A is the linear transformation which transforms Q_S into $Q_{\hat{S}}$. And $\|\cdot\|_F^2$ denotes the matrix Frobenius norm. [66] has proofed that the analytically optimal solution

$$A^* = (U_S \Sigma_S^{+\frac{1}{2}} U_S^T)(U_{T[1:r]} \Sigma_{T[1:r]}^{+\frac{1}{2}} U_{T[1:r]}^T) \tag{25}$$

for the minimization problem above is composed of 2 parts. The first part can be regarded as whitening of the source data, the second part re-colors the whitened source data with target covariance. $U_S, \Sigma_S, U_T, \Sigma_T$ are the eigenvectors and eigenvalues of source and target data respectively. Σ^+ is the Moore-Penrose pseudoinverse. Symbol r indicates the smallest rank of Q_S and Q_T.

Instead of computing the analytically linear transformation, deep Coral replaces the linear transformation of CORAL by an implicit nonlinear transformation in the DNN learned by backpropagation and gradient descent. In semantic segmentation, each pixel in the feature maps is regarded as a sample point. If the output activations of one layer are the feature maps of shape $c \times h \times w$, then the number of samples/features is $n = h \times w$, the dimension of each feature vector is $d = c$. The reshaped matrix of feature maps is notated as feature matrix F with shape $n \times d$. Based on this, feature matrix for source and target data can be notated as F_S and F_T respectively. The number of source feature samples is n_S and the number of target feature samples is n_T. Then the unbiased covariances of source and target features can be computed with the following equations.

$$Q_S = \frac{1}{n_S - 1}(F_S^T F_S - \frac{1}{n_S}(F_S)^T(F_S)) \tag{26}$$

$$Q_T = \frac{1}{n_T - 1}(F_T^T F_T - \frac{1}{n_T}(F_T)^T(F_T)) \tag{27}$$

Deep Coral minimizes the CORAL loss

$$L_{CORAL} = \frac{1}{4d^2}\|Q_S - Q_T\|_F^2 \tag{28}$$

to align covariance of source and target features. This loss can be applied to any layer in the semantic segmentation DNN.

4.4. Cycle-Consistent Adversarial Networks(Cycle GAN)

Different from MMD and Deep Coral, Cycle GAN[82] doesn't need the assumption that source and target images can be mapped to the same feature space. It simply transfers the appearance of the images from one domain to another. Compared to the previous work, paired image-to-image translation[36] which learns a function of mapping an input image to an output image from the sufficient training data of paired images, the training images for Cycle GAN do not have to be paired. This is very beneficial because in most applications, it is very hard to obtain paired images from source and target domain. In this thesis, Cycle GAN is applied to conduct experiments to show how Cycle GAN performs on synthetic-to-real DA problems for the BMW dataset and on real-to-real DA problems for remote sensing datasets. Since Cycle GAN consists of two Generative Adversarial Networks(GAN), GAN will be introduced at first and then Cycle GAN will be explained.

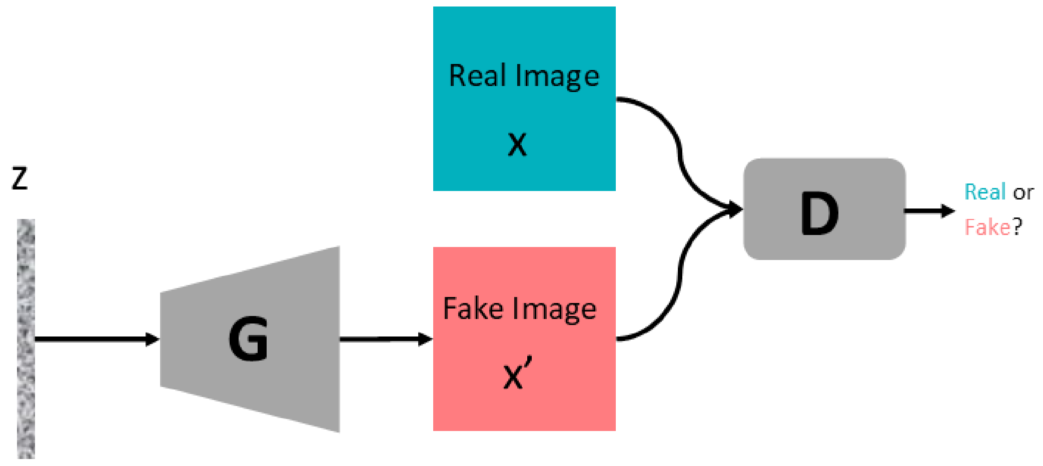


Figure 11: Generative Adversarial Network Schema

GAN is a network, which can learn the marginal distribution of data and generate data from random noise. It is composed by two DNNs, the generator, which can learn the distribution of a given dataset and generate similar data from random Gaussian noise, and the discriminator, which focuses on differentiating the generated data and the real data. These two DNNs act like opponents to each other. Generator aims to generate more realistic data that can "fool" the discriminator, and the discriminator, on the other side, tries to improve its ability to differentiate the real and fake images. Figure 11 shows the idea behind the GAN. The generator network is composed of multiple transposed convolutional layers, and its structure is similar to the decoder(classifier) of Ynet explained in this work. The discriminator network is made by multiple convolutional layers and/or fully connected layers. It can either do image-level classification or pixel-level classification. The 1-dimensional noise vector z is firstly fed to the generator and transferred to a fake image. Then together with real images in one batch or in turn, they are passed to the discriminator to be exterminated for their realness. The parameters of generator and discriminator are updated alternatively by maximizing the discriminator loss

$$L_D = \sum_{x\in X} log D_X(x) + \sum_{z} log(1 - D_X(G(z))) \tag{29}$$

and minimizing the generator loss

$$L_G = \sum_{z} log(1 - D_X(G(z))) \tag{30}$$

There are many different versions of objective functions for calculating the losses, and this is not the emphasis of this work, more extensive reading can be found in [26], [2], [40], [39], etc..

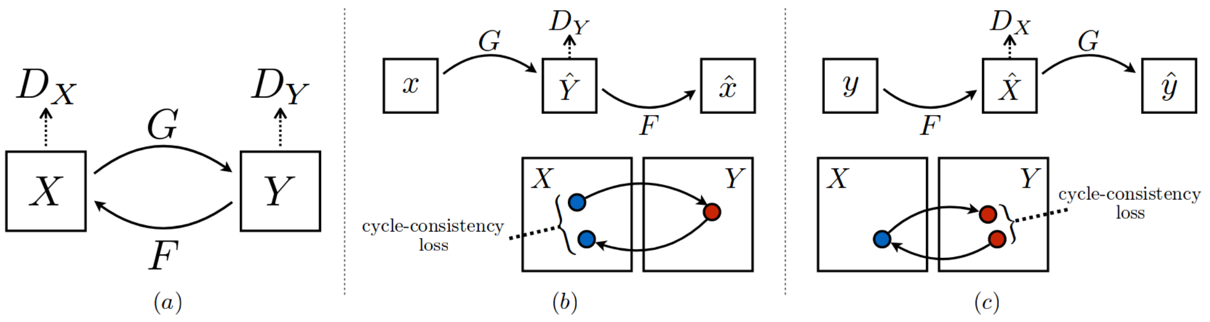


Figure 12: Cycle GAN Schema[82]

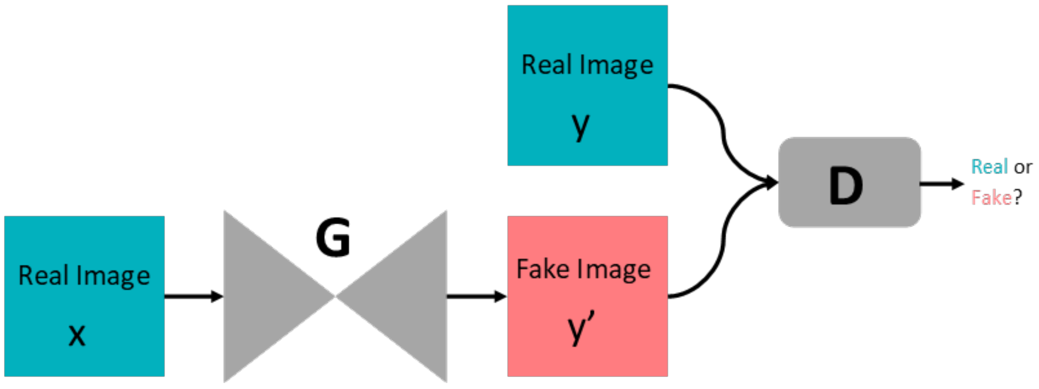


Figure 13: Generative Adversarial Network Schema in Cycle GAN

Cycle GAN contains 2 generators and 2 discriminators as depicted in figure 12(a). G is the forward generator, which maps the input images from domain X into domain Y, and F the backward generator, which maps images in the opposite direction. D_X, D_Y are the two discriminators. D_X learns how to differentiate the generated fake images $F(Y)$ from X and facilitate F to translate images from domain Y into more X-like images, and vice versa for D_Y. The generators in these 2 GANs are different from the GAN explained above, it contains encoder as well as decoder structure and takes in an image as input instead of a random noise vector. The modification is shown in figure 13. Moreover, Cycle GAN introduced two more losses, the cycle consistency loss and the identity loss, which are the constraints for keeping the image structural content during translation. The cycle consistency loss is the L1 distance between the real images and it's "traveled" version, which traveled through the 2 generators and came back again((b) and (c) in figure 12). It can be expressed as

$$L_{cyc}(G,F) = \sum_{x\in X}(\|F(G(x)) - x\|_1) + \sum_{y\in Y}(\|F(G(y)) - y\|_1) \tag{31}$$

The identity loss is optional and is described in equation

$$L_{idt}(G,F) = \sum_{x\in X}(\|G(x) - x\|_1) + \sum_{y\in Y}(\|F(y) - y\|_1) \tag{32}$$

The final objective function is shown by equation

$$L_{cycleGAN} = L_{GAN}(G, D_X, X, Y) + L_{GAN}(F, D_Y, Y, X) + \lambda L_{cyc}(G, F) + \gamma L_{idt}(G, F) \quad (33)$$

The parameter λ and γ are the hyperparamters to control the relative importance between the GAN losses, the cycle consistency loss and the idendity loss. GAN loss L_{GAN} is optimized by maximizing discriminator loss and minimizing generator loss described in equation 29 and 30. But, instead of log likelihood [82] uses least-squares loss[51] for training discriminator D and generator G. The losses for $L_{GAN}(G, D, X, Y)$ are given in the following equations.

$$L_D = \sum_{y \in Y}((D(y) - 1)^2) + \sum_{x \in X}(D(G(x))^2) \quad (34)$$

$$L_G = \sum_{x \in X}((D(G(x)) - 1)^2) \quad (35)$$

The discriminator and generator loss of both $L_{GAN}(G, D_X, X, Y)$ and $L_{GAN}(F, D_Y, Y, X)$ can be derived from the equation 34 and 35 by replacing the correspondent letters.

In this work, the same structure of the Cycle GAN in the original work is used to adapt the appearance of the images from the source domain to the target domain. More details about this network can be found in [82].

4.5. Methodology for DA

In this part, five methods for DA using the techniques mentioned above will be introduced. Approach 1 and 2 uses deep Coral and MMD, respectively, to match the distributions of feature maps of an intermediate layer. In approach 3, deep Coral is used to match the output logit. In approach 4, deep Coral is used to match the distributions of both the intermediate feature maps and the output logits. The last approach is utilizing Cycle GAN to adapt image appearances.

4.5.1. Matching feature maps of an intermediate layer n of the encoder

It is shown in [74] that feature matching of intermediate layers can achieve better results than using early layers because these are heavily correlated with input and late layers are more correlated with the labels. Experiments of matching other layers of Ynet also show degraded performance than matching intermediate layers, so the configuration of matching the intermediate layer is adopted for feature matching with MMD and deep Coral. Figure 14 gives an overview of this adaptation strategy. For the source and target task, two differently parameterized Ynets are used. One Ynet is firstly trained on the source dataset, this trained

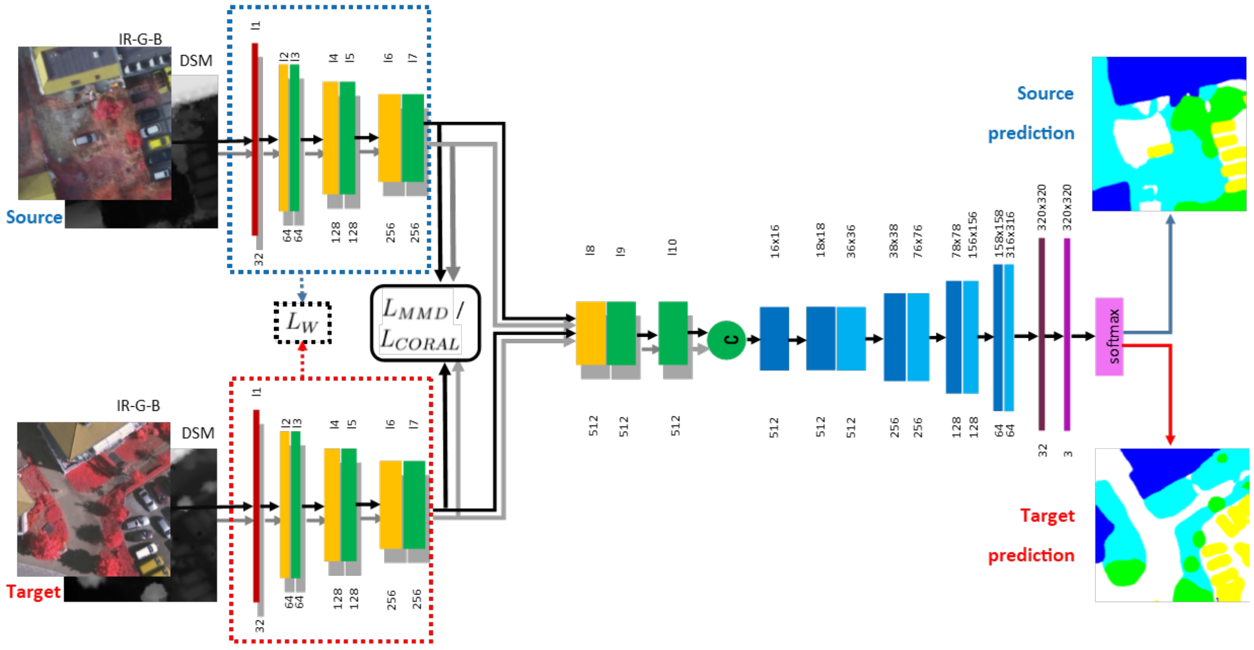


Figure 14: Domain Adaptation with feature matching on intermediate layer n of encoder

network is called source network. Then the weights of the source network is copied to another Ynet which is called target network, and only the parameters of layer 1 to layer n of the target network are to be updated. Source and target network share the parameters of the layers after layer n. This is shown in figure 14 with an example of n=7. Blue indicates source, red indicates target. The weights of layers in the blue dashed rectangle of the source network are fixed, only the weights of layers in the red dashed rectangle of the target network are to be adapted. The weights of the remaining part of the Ynet are shared by the source and the target network. To update the weights of layers in the red dashed rectangle, constraints of MMD or deep Coral loss, as well as the L1 distance between the weights of the source and the target layers in the dashed rectangles, are used. The loss function is summarized in equation

$$L_{mmd,ln} = \alpha L_{MMD} + \beta L_{W,ln} \tag{36}$$

and

$$L_{crl,ln} = \alpha L_{CORAL} + \beta L_{W,ln} \tag{37}$$

L_{MMD} and L_{CORAL} are given in equation 22 and 28 respectively. They both operate on the activations of layer n. $L_{W,ln}$ is the L1 weights loss between source and target parameters expressed in equation

$$L_{W,ln} = \frac{1}{N} \sum |\theta^S_{l1-ln} - \theta^T_{l1-ln}| \tag{38}$$

This constraint is used to preventing target weights from drifting away too much from source weights. θ^S_{l1-ln} and θ^T_{l1-ln} are the parameters in the first n layers of source and target network respectively. N is the number of weights in the first n layers. Hyper-parameters α and β are used to balance $L_{MMD}(L_{CORAL})$ and L_W. In the coming chapter, notations MMD_{ln} and Crl_{ln} are used to identify the experiments of matching activations of layer n by minimizing $L_{mmd,ln}$ and $L_{crl,ln}$ respectively.

4.5.2. Matching output logits

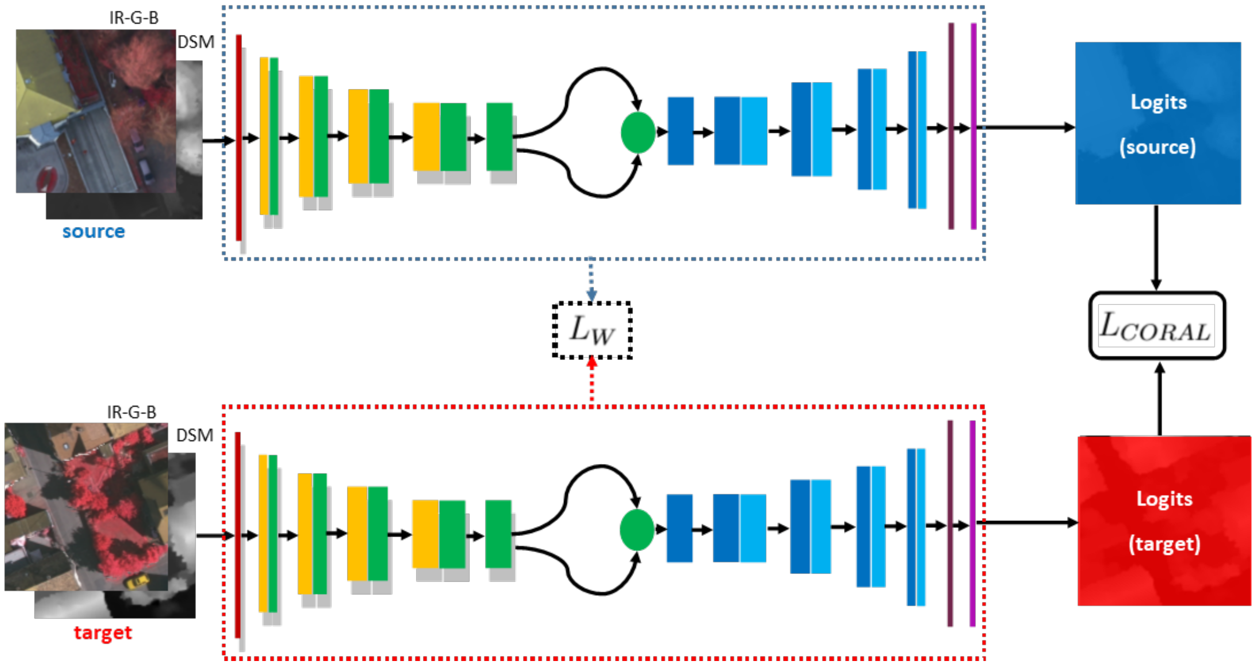


Figure 15: Domain Adaptation with feature matching on output digits

According to the configuration in [65], the weights of the whole network are updated with the gradients of the deep Coral loss between logits output of source and target data and the cross-entropy loss between source predictions and ground truth labels. Deep Coral loss on logits output layer is also used for the DA of semantic segmentation in this work as described in figure 15. Similar to Crl_{ln}, the weights of the trained source network are copied to the target network at the beginning of the adaptation. Then the weights of the target network are updated by minimizing loss

$$L_{crl,log} = \alpha L_{CORAL} + \beta L_W \tag{39}$$

L_{CORAL} operates here on logits outputs. On the right side of figure 15, the logits output of source data is marked with blue and target with red. L_W is the averaged $L1$ norm between the weights of the source network θ^S(blue dashed rectangle in figure 15) and that of the target network θ^T(red dashed rectangle in figure 15), which is shown in equation

$$L_W = \frac{1}{N} \sum |\theta^S - \theta^T| \tag{40}$$

N here is the number of all parameters of Ynet. The method explained above is notated as Crl_{log}.

4.5.3. Appearance adaptation with Cycle GAN

Different from the previous two methods, which adapt the target network based on the trained source network, Cycle GAN translates the images before training FCN. This method can be regarded as a pre-processing step and can be applied additionally to any DA methods discussed above. The approach of using Cycle GAN for DA can be described in 3 steps. First, both source images and target images are used to train the Cycle GAN. Sencond, the source images are fed to the trained Cycle GAN and translated to the target domain. Namely, source images are rendered with the style/appearance of target images. At last, the original source images and the translated source images are used to train FCN from scratch and this FCN is regarded as the target network that is used to predict the labels of target images.

5. Experiments and Results

In this section, the datasets that are used in this work will be described. Then, the experiment settings for semantic segmentation and DA approaches that are explained in the previous section will be given. The results and evaluation of all experiments will be dicussed in the last subsection.

5.1. Datasets

In this work, 3 datasets are used to conduct the experiments. Two(3Cities, ISPRS) are remote sensing datasets, which contain aerial images and DSM images. The third one is the BMW dataset, which contains synthetic and real images of car components.

3City dataset has 3 sub-datasets, each sub-dataset is obtained by taking images of one single city and is regarded as one domain. These 3 cities are Buxtehude(C1), Hannover(C2) and Nienburg(C3) in Germany. Each city has 9 patches of size 3333x3333 pixels, they are cut from one big image of size 10000x10000 pixels. The center patch is taken as the test set and the rest as the training set. Each patch contains 2 images and 1 corresponding label map. The first image is the IR-G-B image, which contains 3 channels, namely InfraRed(IR), Green(G) and Blue(B). The second image is the normalized DSM image, which records the height values above the ground. And the label map is the manually annotated pixel-level label map. The label maps have 3 classes, "ground", "tree" and "building". Figure 16 shows one example patch in this dataset. The left image in the figure is IR-G-B image, the second DSM image is normalized for visualization, and the right label map image is plotted with colors, white indicates ground, green means tree and orange means building. 3City dataset has the Ground Sampling Distance(GSD) of 20cm, which means each pixel covers a 20cmx20cm patch on the ground.

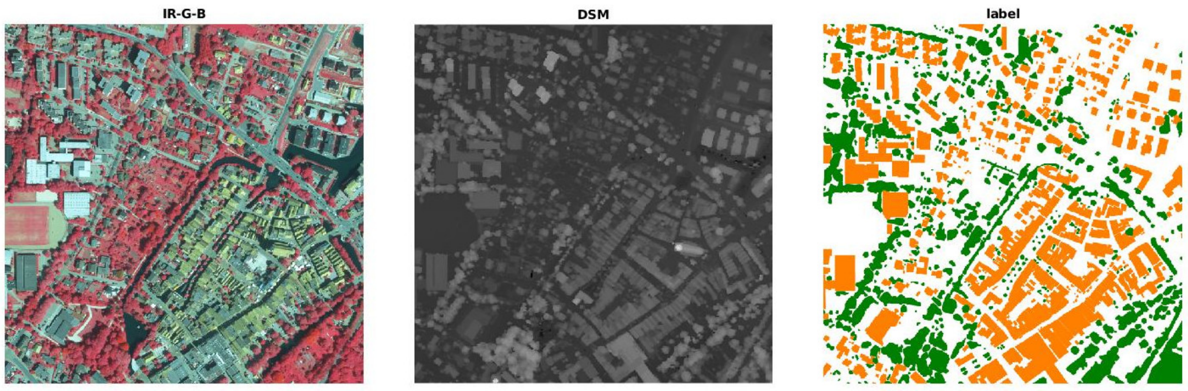


Figure 16: One patch and the corresponding reference of 3City Dataset. Class "ground" is in white, "tree" in green, and "building" in orange

ISPRS Benchmark[1] for 2D semantic labelling consists of 2 sub-datasets(2 domains), Potsdam and Vaihingen. Potsdam has 38 patches and Vaihingen 33 patches. Different patches may have different sizes, the average size is about 2000x2000 pixels. Each patch contains IR-G-B, DSM and label images, but the DSM images of this dataset are not normalized. The official dataset split is used in this work. It has a better label distribution balance between train and test splits. The test sets of both cities are split again into validation and test set. Table 1 gives the detail about the split. The number in the table is patch number. Images of Potsdam have the GSD of 5cm, and Vaihingen 8cm. In order to make GSD the same for both cities, Potsdam is re-sampled to the GSD of 8cm. Patch 1 of Potsdam is shown in figure 17 as an example of this dataset. Different from 3City, ISPRS classifies pixels into 6 categories, they are impervious surfaces, building, low vegetation, tree, car and clutter, respectively. However, Class clutter contains different objects and has many different or even unpredictable features, it influences the DA result a lot, so this class is merged into the class impervious surface for all experiments in this work.

Table 1: ISPRS dataset split

city	training	validation	test
Potsdam	1, 2, 3, 6, 7, 8, 11, 12, 13, 17, 18, 19, 23, 24, 25, 29, 30, 31, 32, 33, 34, 36, 37, 38	4, 5, 9, 10	14, 15, 16, 20, 21, 22, 26, 27, 28, 35
Vaihingen	1, 3, 5, 7, 9, 12, 14, 16, 18, 20, 21, 23, 25, 27, 30, 32	2, 4, 6, 8	10, 11, 13, 15, 17, 19, 22, 24, 26, 28, 29, 31, 33

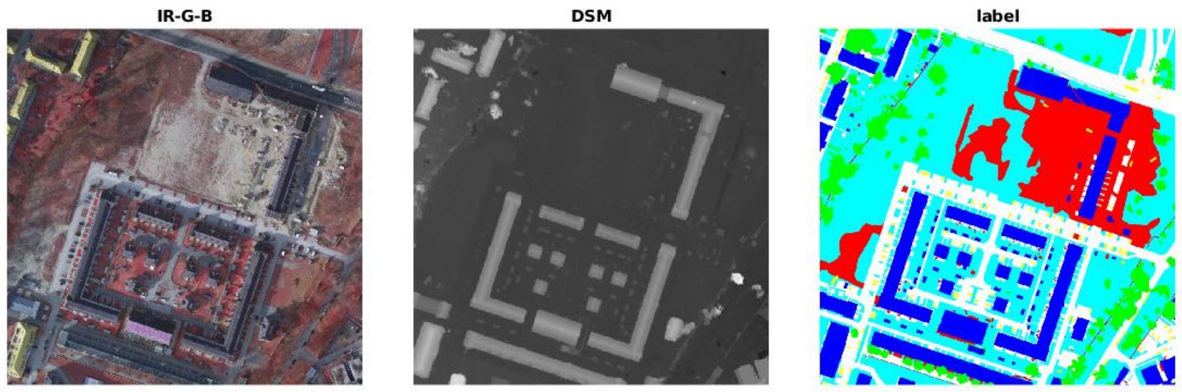


Figure 17: Patch 1 of Potsdam. Class "impervious surfaces" is in white, "building" in blue, "low vegetation" in cyan, "tree" in green, "car" in yellow and "clutter" in red.

For both 3Ctiy and ISPRS, 200 random images of size 320x320 are cropped from each patch for training and validation set. For the test set, the original sized patches are kept, but it will be cropped during testing. In the end, 1600 training images and 1 test patch are obtained for each city of 3City dataset. Potsdam has 7200 training images, 1936 validation images, and 10 test patches. Vaihingen has 4800 training images, 605 validation images, and 13 test patches.

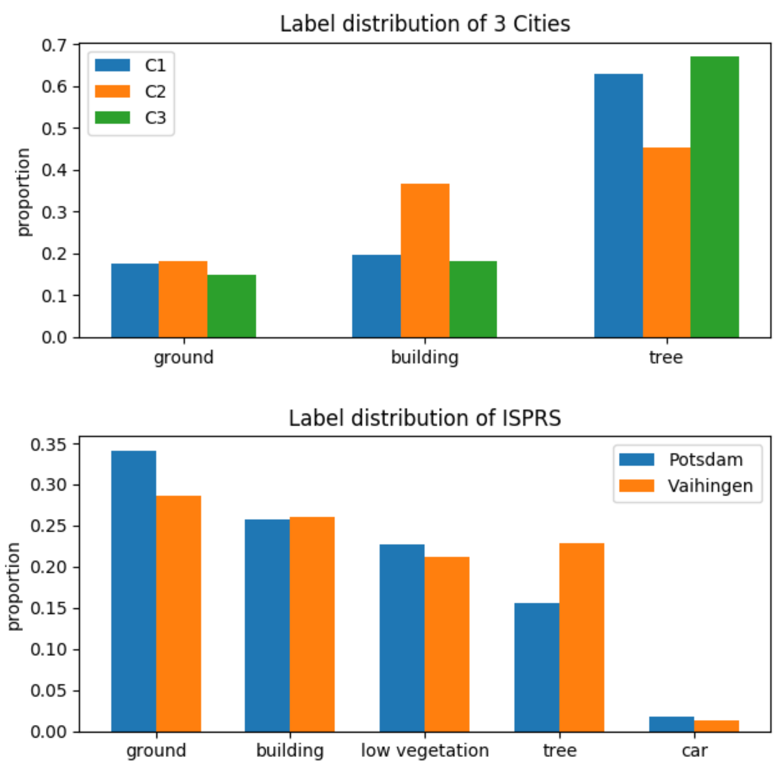


Figure 18: Label distributions in dataset 3City and ISPRS

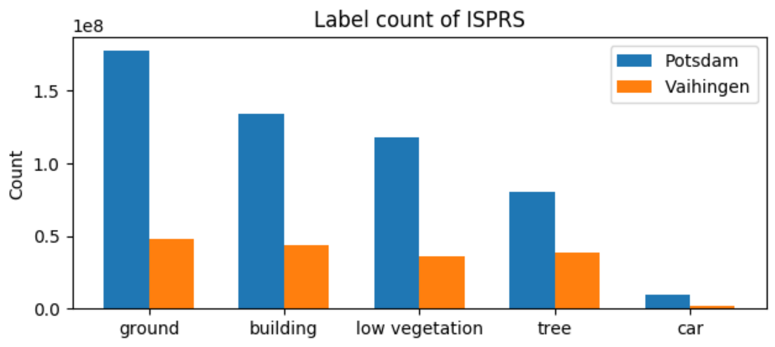


Figure 19: Label count of dataset ISPRS

The label distributions for these two remote sensing datasets are shown in figure 18. It shows that the amount of samples is imbalanced across different classes. In 3City dataset, the samples of the class tree are much more than the other two classes. The label distributions for C1 and C3 are similar while C2 has more buildings and fewer trees than them. In the bottom subplot of figure 18, the label distribution of ISPRS is also imbalanced, especially for the minority class "car". The biggest difference of label distribution between Potsdam and Vaihingen occurs at class "ground" and "tree". Different from 3City dataset, which has the same amount of samples for each city, ISPRS has many more samples in Potsdam than in Vaihingen as shown in figure 19. This might also influence the result of domain adaptation.

BMW dataset has two sub-sets, the real set, and the synthetic set. Real set contains 76 RGB images taken by realsense camera, and 80 synthetic images are generated by Unity using 3d models, they are shown in figure 20. Each image has a size of 720x1280 pixels and shows the car components in a container. There are many different car components, but only one component, which is called "combox", is used in the experiments of this work. The pixels in these images are classified into 3 categories, namely, combox, container, and background.

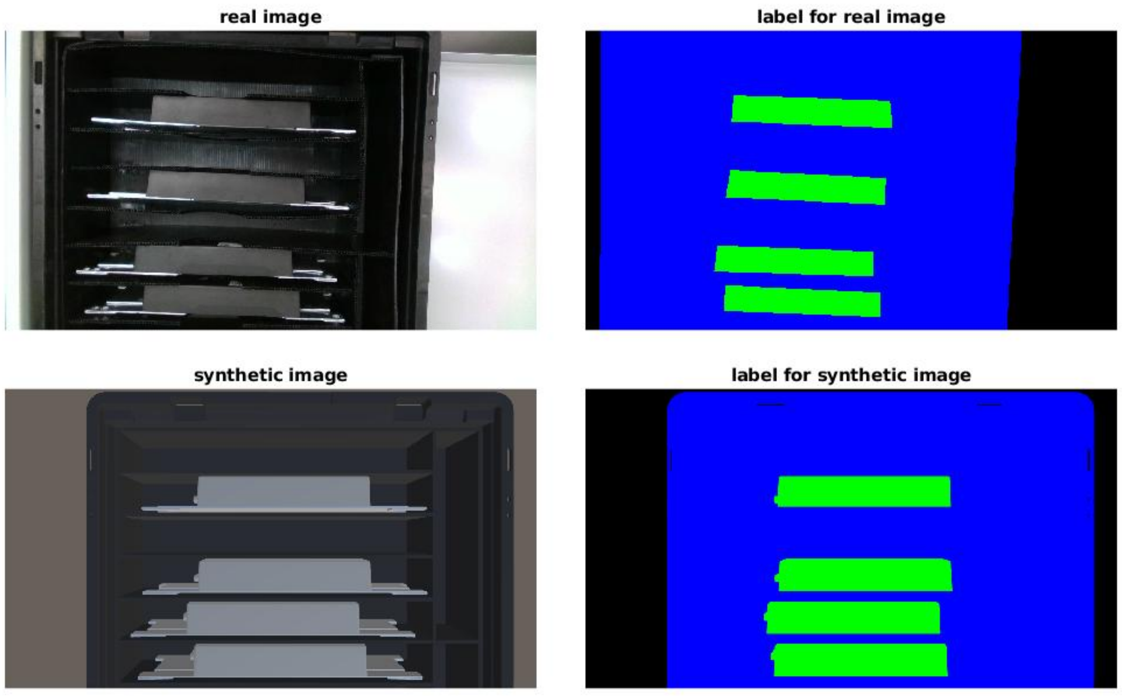


Figure 20: real and syntheic images with it's correspondent label mask. Class "combox" is in green, "container" in blue and "background" in black

5.2. Experiments

Implementation details of training FCNs for semantic segmentation on the datasets explained above will be given in the first part of this subsection. Then the settings for DA experiments using MMD and deep Coral will be described. At last, the experiments of DA using Cycle GAN will be explained.

5.2.1. Semantic segmentation

In this part, the experiment settings of semantic segmentation without DA will be given. The experiments on dataset 3City and ISPRS will be explained first. Then the experiments for BMW data will be discussed.

On 3City and ISPRS dataset, 5 Ynets are trained, one for each city. The data are augmented by random flipping vertically or horizontally, and then by randomly rotating by 0, 90,180, or 270 degrees followed by channel-wise normalization with mean and standard deviation calculated over the whole sub-dataset(domain). To further augment the dataset, online augmentation used in [74] is also applied in this work during training based on the observation that this method can decrease the accuracy drop when test on the target dataset. Each sample is randomly scaled and shifted according to distribution $s \sim N(1.0, 0.1)$ and $h \sim N(0.0, 0.1)$ respectively.

The weights are optimized using Adam optimizer with parameters 0.95 and 0.999 for exponential decay rate of the first- and the second-moment estimates respectively. The learning rate during the whole learning process is 0.0001 and the mini-batch size for each iteration is 4. Each Ynet is trained for 300000 iterations. In each domain of 3 City, the model with the best training accuracy is chosen for testing, and in each domain of ISPRS, the model with the best validation accuracy is chosen. For testing, the sliding-window strategy in [74] is used because this overlapped prediction strategy can improve the OA 1-3%. The window slides with a step size of 160 pixels. In each step, one cropped image is normalized and fed to the network to generate pixel-wise class scores. This leads to multiple predictions on each pixel, from which all predictions are aggregated and the most confident classes are chosen for the final prediction maps. In other words, each pixel is predicted 1 to 3 times, the scores for each pixel are summed up and the most confident class is finally assigned to this pixel.

For the BMW dataset, the same network structure of Ynet is used, but only one encoder is used for the 3 channel inputs. This network is simply called FCNb(”b” means basic). Two FCNbs will be trained, one on the synthetic dataset, one on the real dataset. In the pre-processing step, the training images are randomly flipped vertically or horizontally and then normalized channel-wise by the corresponding mean and standard deviation calculated over all images of the synthetic dataset or the real dataset. The same online augmentation explained above is also used for images of the BMW dataset. The configurations for parameter optimizer, learning rate, and mini-batch size are the same as for the experiments of 3City and ISPRS. Since the test images have the same shape as the training images, no sliding window is needed. Test images are simply normalized and then fed to the network.

5.2.2. DA with MMD and Deep Coral

In 3City dataset, there are 3 cities. Each adaptation chooses one city as source dataset and one as target dataset. So there are 6 adaptations, namely $C1 \rightarrow C2, C1 \rightarrow C3, C2 \rightarrow C1, C2 \rightarrow C3, C3 \rightarrow C1, C3 \rightarrow C2$. In ISPRS, there are only 2 adaptations, Potsdam→ Vaihingen and Vaihingen→ Potsdam. In this section, the experiment settings of DA will be described. All experiments of the method with MMD and deep Coral are performed on all adaptations of RS data. The DA method with Cycle GAN will be tested on both BMW data and RS data. For BMW data, the DA is performed from the synthetic domain to the real domain. For RS data, this method will be performed on 4 adaptations, namely 2 DAs between C2 and C3 and 2 between Potsdam and Vaihingen.

MMD on layer 7

Start from the pre-trained Ynets, DA is achieved in this experiment with MMD by matching the feature activations of an intermediate layer n. Specifically, layer 7 of Ynet is taken for the feature matching because this layer is not very close to the input and saves more computation time than the previous two layers which have more pixels. Besides, it also contains more spatial information than the last 3 layers of the encoder. This experiment is notated as MMD_{l7}. The same pre-processing strategy of pre-training the Ynet is used for DA but without online augmentation. The optimizer used for adaptation is still Adam. Only the exponential decay rate for the first moment estimates of Adam is decreased to 0.5. Since the weights only need to be modified slightly, the learning rate is decreased to $10e^{-5}$. In order to balance the input distribution in each iteration, the mini-batch size is as large as possible. On a 32GB GPU, the biggest mini-batch size can only reach 6, because multi-kernel MMD is very computationally expensive. Since it is hard to find a criterion to terminate the training process of DA, the networks are trained for 50 epochs for all adaptations on 3City dataset and 30 epochs on ISPRS because ISPRS has much more samples than 3City. The hyperparameter β for $l_{W,l7}(n = 7)$ in equation 22 is set to 100. If bigger learning rate $10e^{-4}$ is used instead of $10e^{-5}$, β should also be increased to about 1000. A sign of stable adaptation is that the $l_{W,l7}$ can enter into a stable state within several epochs. In other words, if $l_{W,l7}$ keeps increasing, it is very likely to get negative transfer after several epochs. Since the determination for α is also very tricky, the experiment MMD_{l7} will be conducted with different α values, namely $\alpha = 0.0005, 0.001, 0.005$. Moreover, the Gaussian kernels of MMD are parameterized with $\sigma = [1, 2, 4, 8, 16]$, which is the same as in [39].

Deep Coral on layer 7 and logits

Using deep Coral, experiments are conducted by aligning covariance of activations of the intermediate layer n or logits output of the last layer before softmax. Same as MMD_{l7}, layer n=7 is taken as the intermediate layer for experiment of deep Coral. The two experiments of aligning layer 7 and logits are notated as $Coral_{l7}$ and $Coral_{log}$ respectively. Since the alignment on both layers simultaneously shows worse result than any of them alone, this configuration will not be discussed. Instead, one more experiment is conducted by aligning these two layers jointly based on the observation that $Coral_{log}$ can mostly obtain accuracy gain at the first 15 epochs and the results are also not sensitive to the parameter α. Moreover, after $Coral_{l7}$, the parameters of the last 3 layers of the encoder and the parameters of the decoder are still not adapted, the performance of the target network is possible to be further optimized by aligning logits starting from the model that is adapted by $Coral_{l7}$. Namely, the pre-trained semantic segmentation network is first adapted with $Coral_{l7}$ and then followed by the adaptation $Coral_{log}$. This experiment is notated as $Coral_{l7,log}$. For $Coral_{l7}$, ablation test is done with $\alpha = 100, 200, 400, 800$. The experiments of $Coral_{log}$ are conducted with $\alpha = 0.01, 0.001, 0.0001, 0.00001$ and the learning rate is set to $10e^{-6}$. The α values for $Coral_{log}$ are much smaller because the pre-trained Ynet has bigger values as well as bigger covariance in the output of the last layer of a decoder than the 7-th layer of the encoder. The deep coral loss L_{CORAL} is much larger for this configuration. In order to balance this term with l_W, small α values are used. The other hyperparameters not mentioned for $Coral_{l7}$ and $Coral_{log}$ keep the same as for MMD_{l7}. For $Coral_{l7,log}$, $\alpha = 200$ is chosen for adapting layer 7 and $\alpha = 0.01$ is used for adapting logits, other configurations keep the same as $Coral_{l7}$ and $Coral_{log}$.

5.2.3. Appearance Adaptation with Cycle GAN on input images

Experiments of DA with Cycle GAN are conducted by changing the appearance of images. It can be very helpful to translate the images from synthetic domain to the real domain so that the network can be trained on the automatically labeled and translated fake images in stead of real images that should be labeled manually. For example, there are thousands of parts for a car, in order to automate the whole process of the logistic transportation of these parts in the factory, the recognition of them is necessary. But annotating all these components is very time-consuming. So in this work, the images of one example car component, namely, the BMW dataset that contains the component "combox" is taken for the experiment of transferring the appearance information from the real data to the synthetic data. This method can be applied to other car components straight forward.

In addition to synthetic-to-real adaptation for BMW data, more experiments on RS data are conducted to test if Cycle GAN can also help DA between 2 RS domains that only contains real images. The combinations of Potsdam-Vaihingen and C2-C3 are chosen because they have a different type of domain discrepancies compared with BMW data. As shown in figure 20, synthetic images are different from real images mostly in the appearance aspect. The structure and geometry of the objects in the scene are roughly the same. However, in

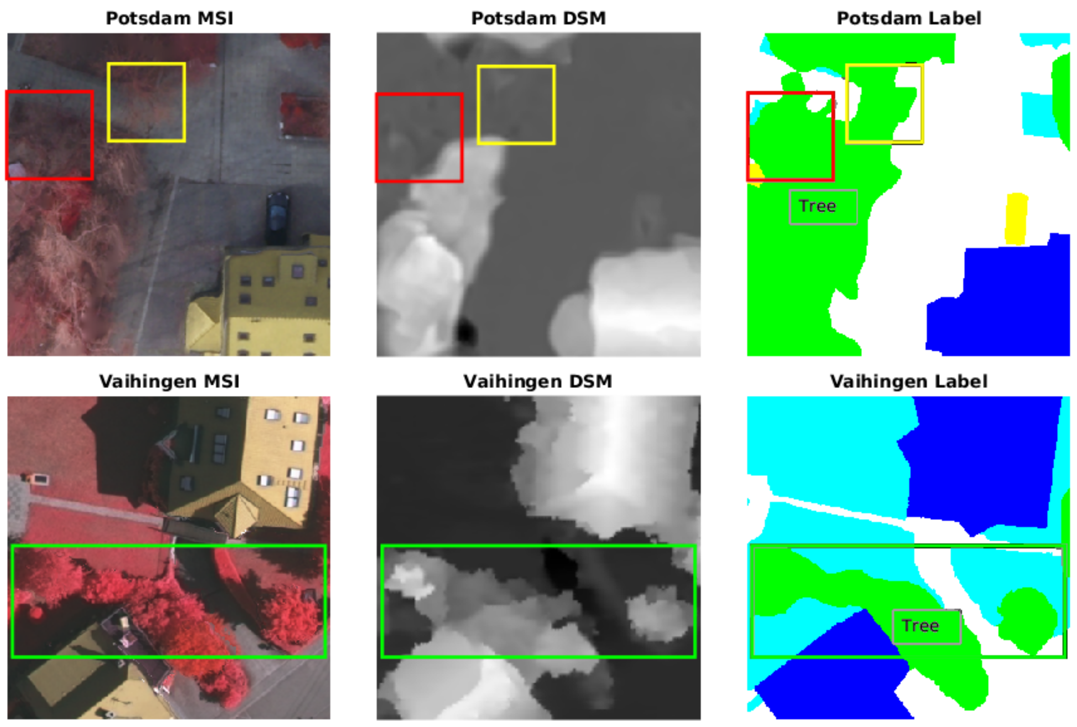


Figure 21: Comparison on Potsdam and Vaihingen training data

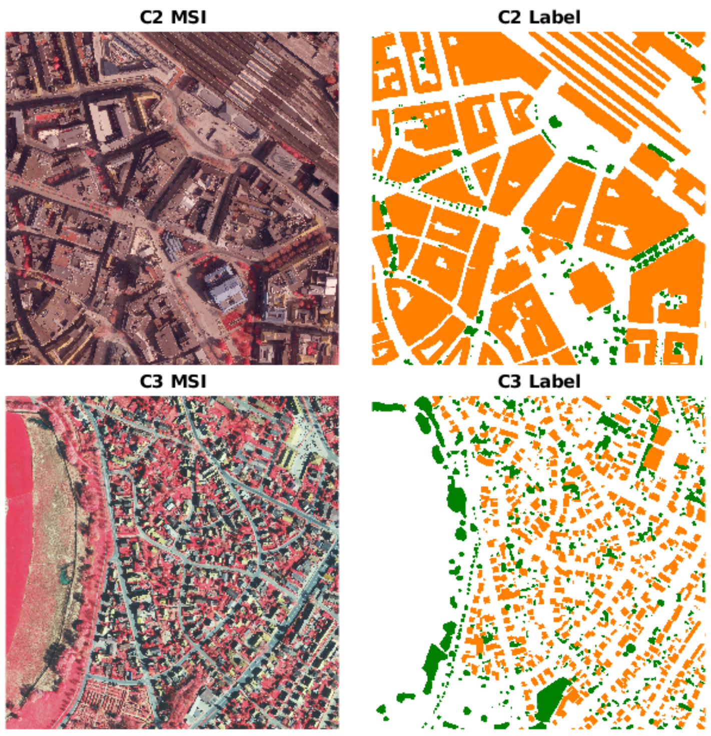


Figure 22: Comparison on C2 and C3 training data

addition to the global appearance change, there is a category-specific difference in structure or content between the trees of Potsdam and Vaihingen because of season change. Figure 21 shows the difference, trees of domain Potsdam in the red and yellow rectangles look totally different from that in the green rectangle of Vaihingen. Moreover, trees without leaves in domain Potsdam even don't have any noticeable clue in the DSM image when the trees in the DSM image of Vaihingen is more identical to the label image. Besides, C2-C3 combination is chosen for this experiment because these two domains have different label distributions and object sizes just as shown in figure 22. It is very obvious that the buildings in C2 are much larger and denser than that in C3, and on the other hand, C3 has a large size of crop fields and more trees.

In the experiments of the method with Cycle GAN, the appearance of source images are first translated to that of the target images. Then both the translated source images and the original source images are used to train the target networks from scratch. In this work, Cycle GAN uses 9block-Resnet as the generator, one layer of Convolution-LeakyReLU block and three layers of Convolution-BatchNorm-LeakyReLU block for the discriminator. More details about this network can be found in the code[1] of [82]. In order to keep the structural content of the images, the parameter lambda identity of Cycle GAN is set to 1. The original size of BMW images are 720x1280 pixels, they are pre-processed by resizing to 360x640 pixels and randomly flipped horizontally and vertically. Since only about 80 images are available for the synthetic domain as well as for the real domain, the Cycle GAN network is trained for 200 epochs. The same configuration is used for training Cycle GAN on RS data, except that the number of training epochs is decreased to 10 because many more images are available in domains of RS dataset than the BMW dataset.

After image translation, the models of epoch 120, 150 and 200 are used for generating the BMW fake images from synthetic images. Then a FCNb is trained from scratch on the new extended dataset, which contains about 80 synthetic images and 240 fake images. For the RS dataset, the fake images are generated by using the trained Cycle GAN of epoch 10. Then Ynets are trained from scratch by using the extended real-plus-fake dataset. Specifically, 4 Ynets are trained based on dataset $\text{C2}_{real}+\text{C2}_{fake}$, $\text{C3}_{real}+\text{C3}_{fake}$, $\text{Potsdam}_{real}+\text{Potsdam}_{fake}$ and $\text{Vaihingen}_{real}+\text{Vaihingen}_{fake}$, respectively. C2_{fake} is obtained by styling images in C2 with the appearance of the images in C3, and C3_{fake} in the reversed direction of Cycle GAN. Potsdam_{fake} and Vaihingen_{fake} are generated in the same way.

5.3. Result and Evaluation

In this section, the result of semantic segmentation without DA will be discussed at first. Then the result of DAs with MMD and deep Coral will be evaluated respectively. In the following subsection, MMD and deep Coral will be compared based on their performances. At last, the result of appearance adaptation using Cycle GAN will be discussed.

[1]https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

5.3.1. Semantic Segmentation without DA

The result of semantic segmentation without DA on 3City will be discussed at first and then followed by the result on ISPRS and BMW dataset. For each combination of 2 domains, the OA, the mean IOU, as well as the class-wise IOU, will be given. OAs and mIOUs in the tables are marked bold.

SD \ TD		C1	C2	C3
C1	OA	**91.1**	**84.3**	**86.8**
	mIOU	**82.3**	**66.1**	**69.1**
	g	75.6	47.6	44.2
	b	85.7	74.8	79.9
	t	85.6	76.0	83.1
C2	OA	**84.0**	**90.8**	**81.3**
	mIOU	**71.0**	**79.3**	**62.6**
	g	63.4	68.3	48.7
	b	74.5	85.3	63.5
	t	75.1	84.2	75.5
C3	OA	**88.0**	**83.3**	**90.6**
	mIOU	**77.1**	**66.9**	**78.9**
	g	70.8	54.1	67.3
	b	79.4	72.7	82.4
	t	81.2	74.0	87.1

Table 2: Test results of Ynet on 3City datasets(%). SD: source domain, TD: target domain

The semantic segmentation test result of 3City is given in table 2. The first column indicates the source domain(SD) and the first row gives the target domain(TD) name. Letter "g", "b" and "t" indicate class "ground", "building" and "tree" respectively. The performance of all 3 cities is over 90% which is only slightly worse than the result in [74]. Considering that the input image size is decreased from 640x640 to 320x320, the skip-connections are omitted and also 2 convolutional/transposed convolutional layers in encoder and decoder are deleted, this network configuration is not only acceptable but also beneficial for saving training time during DA. The average OA drop of the cross-city tests is about 6%. And as expected by observing the difference of label distribution of dataset in figure 18, the accuracy drop between C2 and the other 2 cities is obviously larger than between C1 and C3 themselves, especially from C2 to C3(90.8% to 81.3%). This means, the more labels are differently distributed, the higher the accuracy drop would be observed. However, this conclusion should also be conditioned by similar global and category-wise appearances in both domains base on the observation that datasets(i.e. BMW dataset,) without fulfilling this condition can show an even larger accuracy drop. The class-wise result in table 2 shows that imbalanced label distribution leads to an imbalanced class-wise accuracy. The more samples one class has, the higher the accuracy it would get. In contrast, the class accuracy drop between cities is relatively more random rather than relevant to label distribution. For instance, the IOU of class 0(”ground”) in C1 dropped from 75.6% to 44.2% in C3 when it increased from 67.3% to 70.8% in the inverse direction even the proportion of this label in both cities is similar.

SD \ TD		Potsdam	Vaihingen
Potsdam	OA	**89.0**	**67.4**
	mIOU	**77.1**	**45.8**
	g	83.2	51.5
	b	91.2	41.9
	lv	68.3	51.3
	t	70.0	62.8
	c	73.0	21.4
Vaihingen	OA	**71.6**	**87.3**
	mIOU	**54.4**	**72.9**
	g	60.6	80.9
	b	71.6	85.7
	lv	42.6	67.3
	t	34.7	74.5
	c	62.2	56.4

Table 3: Test results of Ynet on ISPRS dataset(%).SD: source domain, TD: target domain

The semantic segmentation result of ISPRS is shown in table 3. The first column and the first row indicates the source and the target city respectively. Letter ”g”, ”b”, ”lv”, ”t” and ”c” indicate class ”ground”, ”building”, ”low vegetation”, ”tree” and ”car”. Compared to 3City, ISPRS has a larger accuracy drop. The average OA drop between Potsdam and Vaihingen is nearly 20%, which is three times bigger than the accuracy drop in 3City. And the mIOU drop is even bigger, it is about 25%. This is because these two cities not only have the label distribution difference but also have different features for a specific category because of the season change, which is shown in figure 21. The pattern of class-wise accuracy is very similar to that of 3 City. And it is also very interesting to observe that the accuracy change of the minority class 5(”car”) from the source domain to the target domain. From Potsdam to Vaihingen, the accuracy drops dramatically from 73.0% to 21.4%, but surprisingly, it increased a little bit in the inverse direction. This is similar to the case of class ”ground” in 3City between C1 and C3, only this phenomenon in ISPRS is more obvious because of the bigger difference in the number of car samples. The reason might be that the minority class "car" samples in Vaihingen have much stronger shadows than in Potsdam. The DNN trained on Potsdam can hardly get rid of the influence of car shadows when predict on Vaihingen. In contrast, the DNN can learn more discriminative and general features of cars on Vaihingen.

	synth→synth	real→real	synth→real
OA	**99.9**	**99.7**	**88.0**
mIOU	**99.6**	**98.8**	**61.4**
combox	99.9	97.4	23.2
container	99.9	99.6	84.6
background	99.1	99.5	76.4

Table 4: Semantic segmentation result on BMW data without domain adaptation(in %)

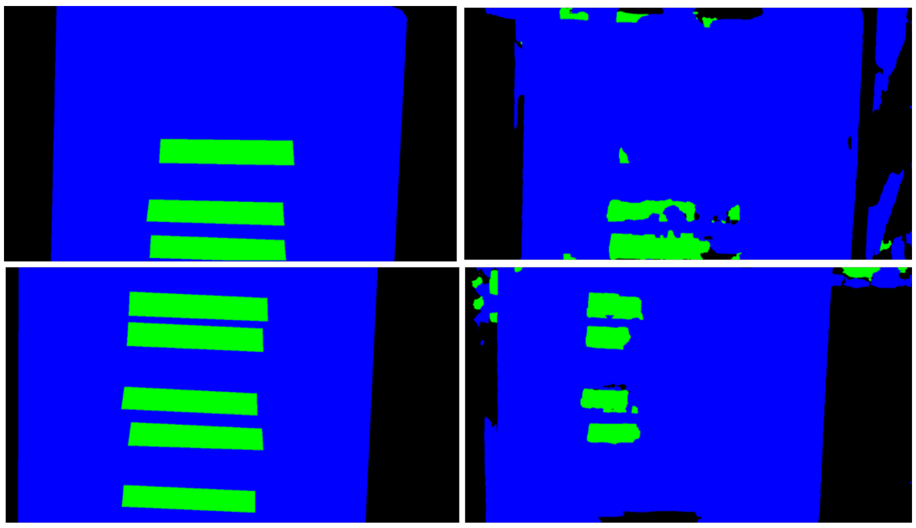


Figure 23: Model trained on synthetic data and tested on real data(left: ground truth, right: predictions on real images)

At last, the semantic segmentation result on BMW data is shown in table 4. The first row indicates the training domain and test domain. Row 2 and 3 give the OAs and mIOUs, the last 3 rows are the class accuracies. The second column is the result of the classifier trained on synthetic data and tested on synthetic data, and similar for the third and the last column. FCN performs quite well on this dataset when it is trained and tested in the same domain since this segmentation problem is much easier than RS data. The shape and size of the objects in this dataset are highly consistent while they are very diverse in RS data. Because of the big appearance difference of "combox" between synthetic and real domain, the IOU accuracy drops dramatically from 99.9% to 23.2%, which then leads to a worse mIOU performance. However, "combox" is a minority class, so the overall accuracy only dropped about 12%, which seems even better than ISPRS. But in real use, "combox" is the most important class to be identified. In figure 23, the predictions of both synthetic images and real images are shown, the result is totally a mess and not acceptable.

5.3.2. DA with MMD and Deep Coral

In this part, the result of MMD_{l7} will be discussed at first. Then the result of DA with deep Coral will be discussed.

MMD on layer 7

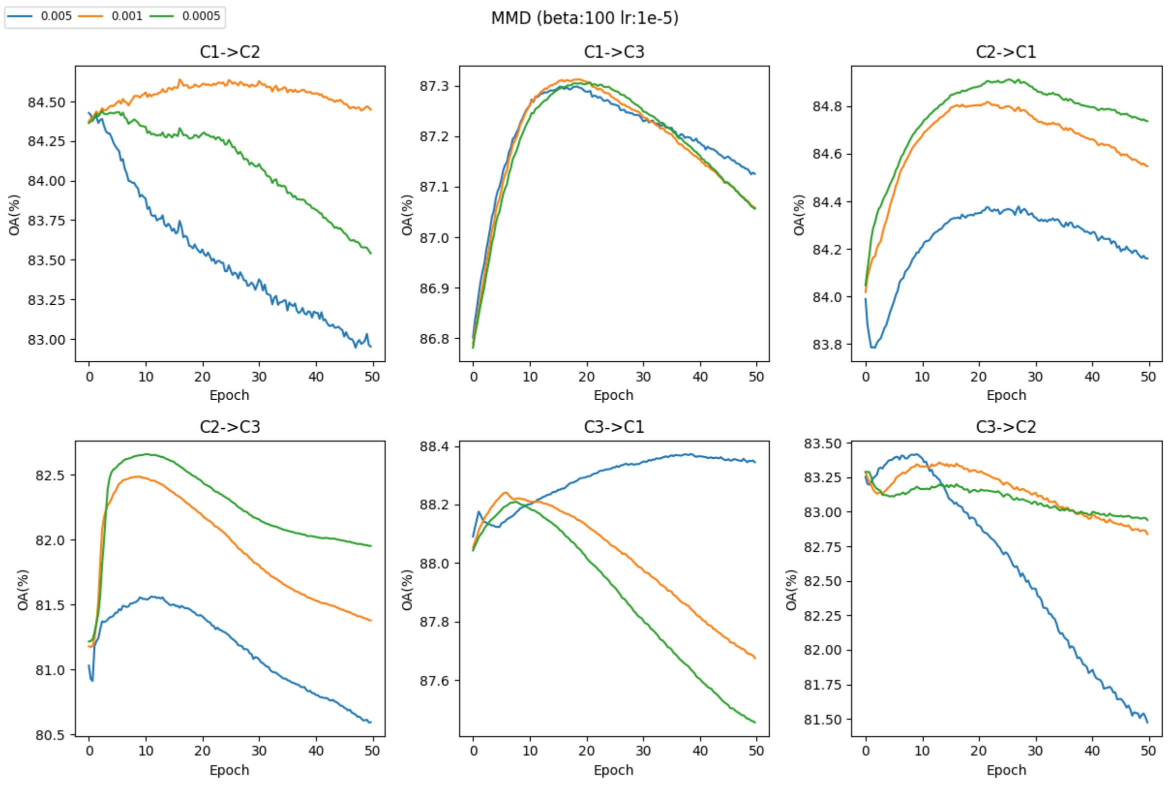


Figure 24: Overall accuracy of adaptation on 3 City with MMD_{l7}

The results of DA with MMD are shown in figure 24 and 25. Each subplot in the figure shows how OAs of the corresponding adaptation, which is notated in the title of each sub-plot, change along the training epochs. Each line in the sub-plot indicates one adaptation with a specific parameter α. It is hard to find a common parameter α that performs the best across all adaptations in both figures. $\alpha = 0.005$ performs the best on adaptation C3→C1 as well as for ISPRS when it performs the worst on other adaptations of 3City dataset. $\alpha = 0.0005$ performs the best for C2→C1 and C2→C3 but the worst for ISPRS. As an average, $\alpha = 0.001$ is taken as the best for the final evaluation. All accuracy gains for 3City are very limited(within 1%). The performance in ISPRS is better, the best performance occurs at $\alpha = 0.005$, which achieves an average OA gain of about 7%.

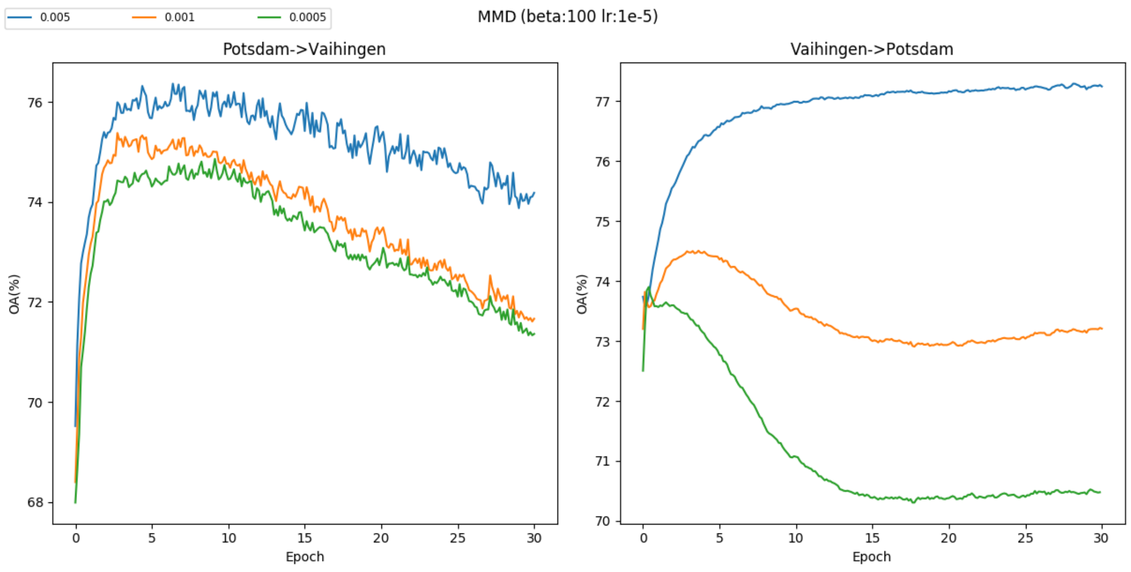


Figure 25: Overall accuracy of adaptation on ISPRS with MMD_{l7}

Deep Coral on layer 7 and logits

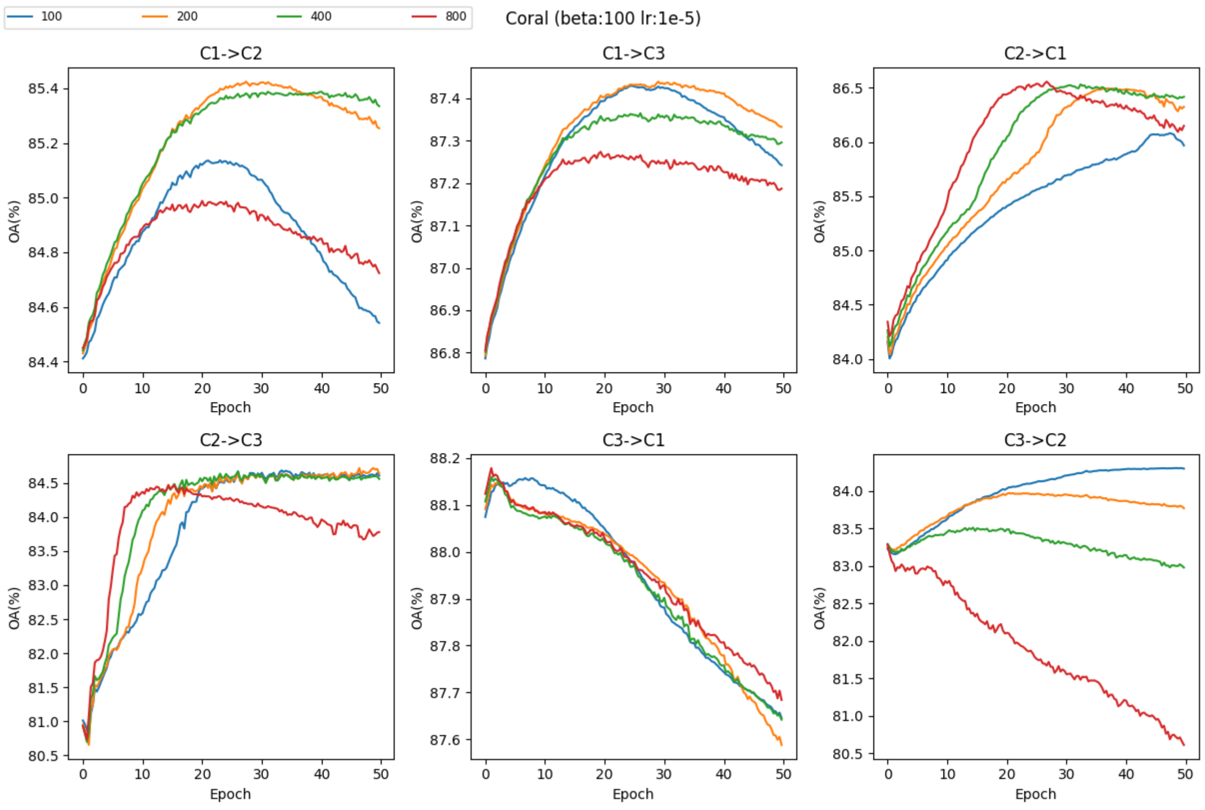


Figure 26: Overall accuracy of experiments with $Coral_{l7}$ on 3City.

The results of ablation test for $Coral_{l7}$ on 3City and ISPRS are shown in figure 26 and 27 respectively. For different parameters, α, the trends of OA changes of the experiments $Coral_{l7}$ are shown in different colors. As shown in both figures, the accuracy could decrease if the networks are trained for too many epochs. For 3City, the average performance is the

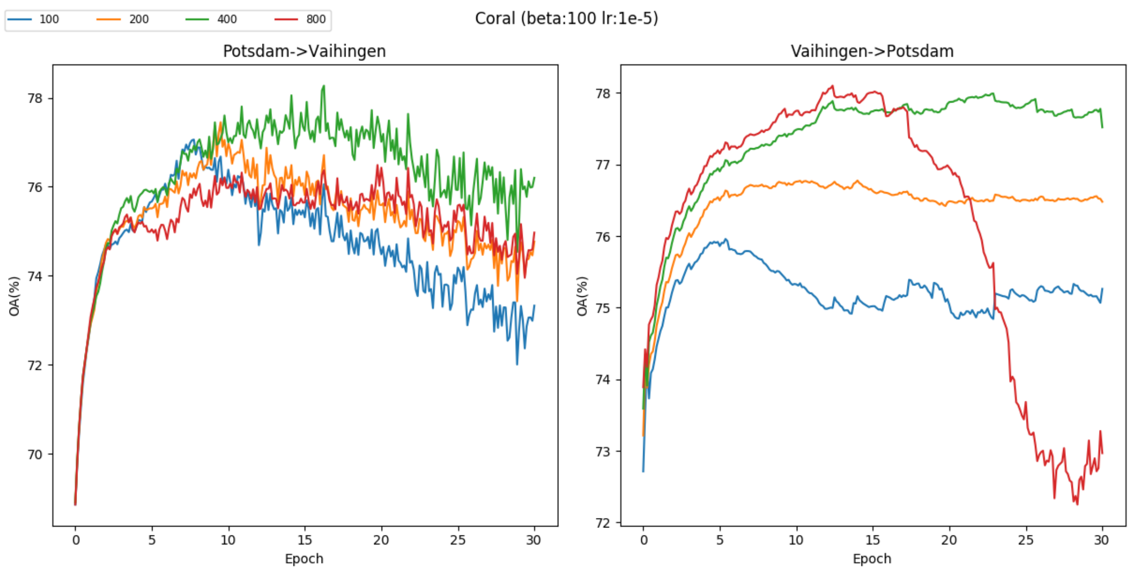


Figure 27: Overall accuracy of experiments with $Coral_{l7}$ on ISPRS.

best when $\alpha = 200$(orange line). If the value of α is too big or too small, the accuracy seems more unstable during adaptation. For ISPRS, the best performance occurs at $\alpha = 400$(green line). Since the accuracy drop for ISPRS is larger than 3City, it is also easier to get more accuracy gain using DA. The accuracy gain for 3City is within 4% while that of ISPRS is more than 7%. As a result, the larger the accuracy drop is, the harder the adaptation will be. The adaptation between C1 and C3 as well as C2 and C3 shows that $Coral_{l7}$ can perform better in one direction but worse in the other direction, and better performance also occurs in the direction which has bigger accuracy gap(C1$\rightarrow$C3, C2$\rightarrow$C3). In the other direction, the accuracy of the model trained on C3 only dropped 3% when the model is tested directly on C1 so that the adaptation accuracy reaches peak very fast and then decreases in the remain epochs. To avoid this, it is better to decrease the learning rate if the source domain and the target domain look very similar both in label distribution and feature distribution.

The results of $Coral_{log}$ on 3City and ISPRS are shown in figure 28 and 29 respectively. It is obvious that $Coral_{log}$ performs much worse than $Coral_{l7}$. The accuracy gain can hardly exceed 1% in 3City and is even negative in any epoch of the adaptation C1$\rightarrow$C3 and Potsdam$\rightarrow$Vaihingen. However, in the reversed direction of these two adaptations, the accuracy both increased in the first 10 epochs. One possible reason could be that the adaptation performance also depends on the covariance state before DA. Since this phenomenon is more obvious on ISPRS dataset, the output logits of ISPRS is visualized in 2d images in figure 30 by utilizing tSNE[50] to decrease the dimension of the sample from 5 to 2. In each 5x320x320 output, 25 sample pixels are drawn for the visualization. Because of the complexity of tSNE, only 15000 samples are drawn from the output of each network. In figure 30, different classes are plotted with different colors. In the legend of each sub-figures, P, V and C refers to Potsdam, Vaihingen and $Coral_{log}$, respectively. The first

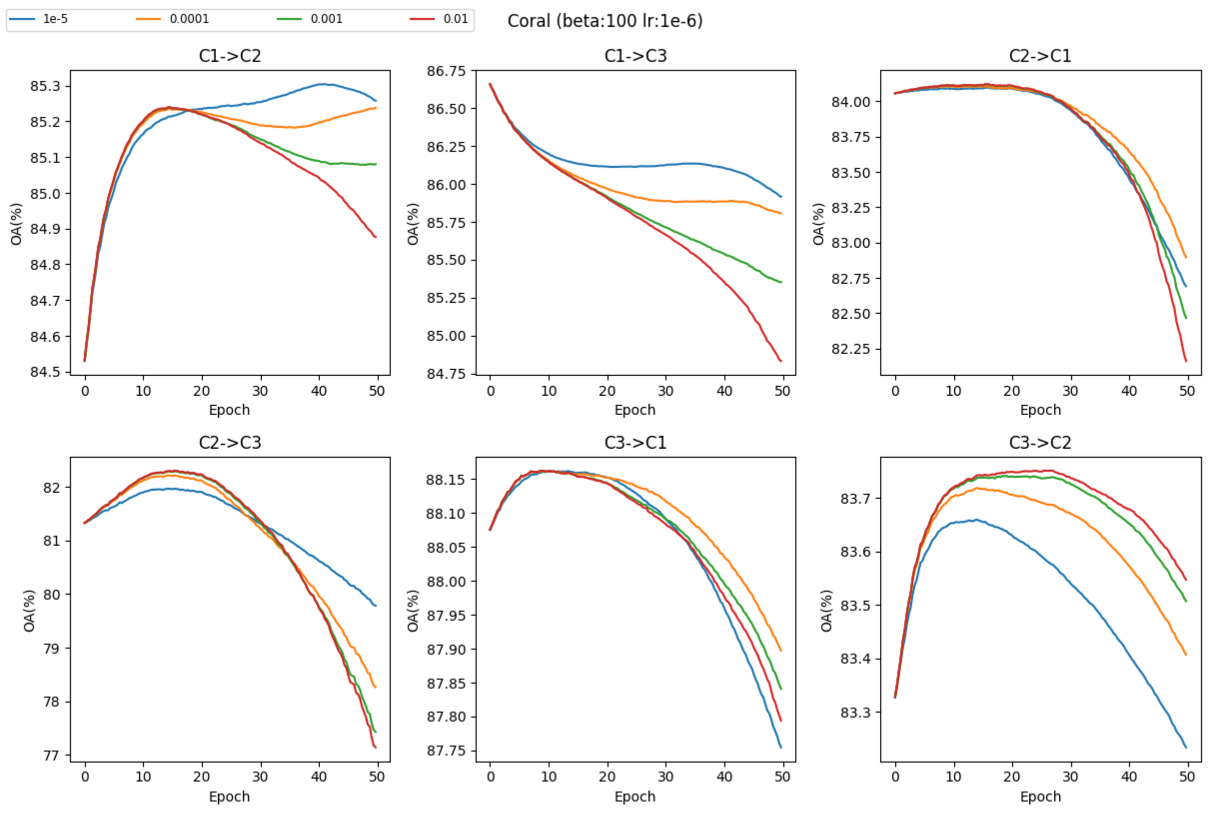


Figure 28: Overall accuracy of experiments with $Coral_{log}$ on 3City.

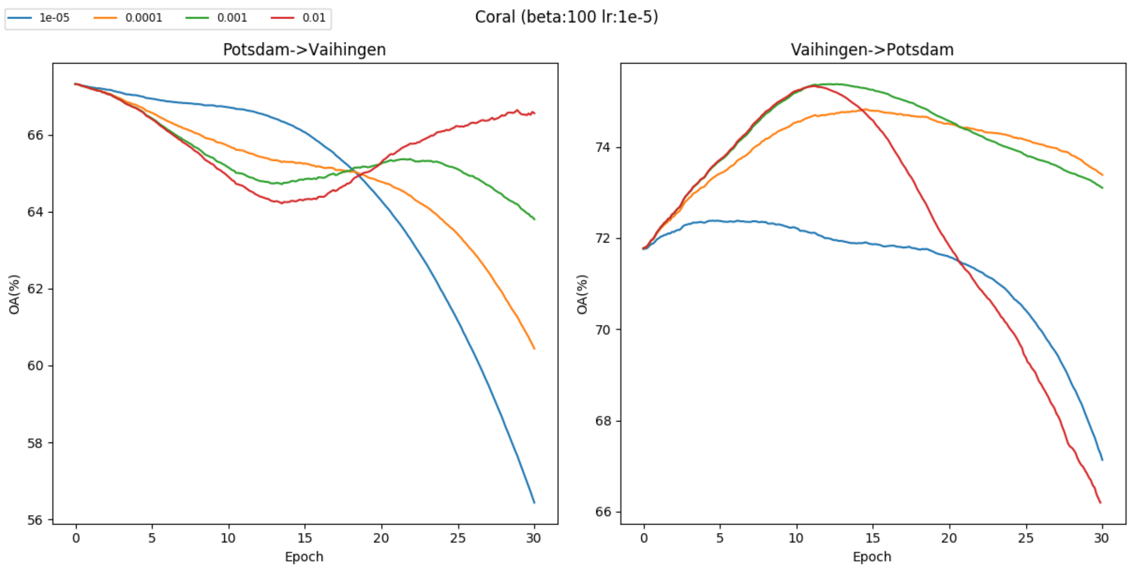


Figure 29: Overall accuracy of experiments with $Coral_{log}$ on ISPRS.

alphabet of each legend name indicates the source domain, the second the target domain. The numbers indicate classes. For example, PVC 0 indicates the pixel drawn from the output of the network adapted from Potsdam to Vaihingen by $Coral_{log}$ and PV 0 means the pixel is drawn from the output the unadapted source(Potsdam) network by passing target(Vaihingen) images to the network. All samples of PP, PV and PVC(or VV, VP,

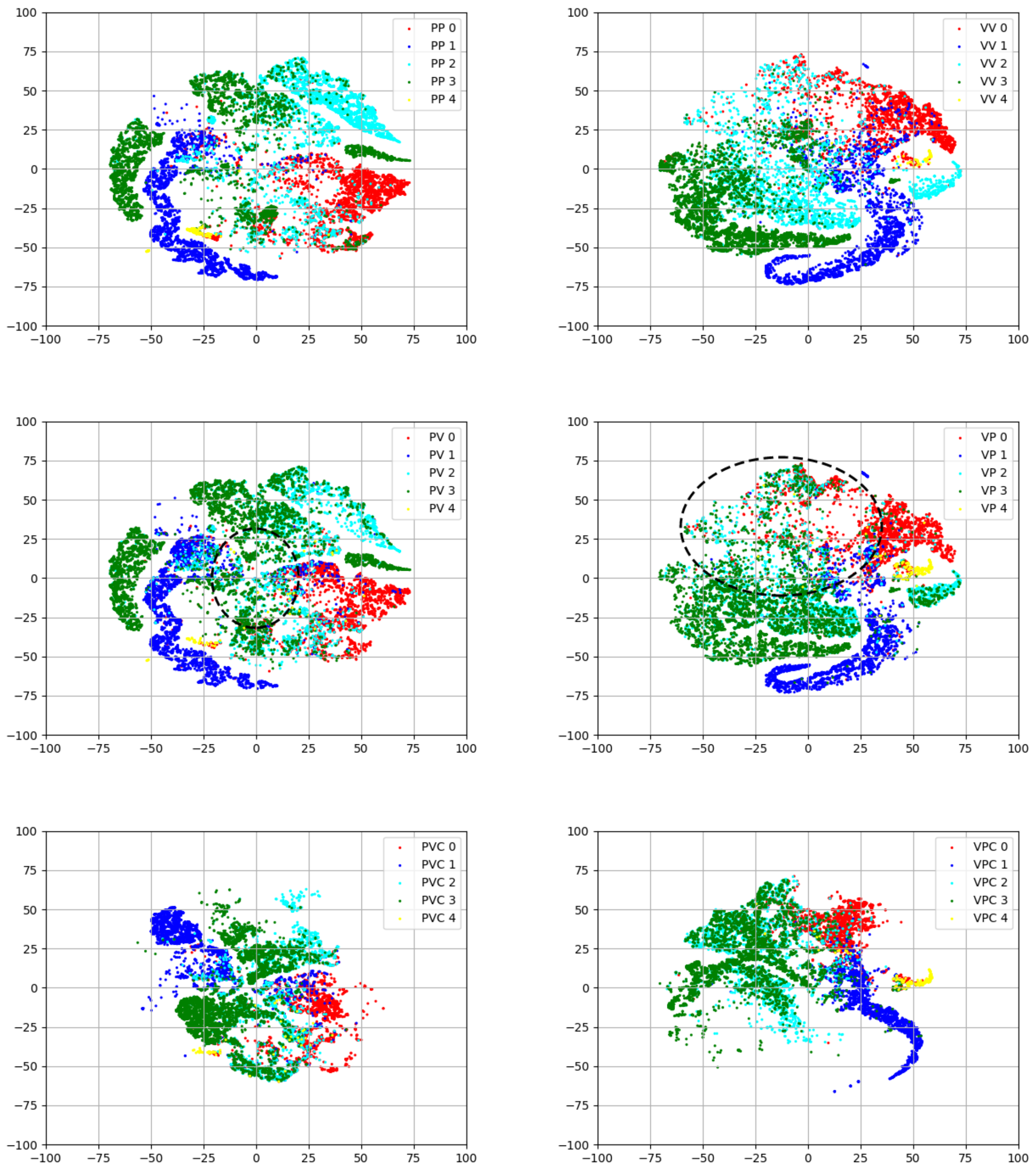


Figure 30: Visualization of the distribution of output logits of ISPRS data before DA and after DA with $Coral_{log}$. Class 0:impervious surface in red,1: building in blue, 2: tree in green, 3: low vegetation in cyan, 4: car in yellow

VPC) are embedded together into the same space, but plotted in 3 different sub-figures in one column.

As shown in the first 2 rows of figure 30, the result in the same domain(PP/VV) and the result for cross domains(PV/VP) are very similar when not regarding the classes. Namely, these samples are similarly distributed in this dimensionally decreased 2d space. Considering the classes, samples of "low vegetation(2)" and "tree(3)" are more likely to be overlapped in the subplots of the second row of the figure than in the subplots of the first row. Deep Coral aims to align covariances between two domains globally without regarding classes. The last row of figure 30 shows that the covariance itself is minimized after the alignment of logits with deep Coral. It pushes samples to the center and squashes the sparsely scattered areas which are marked with black dash line circles. In the left bottom subfigure, all samples are pushed exactly to the center, which makes all samples as well as all classes nearer to each other so that it is harder to differentiate different classes. As a result, the classification accuracy decreased in this case. But in the reverse direction(V$\rightarrow$P) in the right column, the sparse samples are mostly located on the upper part of the right-middle plot. The samples around this area are been pulled towards this area, which leads to the result in the right-bottom plot. In this plot, the points are more densely distributed without deteriorating the overlapping problem so that the accuracy of the pixel classification is increased.

As mentioned above, $Coral_{log}$ can still achieve small accuracy gain in most cases and the weights of the decoder are still not adapted after $Coral_{l7}$, there might still be a chance to optimize the accuracy with $Coral_{log}$ based on the adapted models of $Coral_{l7}$. Because the models are already adapted once, they are trained for 10 more epochs with $Coral_{log}$ and $\alpha = 0.01$ both for 3City and ISPRS. The result is shown in the last column of table 5. A small accuracy gain is achieved in most cases compared to the result of $Coral_{l7}$.

5.3.3. Comparison of Performances of Deep Coral and MMD

According to the figures shown in the previous section. Deep Coral performs better than MMD no matter in stability or in accuracy gain because there are more adaptation experiments with deep Coral that can enter into a stable state without too much OA decreasing in the last 30 epochs. In order to have a clearer comparison, the best average result of each method is chosen for the evaluation. The parameter configuration keeps the same for ISPRS and 3City since the best parameters for a specific unsupervised DA task in real application is not known. The results of DA are given in mean and standard deviation computed based on the OAs of certain epochs of DA experiments. The best results of $Coral_{l7}$ are computed by averaging OAs from epoch 30 to epoch 35 for 3 City and from epoch 9 to epoch 10 for ISPRS. The epochs chosen for evaluation are different for 3City and ISPRS because ISPRS has much more images than 3City so that each epoch of ISPRS has more iterations than 3 City. The parameter $\alpha = 200$ is chosen for the evaluation of this method. For adaptation with MMD_{l7}, the best average result occurs at $\alpha = 0.001$ and computed by averaging the OAs between epoch 15 and 20 for 3City and between epoch 9 and 10 for ISPRS. For $Coral_{log}$, OAs of epoch 15 to 20 and $\alpha = 0.01$ are taken for the evaluation. For $Coral_{l7,log}$, the means and standard deviations are computed by averaging OAs of epoch 8 to 10 both for 3City and ISPRS. The DA results are listed in the four columns on the right of

	tgt only	src only	$Coral_{l7}$	MMD_{l7}	$Coral_{log}$	$Coral_{l7,log}$
C1-C2	90.8	84.3	85.4±0.01	84.6±0.01	85.2±0.01	**85.6±0.00**
C1-C3	90.6	86.8	**87.4±0.00**	87.3±0.00	86.0±0.03	87.1±0.02
C2-C1	91.1	84.0	86.4±0.05	84.8±0.00	84.1±0.00	**86.5±0.00**
C2-C3	90.6	81.3	84.6±0.03	82.3±0.05	82.3±0.02	**84.9±0.04**
C3-C1	91.1	88.0	87.9±0.02	88.2±0.02	88.2±0.00	**88.3±0.00**
C3-C2	90.8	83.3	83.9±0.01	83.3±0.02	83.8±0.00	**84.2±0.01**
P-V	87.3	67.4	**76.8±0.32**	74.9±-0.11	65.2±0.17	73.6±0.30
V-P	89.0	71.6	76.7±0.03	73.7±0.10	75.0±0.13	**77.8±0.02**
ave. gain 3city	-	-	1.3	0.5	0.3	**1.5**
max. gain 3city	-	-	3.3	1.0	1.0	**3.6**
ave. OA 3city	90.8	84.6	85.9	85.1	84.9	**86.1**
ave. gain isprs	-	-	**7.3**	4.8	0.6	6.2
max. gain isprs	-	-	**9.5**	7.6	3.5	6.2
ave. OA isprs	88.1	69.5	**76.8**	74.3	70.1	75.7

Table 5: Overall Accuracy of Adaptation on 3City and ISPRS(%).

table 5. The first column of table 5 indicates the source and target city. For example, C1-C2 means adaptation from domain C1 to domain C2. The second column gives the accuracy of target networks tested on images of the target city and the third column shows the accuracy of the source network tested on the images of the target city. The last 6 rows summarizes the average OA gain, the maximum OA gain and the average OA after DA for 3City and ISPRS separately based on the results in the top part of the table. All results in the table are overall accuracy in percentage.

If comparing the results in the vertical direction of table 5, it is shown that domains with larger accuracy drop can generally obtain more OA gain. This phenomenon is very evident when comparing OA drops in 3City dataset and ISPRS dataset. Most positive transfers on ISPRS have more than 5% OA gain while most adaptations in 3City are less than 2%. Moreover, the standard deviation of the results on ISPRS is in general larger than that on 3City. This might be caused by the resolution difference between the images of 3City and ISPRS. ISPSR has a higher resolution which is 8cm and 3City has a resolution of 20cm.

This leads to a biased sampling problem which means that it is very likely that there is only one class or two classes in one sample image of ISPRS. In contrast, sample images in 3City have more balanced class label distributions.

In horizontal direction of table 5, the performance of different DA methods can be compared. Negative transfer(marked red) only occurred in deep Coral alignment. MMD_{l7} has no negative transfer, but its overall performance is worse than $Coral_{l7}$, however, slightly better than $Coral_{log}$. $Coral_{l7}$, in contrast, has one small negative transfer but the results of all other adaptations are better than MMD_{l7} and $Coral_{log}$. It is not guaranteed that all transfers with deep Coral on a single layer are positive, no matter on layer 7 of the encoder or the last layer of the decoder. However, this can be achieved by adapting these two layers jointly. The result of $Coral_{l7,log}$ shown in the last column of table 5 has no negative transfers and most of them are marked bold, which means that they are the best result across the corresponding row. Overall, $Coral_{l7,log}$ has the best performance. The average OA gain of $Coral_{l7,log}$ on 3City and ISPRS is 1.5% and 6.2% respectively. The corresponding maximum OA gain is 3.6% on 3 City and 6.2% on ISPRS.

It is possible that different network architectures or configurations lead to different accuracy drops before DA, so it is unfair to compare different DA methods only based on accuracy gain. So the results in the first 8 rows are averaged separately for 3City and ISPRS in the 10th and the 13th row to compare the final classification performance after DA. 3City has an average OA of 84.6% before adaptation which is similar to the result in the baseline work[74]. However, a better DA result is obtained in this work, which is 86.1% vs. 85.4% in [74]. Besides, the best average result for ISPRS is achieved by $Coral_{l7}$, it increased from 69.5% to 76.8%.

5.3.4. Appearance Adaptation with Cycle GAN on input images

Some examples of the results of image translation are shown in figure 31. For all image pairs, the structures, geometry shapes are consistent. Only the appearance, in other words, the color and the texture, are changed. There is still some artificial effect in all translations, especially between C2 and C3. The shadow in C3 is very dark and some may even contain the high value in the red channel so that the texture of the shadow is very similar to that of some areas of the class tree in C2. This makes it harder for Cycle GAN to discriminate these two classes in some specific pixel patches which leads to unfavorable translations just as the four images shown on the left bottom side of figure 31. On fake images of C2, some trees are rendered black, and in the fake domain of C3, some roof pixels are rendered with textures of the tree. In the translation of images between Potsdam and Vaihingen, the texture and color of trees are well translated, the structure of the tree crown is kept. As for BMW data, the appearance of Combox, as well as the container, is good translated from the synthetic

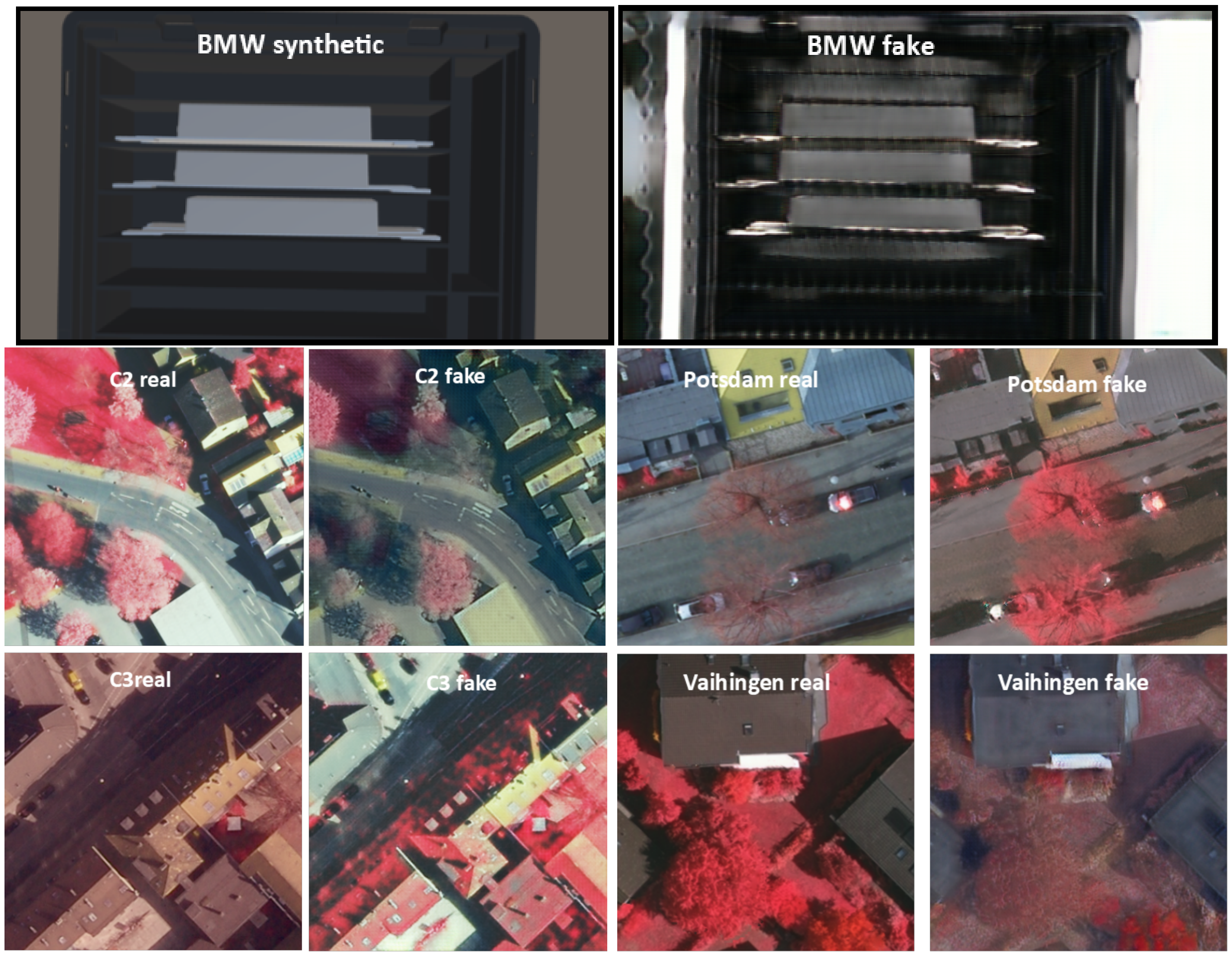


Figure 31: Appearance adaptation result

domain to the real domain.

Table 6: Result of Appearance Adaptation on BMW dataset(%)

	real	synth→ *real*	synth+fake→ *real*
OA	99.7	88.0	91.7
mIOU	98.8	61.4	82.0
combox	97.4	23.2	78.9
container	99.6	84.6	90.2
background	99.5	76.4	76.9

The performance of image translation for DA on BMW data is shown in table 6. The accuracy is dramatically increased no matter for OA or mIOU. OA only increased by about 4%, but because of the huge accuracy gain of class ”combox”, the mean IOU increased by about 20%. This is beneficial for detecting comboxes in real applications of object detection. Two prediction examples are shown in figure 32, The left column shows the ground truth of these two examples. The middle column shows the prediction result of the model trained on synthetic images and tested on real images, and on the right, the labels of real images are predicted by the model trained on the extended BMW dataset(synthetic images and fake images). It is obvious that the predictions on the right are much more favorable than the middle ones.

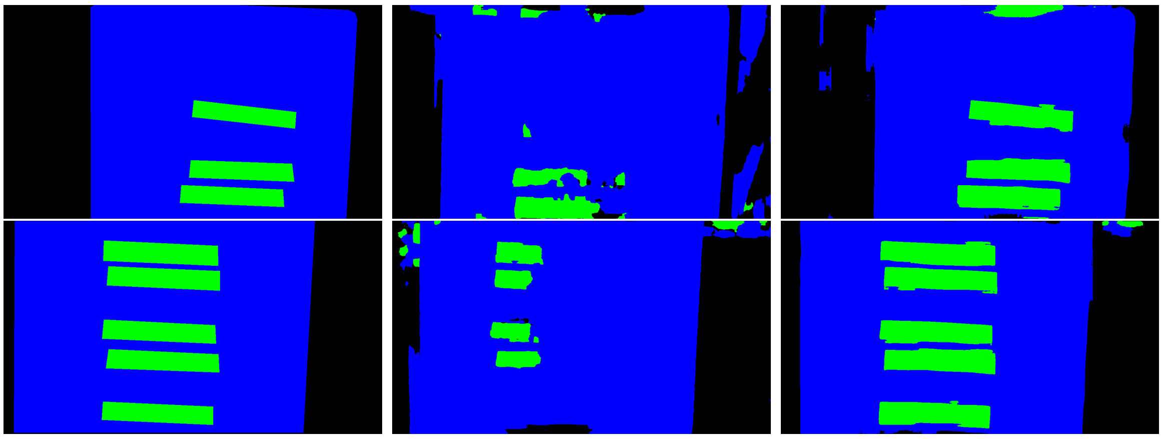


Figure 32: Ground truth and predictions with and without data augmentation by image translation(left: ground truth, middle: predictions of model synth to real, right: predictions of model synth+fake to real)

Table 7: Result of Appearance Adaptation between C2 and C3(%)

	$C2_{real} \rightarrow C3$	$C2_{real} + C2_{fake} \rightarrow C3$	$C3_{real} \rightarrow C2$	$C3_{real} + C3_{fake} \rightarrow C2$
OA	81.3	79.5	83.3	83.9
mIOU	62.8	61.3	66.9	67.2
tree	48.7	56.1	54.1	51.5
building	63.5	54.9	72.7	75.4
ground	75.5	72.9	74.0	74.8

Table 8: Result of Appearance Adaptation on ISPRS dataset(%)

	$P_{real} \rightarrow V$	$P_{real} + P_{fake} \rightarrow V$	$V_{real} \rightarrow P$	$V_{real} + V_{fake} \rightarrow P$
OA	67.4	65.1	71.6	77.3
mIOU	45.8	46.3	54.4	60.5
ground	51.5	50.8	60.6	70.3
building	41.9	39.6	71.6	78.2
low vegetation	51.3	46.9	42.6	48.8
tree	62.8	56.9	34.7	38.2
car	21.4	37.4	62.2	66.8

In contrast, image translation seems not so helpful for DAs in RS. The results of RS are shown in table 7 and 8. For adaptations between C2 and C3 and between Potsdam and

Vaihingen, the accuracy gain is observed only in one direction. The small accuracy gain in the experiment C3→ C2 is highly dependent on the class "building" because this class has more samples in the target domain C2 than in the source domain C3. However, the IOU of class "tree" of adaptation from Vaihingen to Potsdam is increased from 34.7% to 38.2% despite the target city Potsdam has less proportion of trees than the source city. Besides, the proportion of class "building" is similar in both source and target dataset in adaptation Vaihingen→Potsdam, the IOU of this class increased more than 6%. For other classes, since the label density of the target domain is higher than the source domain, it is hard to tell how much accuracy gain is obtained by image translation.

As a conclusion, appearance translation can be very helpful for narrowing the domain gap for applications that have high appearance discrepancy like the difference between synthetic and realistic images. They are only different in color and texture but have similar shapes and sizes for the same type of objects or similar label distributions. On the other hand, DA for aerial images is much more complex because most similarities mentioned for the synthetic-real case are disappeared. The performance of appearance translation is very case-dependent. Mostly, appearance adaptation is not very helpful for RS, especially when the adaptation task is hard, which means that the source domain and the target domain are very similar(i.e domains in 3City dataset). There is no guarantee that image translation can mitigate the domain gap between source city and target city for RS aerial images. The reason for the good performance in adaptation $V_{real} + V_{fake} \rightarrow P$ might be that images of the target domain Potsdam has more discriminative appearance features, which are crucial for the correct prediction, that can be applied and drawn on the images of the source domain Vaihingen. For example, the buildings in Potsdam mostly have yellow or gray roofs while roofs in the other city are more diverse. Once these simple but crucial patterns of target domain are learned and rendered on the source domain images, it is very likely that the model will learn these features of the target domain for the final prediction. But it is not guaranteed that for all applications, these useful features can be captured by Cycle GAN.

6. Conclusion

In this paper, semantic segmentation and deep DA is introduced, and then the related works of these two researching directions are discussed following by a taxonomy of deep DA for semantic segmentation which is summarized in figure 1. The network used in this work is a modified version of the network in [74] by decreasing input image size and removing 2 convolutional layers as well as 2 transposed convolutional layers. Besides, skip-connections are also removed. This network used for remote sensing datasets is called Ynet(figure 8). For the BMW dataset, one encoder of Ynet is removed. The details of the network are discussed in chapter Semantic Segmentation. With this simplified model, the classification accuracy in the source domain only dropped a little comparing to the result in [74]. However, the training time is shortened a lot. Moreover, the average accuracy in the target domain before DA is similar to the result of [74].

For deep DA, there are rarely works using Deep Coral and MMD for DA of semantic segmentation, so experiments about DA with deep Coral and MMD are conducted in this work. Besides, many works based on Cityskypes dataset use Cycle GAN to decrease the domain discrepancy. So appearance adaptation using Cycle GAN is tested in this thesis to check if Cycle GAN can also have good performance on remote sensing data. For deep Coral, 3 experiments are performed. The first one aligns activations of layer 7 of encoders. The second one aligns the output logits of the decoder. The last one aligns output logits after the alignment of layer 7. In comparison, matching layer 7 performs better than matching logits, and the best performance is observed in the last experiment. For MMD, only one experiment is performed by aligning activations of layer 7, because align the last output layer is too computationally expensive. This method can get small but stable accuracy gain. It performs worse than deep Coral on layer 7 but better than on logits layer. Overall, the best result is obtained in the third experiment of deep Coral. The DA with deep Coral by aligning layer 7 and logits jointly performs slightly better than the adversarial adaptation in the baseline work[74] on 3City dataset. The average overall accuracy in this work is 86.08%, and in [74] it is 85.5%. As a conclusion, deep DA for semantic segmentation of RS images with deep Coral can obtain a better result than the currently very popular adversarial adaptation and the training process of deep Coral is also easier than adversarial training since it is not guaranteed that GAN can reach the nash equilibrium. Moreover, the discriminator also needs more additional computational power than deep Coral. At last, experiments using Cycle GAN are conducted both on the BMW dataset and selected RS datasets. The images of the BMW dataset only contain car components, containers, and background. The experiment results show that the accuracy of the BMW dataset is dramatically increased after the appearance adaptation with Cycle GAN, but it is not the case for the RS datasets. In the 4 experiments for RS data, the accuracies of two experiments have dropped, one increased a little bit, one increased 6%. In conclusion, it is easier to obtain accuracy gain between the synthetic and the real domain like the adaptation on the BMW dataset. In contrast, there is no guarantee that Cycle GAN can help to narrow the domain discrepancy for RS data.

The contributions of this work can be summarized into 3 points. First, Ynet, a simplified version of the network in [74], is designed. The average accuracy of Ynet on the source domain only dropped slightly but it is nearly the same when testing on the target domain. Second, it is proven by the experiment result that the statistic-based method MMD and deep Coral can both help to decrease the domain discrepancy. Deep Coral performs better than MMD when they are applied to the features of medium layers of the encoder. At last, the experiment results show that there is no guarantee that Cycle GAN can mitigate the domain discrepancy for RS datasets that only contains real images.

Since all experiments are performed on the same network structure, future research should do more experiments with different network architectures to verify if MMD or deep Coral can generally perform well on all models. An in stead of only aligning layer 7 and logits, more different layers can be aligned. Besides, Cycle GAN in this work is only used to translate images from source to target domain, more experiments can be done by translating target images to source domain and then predictions can be performed by source network on the translated target images. Moreover, category-wise feature alignment might further decrease the domain discrepancy. It can be achieved by ensemble of networks or pseudo labels. As for ensemble, each class can have it's own classifier which can have a simpler network architecture than a multi-class classifier. The alignment of source and target features can be done on the classifier of specific classes. For Pseudo labels, the most confident predictions can be chosen for the category-wise feature alignment.

References

[1] *2D Semantic Labeling Contest, ISPRS WG III/4*. URL: http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html.

[2] Martin Arjovsky et al. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[3] Vijay Badrinarayanan et al. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *CoRR* abs/1511.00561 (2015). arXiv: 1511.00561. URL: http://arxiv.org/abs/1511.00561.

[4] Bilel Benjdira et al. "Unsupervised Domain Adaptation Using Generative Adversarial Networks for Semantic Segmentation of Aerial Images". In: *Remote Sensing* 11.11 (June 2019), p. 1369. ISSN: 2072-4292. DOI: 10.3390/rs11111369. URL: http://dx.doi.org/10.3390/rs11111369.

[5] Michele Cavaioni. *DeepLearning series: Convolutional Neural Networks*. 2018. URL: https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524 (visited on 02/24/2018).

[6] Wei-Lun Chang et al. *All about Structure: Adapting Structural Information across Domains for Boosting Semantic Segmentation*. 2019. arXiv: 1903.12212 [cs.CV].

[7] Yi-Hsin Chen et al. *No More Discrimination: Cross City Adaptation of Road Scene Segmenters*. 2017. arXiv: 1704.08509 [cs.CV].

[8] Liang-Chieh Chen et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *CoRR* abs/1606.00915 (2016). arXiv: 1606.00915. URL: http://arxiv.org/abs/1606.00915.

[9] Liang-Chieh Chen et al. "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs". In: (2014). arXiv: 1412.7062 [cs.CV].

[10] Yuhua Chen et al. *Learning Semantic Segmentation from Synthetic Data: A Geometrically Guided Input-Output Adaptation Approach*. 2018. arXiv: 1812.05040 [cs.CV].

[11] Yun-Chun Chen et al. "CrDoCo: Pixel-Level Domain Transfer With Cross-Domain Consistency". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[12] Jaehoon Choi et al. *Self-Ensembling with GAN-based Data Augmentation for Domain Adaptation in Semantic Segmentation*. 2019. arXiv: 1909.00589 [cs.CV].

[13] Djork-Arné Clevert et al. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. 2015. arXiv: 1511.07289 [cs.LG].

[14] *Complete Guide of Activation Functions*. 2019. URL: https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044 (visited on 06/12/2019).

[15] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[16] Gabriela Csurka. “Domain Adaptation for Visual Applications: A Comprehensive Survey”. In: *CoRR* abs/1702.05374 (2017). arXiv: 1702.05374. URL: http://arxiv.org/abs/1702.05374.

[17] N. Dalal et al. “Histograms of oriented gradients for human detection”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. June 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[18] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*. 2009.

[19] Liang Du et al. “SSF-DAN: Separated Semantic Feature Based Domain Adaptation Network for Semantic Segmentation”. In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2019.

[20] Vincent Dumoulin et al. *A guide to convolution arithmetic for deep learning*. 2016. arXiv: 1603.07285 [stat.ML].

[21] M. Everingham et al. “The Pascal Visual Object Classes Challenge: A Retrospective”. In: *International Journal of Computer Vision* 111.1 (Jan. 2015), pp. 98–136.

[22] Theodoros Evgeniou et al. “Support Vector Machines: Theory and Applications”. In: vol. 2049. Jan. 2001, pp. 249–257. DOI: 10.1007/3-540-44673-7_12.

[23] Yaroslav Ganin et al. *Domain-Adversarial Training of Neural Networks*. 2015. arXiv: 1505.07818 [stat.ML].

[24] Andreas Geiger et al. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[25] Rui Gong et al. *DLOW: Domain Flow for Adaptation and Generalization*. 2018. arXiv: 1812.05418 [cs.CV].

[26] Ian Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[27] Arthur Gretton et al. “A Kernel Two-sample Test”. In: *J. Mach. Learn. Res.* 13 (Mar. 2012), pp. 723–773. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2188385.2188410.

[28] Arthur Gretton et al. “Optimal kernel choice for large-scale two-sample tests”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1205–1213. URL: http://papers.nips.cc/paper/4727-optimal-kernel-choice-for-large-scale-two-sample-tests.pdf.

[29] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[30] Tin Kam Ho. “Random Decision Forests”. In: *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*. ICDAR ’95. USA: IEEE Computer Society, 1995, p. 278. ISBN: 0818671289.

[31] Judy Hoffman et al. *CyCADA: Cycle-Consistent Adversarial Domain Adaptation*. 2017. arXiv: 1711.03213 [cs.CV].

[32] Judy Hoffman et al. *FCNs in the Wild: Pixel-level Adversarial and Constraint-based Adaptation*. 2016. arXiv: 1612.02649 [cs.CV].

[33] Weixiang Hong et al. “Conditional Generative Adversarial Network for Structured Domain Adaptation”. In: June 2018, pp. 1335–1344. DOI: 10.1109/CVPR.2018.00145.

[34] Sergey Ioffe et al. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[35] Javed Iqbal et al. *MLSL: Multi-Level Self-Supervised Learning for Domain Adaptation with Spatially Independent and Semantically Consistent Labeling*. 2019. arXiv: 1909.13776 [cs.CV].

[36] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. 2016. arXiv: 1611.07004 [cs.CV].

[37] Jeremy Jordan. *An overview of semantic image segmentation*. 2018. URL: https://www.jeremyjordan.me/semantic-segmentation/ (visited on 03/18/2018).

[38] Alex Krizhevsky et al. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. 2012, pp. 1106–1114. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

[39] Chun-Liang Li et al. *MMD GAN: Towards Deeper Understanding of Moment Matching Network*. 2017. arXiv: 1705.08584 [cs.LG].

[40] Yujia Li et al. *Generative Moment Matching Networks*. 2015. arXiv: 1502.02761 [cs.LG].

[41] Yunsheng Li et al. *Bidirectional Learning for Domain Adaptation of Semantic Segmentation*. 2019. arXiv: 1904.10620 [cs.CV].

[42] Qing Lian et al. *Constructing Self-motivated Pyramid Curriculums for Cross-Domain Semantic Segmentation: A Non-Adversarial Approach*. 2019. arXiv: 1908.09547 [cs.CV].

[43] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. arXiv: 1405.0312 [cs.CV].

[44] Guilin Liu et al. *Partial Convolution based Padding*. 2018. arXiv: 1811.11718 [cs.CV].

[45] Mingsheng Long et al. *Learning Transferable Features with Deep Adaptation Networks*. 2015. arXiv: 1502.02791 [cs.LG].

[46] Mingsheng Long et al. “Learning Transferable Features with Deep Adaptation Networks”. In: *CoRR* abs/1502.02791 (2015). arXiv: 1502.02791. URL: http://arxiv.org/abs/1502.02791.

[47] David Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60 (Nov. 2004), pp. 91–. DOI: 10.1023/B:VISI.0000029664.99615.94.

[48] Yawei Luo et al. *Significance-aware Information Bottleneck for Domain Adaptive Semantic Segmentation*. 2019. arXiv: 1904.00876 [cs.CV].

[49] Yawei Luo et al. *Taking A Closer Look at Domain Shift: Category-level Adversaries for Semantics Consistent Domain Adaptation*. 2018. arXiv: 1809.09478 [cs.CV].

[50] Laurens van der Maaten et al. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605. URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[51] Xudong Mao et al. *Least Squares Generative Adversarial Networks*. 2016. arXiv: 1611.04076 [cs.CV].

[52] Bjoern Menze et al. “The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)”. In: *IEEE Transactions on Medical Imaging* 34.10 (Oct. 2014), pp. 1993–2024. DOI: 10.1109/TMI.2014.2377694. URL: https://hal.inria.fr/hal-00935640.

[53] Umberto Michieli et al. *Adversarial Learning and Self-Teaching Techniques for Domain Adaptation in Semantic Segmentation*. 2019. arXiv: 1909.00781 [cs.CV].

[54] Zak Murez et al. *Image to Image Translation for Domain Adaptation*. 2017. arXiv: 1712.00479 [cs.CV].

[55] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013. ISBN: 9780262018029 0262018020. URL: https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr_1_2?ie=UTF8&qid=1336857747&sr=8-2.

[56] Hyeonwoo Noh et al. “Learning Deconvolution Network for Semantic Segmentation”. In: *CoRR* abs/1505.04366 (2015). arXiv: 1505.04366. URL: http://arxiv.org/abs/1505.04366.

[57] Annegreet van Opbroek et al. “Transfer Learning Improves Supervised Image Segmentation Across Imaging Protocols”. In: *IEEE Transactions on Medical Imaging* 34 (2015), pp. 1018–1030.

[58] Sinno Pan et al. “A Survey on Transfer Learning”. In: *Knowledge and Data Engineering, IEEE Transactions on* 22 (Nov. 2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

[59] *Reproducing kernel Hilbert space*. URL: https://en.wikipedia.org/wiki/Reproducing_kernel_Hilbert_space.

[60] Rob Romijnders et al. *A Domain Agnostic Normalization Layer for Unsupervised Adversarial Domain Adaptation*. 2018. arXiv: 1809.05298 [cs.CV].

[61] Olaf Ronneberger et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[62] Claude Elwood Shannon. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27.3 (July 1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x. URL: https://ieeexplore.ieee.org/document/6773024/.

[63] Evan Shelhamer et al. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.4 (Apr. 2017), pp. 640–651. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2572683. URL: https://doi.org/10.1109/TPAMI.2016.2572683.

[64] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[65] Baochen Sun et al. "Deep CORAL: Correlation Alignment for Deep Domain Adaptation". In: *CoRR* abs/1607.01719 (2016). arXiv: 1607.01719. URL: http://arxiv.org/abs/1607.01719.

[66] Baochen Sun et al. "Return of Frustratingly Easy Domain Adaptation". In: *CoRR* abs/1511.05547 (2015). arXiv: 1511.05547. URL: http://arxiv.org/abs/1511.05547.

[67] Jonathan Tompson et al. *Efficient Object Localization Using Convolutional Networks*. 2014. arXiv: 1411.4280 [cs.CV].

[68] Yi-Hsuan Tsai et al. *Learning to Adapt Structured Output Space for Semantic Segmentation*. 2018. arXiv: 1802.10349 [cs.CV].

[69] *Two-sample hypothesis testing*. URL: https://en.wikipedia.org/wiki/Two-sample_hypothesis_testing.

[70] Eric Tzeng et al. "Adversarial Discriminative Domain Adaptation". In: *CoRR* abs/1702.05464 (2017). arXiv: 1702.05464. URL: http://arxiv.org/abs/1702.05464.

[71] Eric Tzeng et al. "Deep Domain Confusion: Maximizing for Domain Invariance". In: (2014). arXiv: 1412.3474 [cs.CV].

[72] *Unit sphere*. URL: https://en.wikipedia.org/wiki/Unit_sphere.

[73] Tuan-Hung Vu et al. *DADA: Depth-aware Domain Adaptation in Semantic Segmentation*. 2019. arXiv: 1904.01886 [cs.CV].

[74] Dennis Wittich et al. "ADVERSARIAL DOMAIN ADAPTATION FOR THE CLASSIFICATION OF AERIAL IMAGES AND HEIGHT DATA USING CONVOLUTIONAL NEURAL NETWORKS". In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-2/W7 (Sept. 2019), pp. 197–204. DOI: 10.5194/isprs-annals-IV-2-W7-197-2019.

[75] C. Yang et al. "TOWARDS BETTER CLASSIFICATION OF LAND COVER AND LAND USE BASED ON CONVOLUTIONAL NEURAL NETWORKS". In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W13 (2019), pp. 139–146. DOI: 10.5194/isprs-archives-XLII-2-W13-139-2019. URL: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2-W13/139/2019/.

[76] Fisher Yu et al. *Dilated Residual Networks*. 2017. arXiv: 1705.09914 [cs.CV].

[77] Fangneng Zhan et al. *GA-DAN: Geometry-Aware Domain Adaptation Network for Scene Text Detection and Recognition*. 2019. arXiv: 1907.09653 [cs.CV].

[78] Yang Zhang et al. "Curriculum Domain Adaptation for Semantic Segmentation of Urban Scenes". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017). DOI: 10.1109/iccv.2017.223. URL: http://dx.doi.org/10.1109/ICCV.2017.223.

[79] Hengshuang Zhao et al. *ICNet for Real-Time Semantic Segmentation on High-Resolution Images*. 2017. arXiv: 1704.08545 [cs.CV].

[80] Hengshuang Zhao et al. "Pyramid Scene Parsing Network". In: (2016). arXiv: 1612.01105 [cs.CV].

[81] Shuai Zheng et al. "Conditional Random Fields as Recurrent Neural Networks". In: *CoRR* abs/1502.03240 (2015). arXiv: 1502.03240. URL: http://arxiv.org/abs/1502.03240.

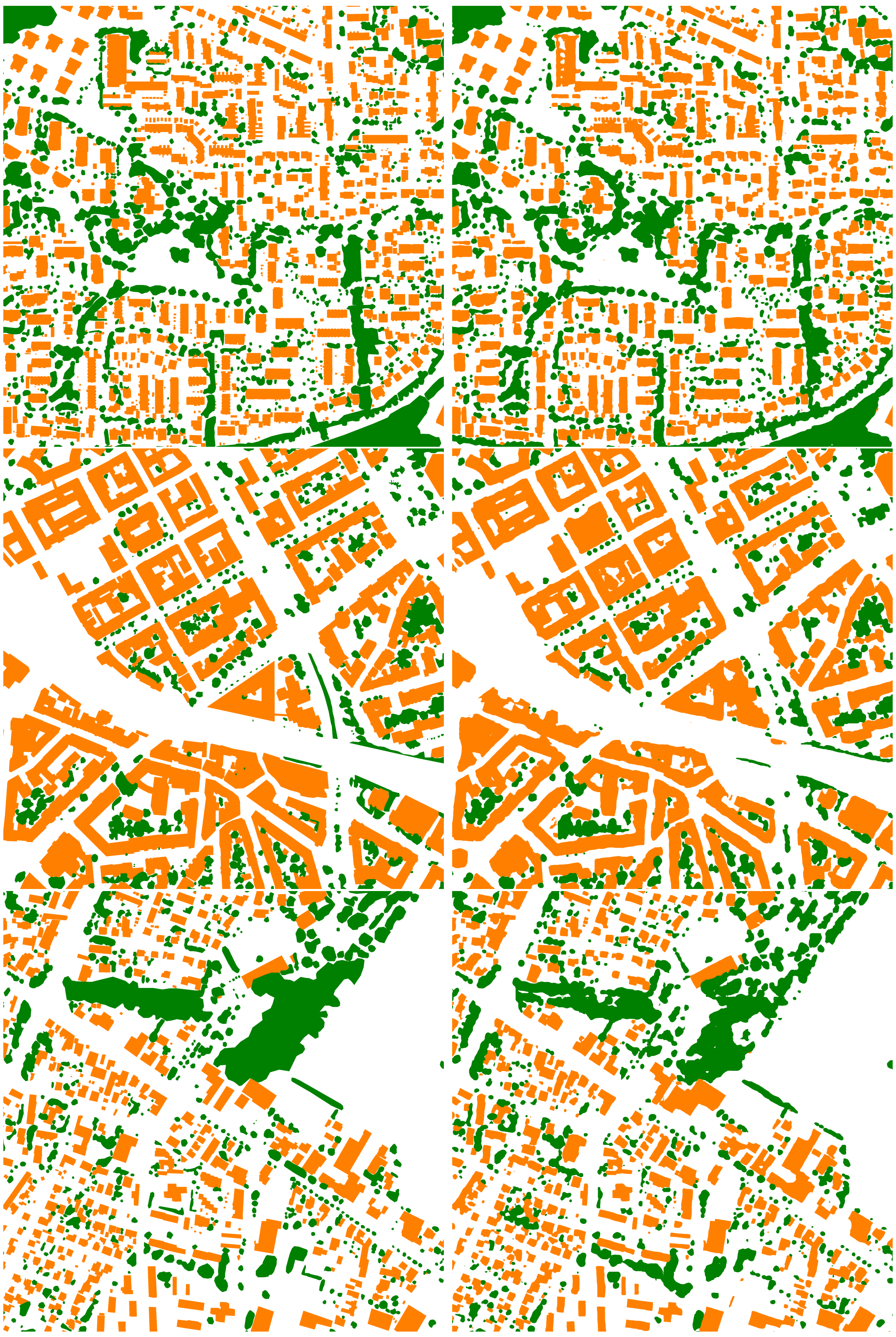
[82] Jun-Yan Zhu et al. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. 2017. arXiv: 1703.10593 [cs.CV].

[83] Yang Zou et al. *Domain Adaptation for Semantic Segmentation via Class-Balanced Self-Training*. 2018. arXiv: 1810.07911 [cs.CV].

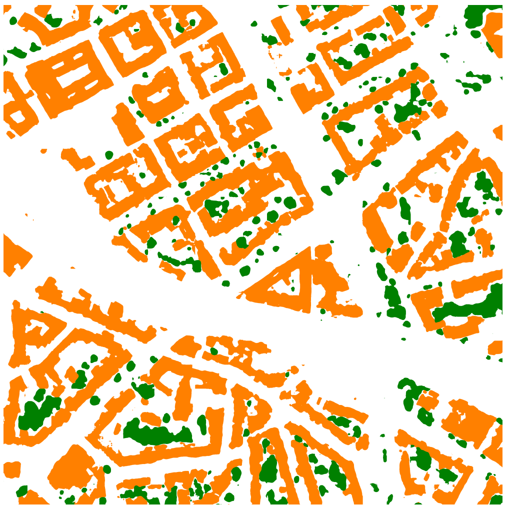
A. Predictions on 3City data set

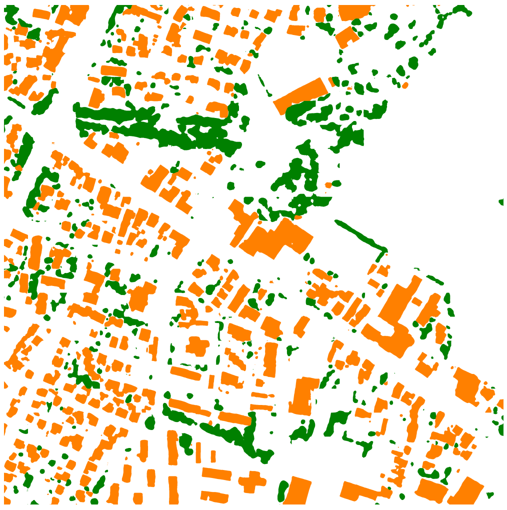
A.1. Predictions in the same domain

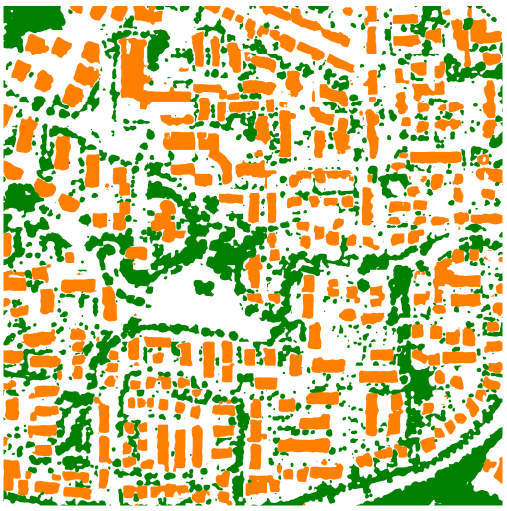
Figure 33: Left: ground truth, right: prediction, top to bottom: C1, C2, C3

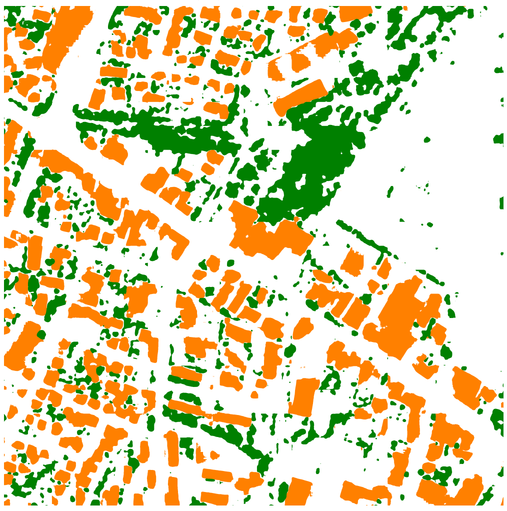


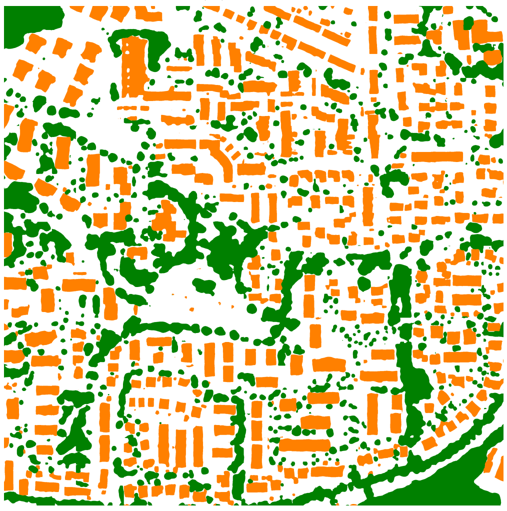
A.2. Predictions on the target domain without DA

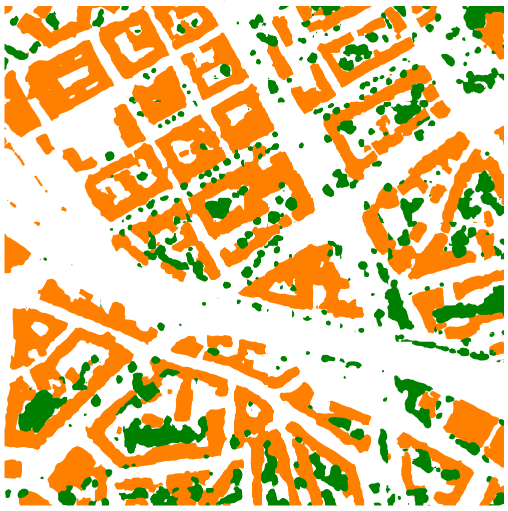

(a) C1-C2


(b) C1-C3

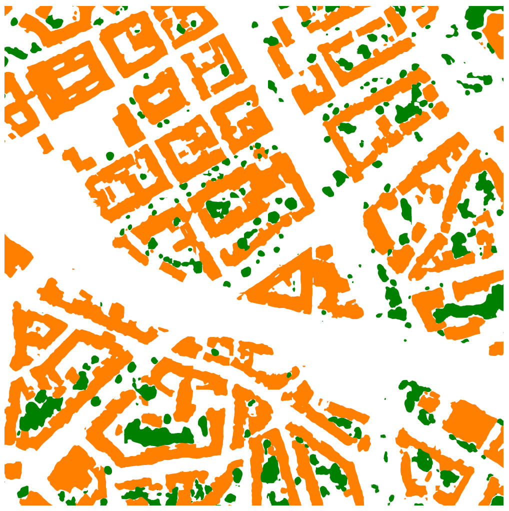

(c) C2-C1

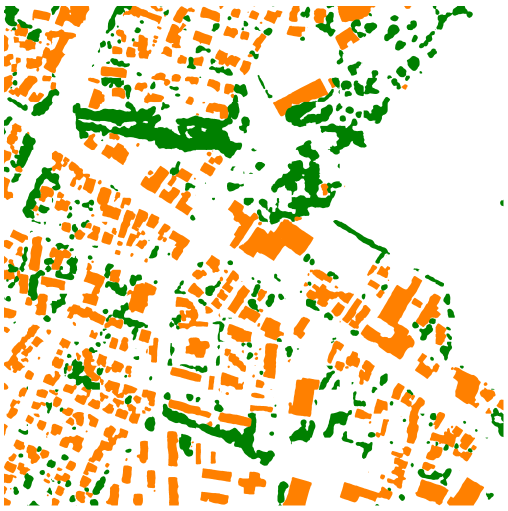

(d) C2-C3

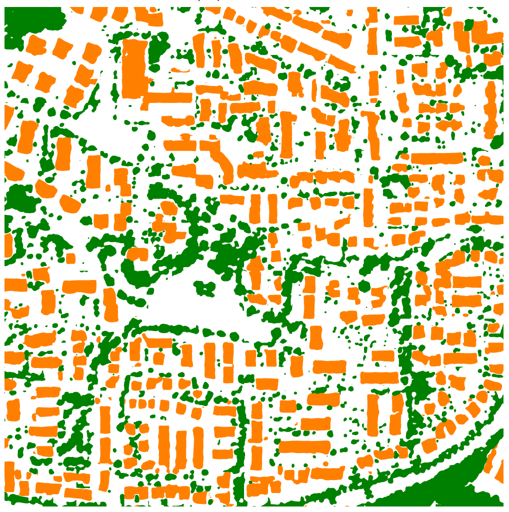

(e) C3-C1

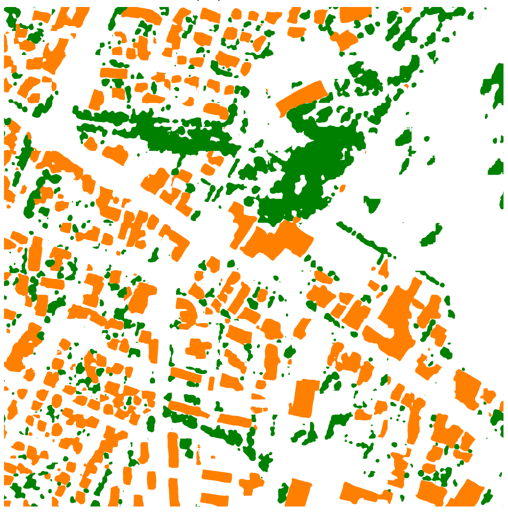

(f) C3-C2

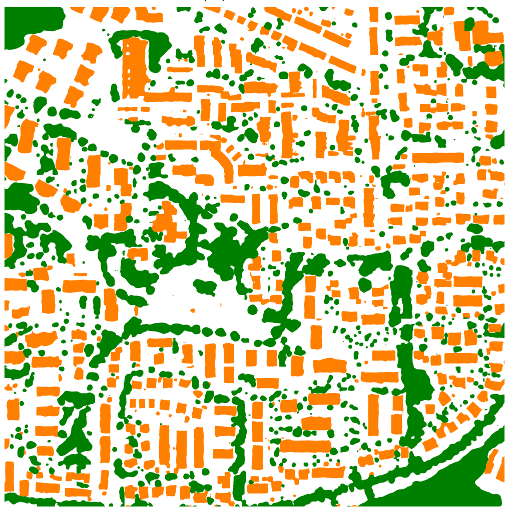
A.3. Predictions after deep Coral adaptation on l7

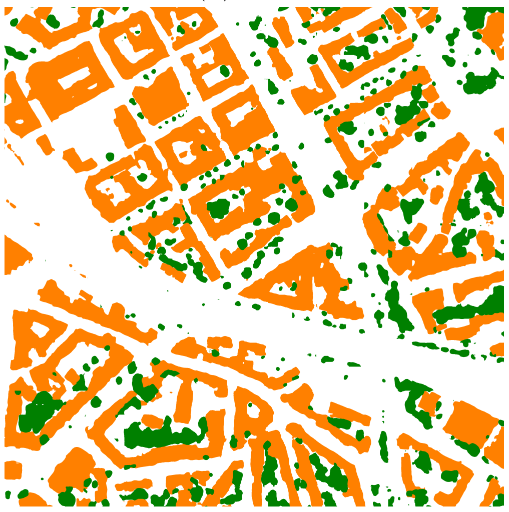

(a) C1-C2


(b) C1-C3

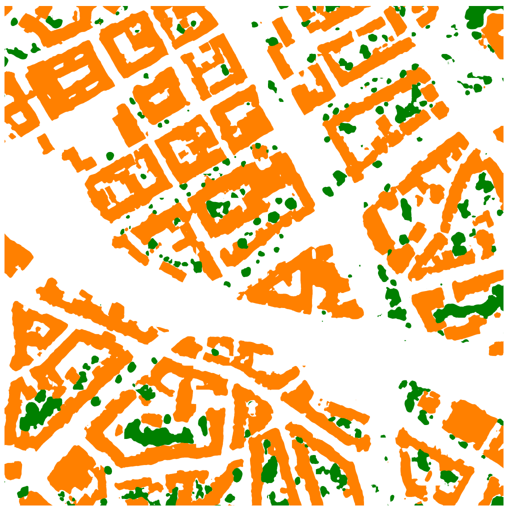

(c) C2-C1

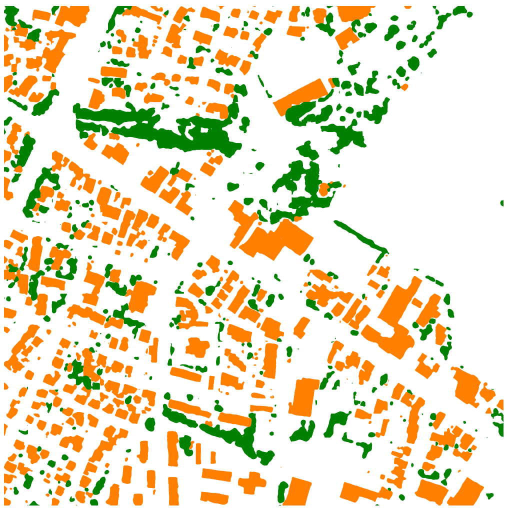

(d) C2-C3

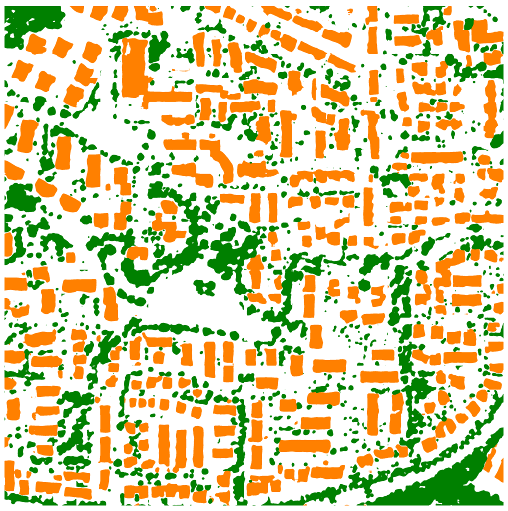

(e) C3-C1

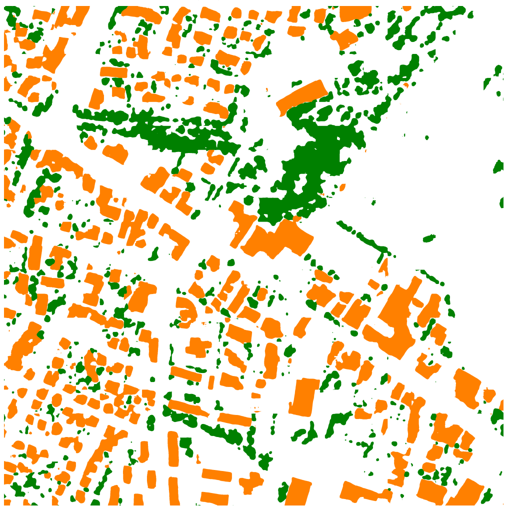

(f) C3-C2

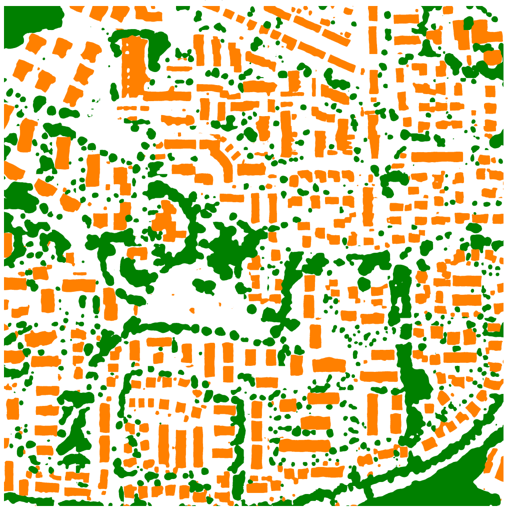
A.4. Predictions after deep Coral adaptation on l7 and logits

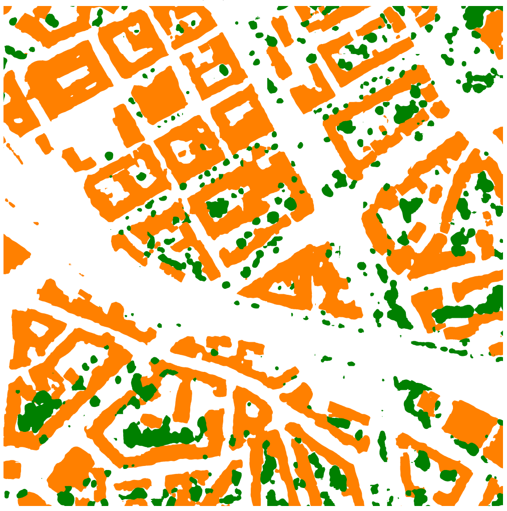

(a) C1-C2


(b) C1-C3

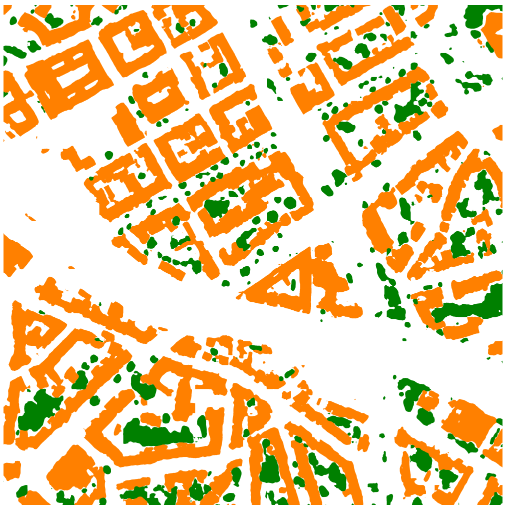

(c) C2-C1

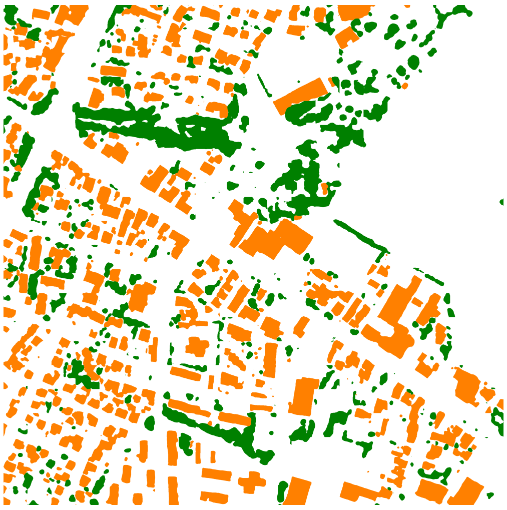

(d) C2-C3

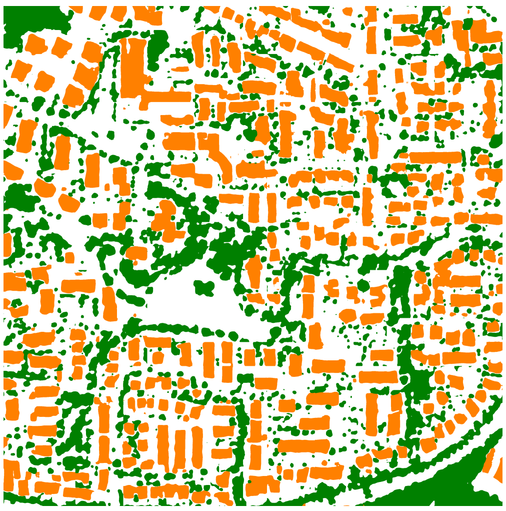

(e) C3-C1

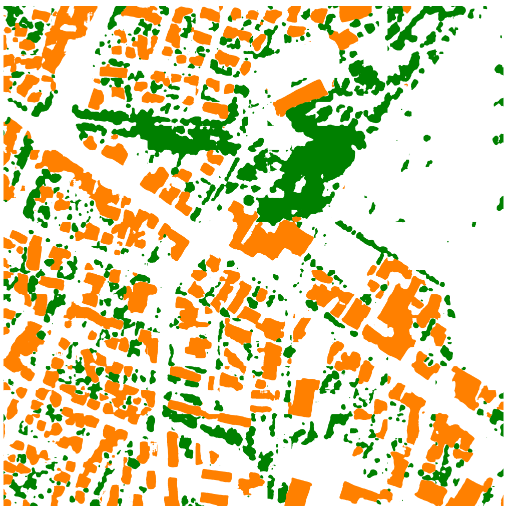

(f) C3-C2

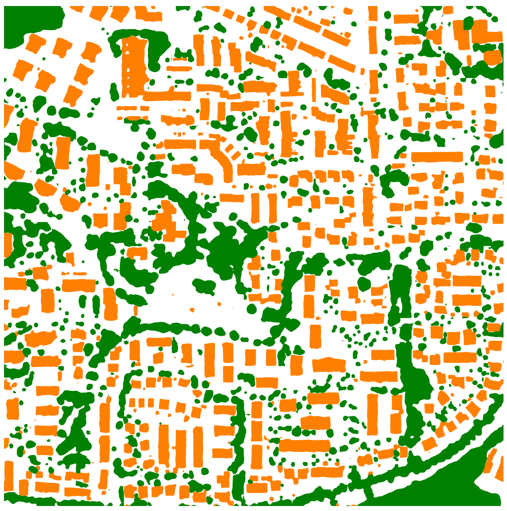
A.5. Predictions after MMD adaptation on l7

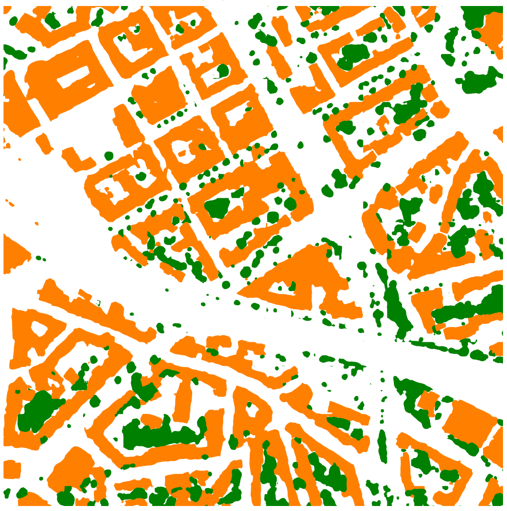

(a) C1-C2


(b) C1-C3


(c) C2-C1


(d) C2-C3


(e) C3-C1


(f) C3-C2

B. Predictions on ISPRS data set

B.1. Predictions in the same domain

Figure 38: Potsdam(Left: ground truth, right: prediction, top to bottom: patch 14, 22, 35)

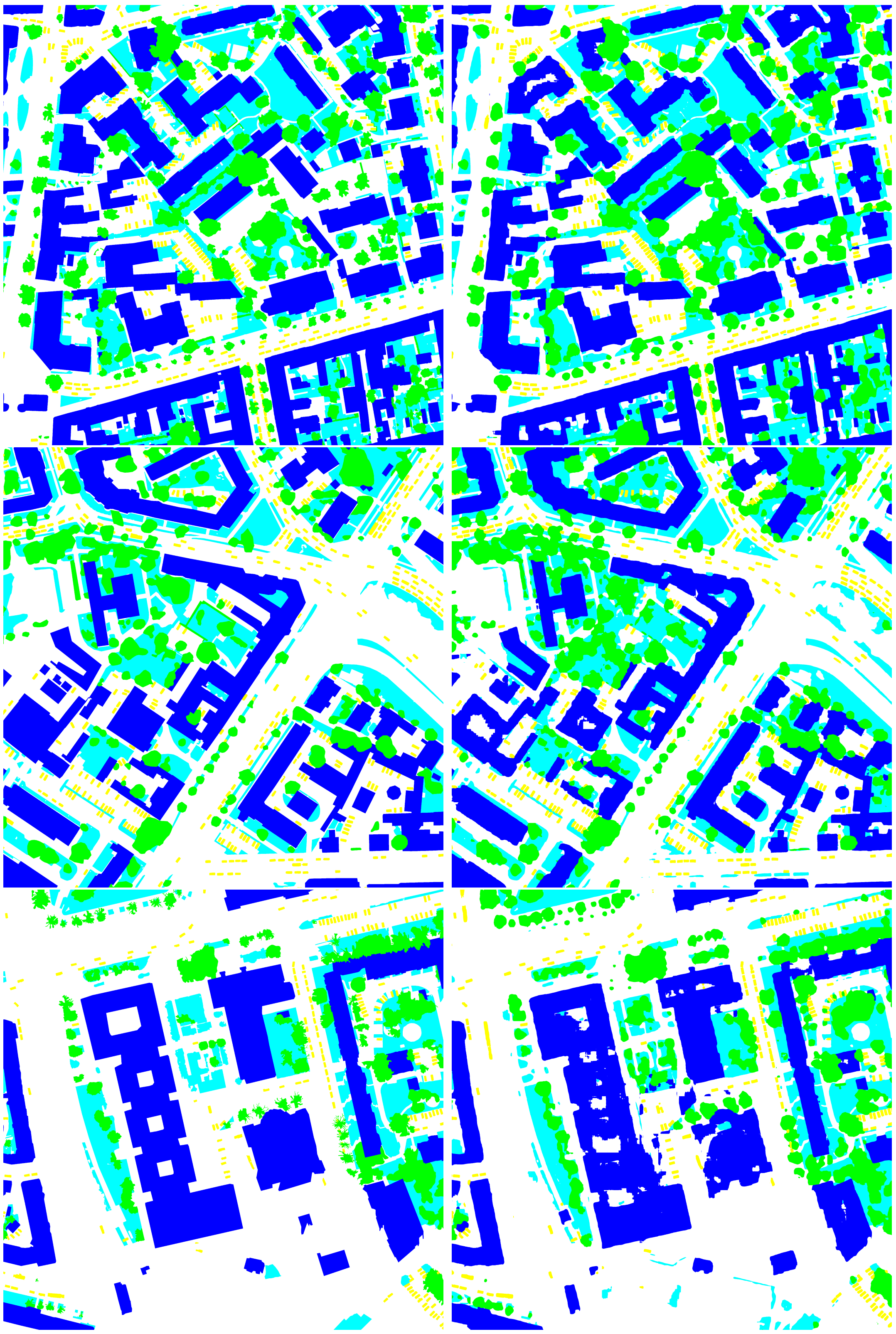
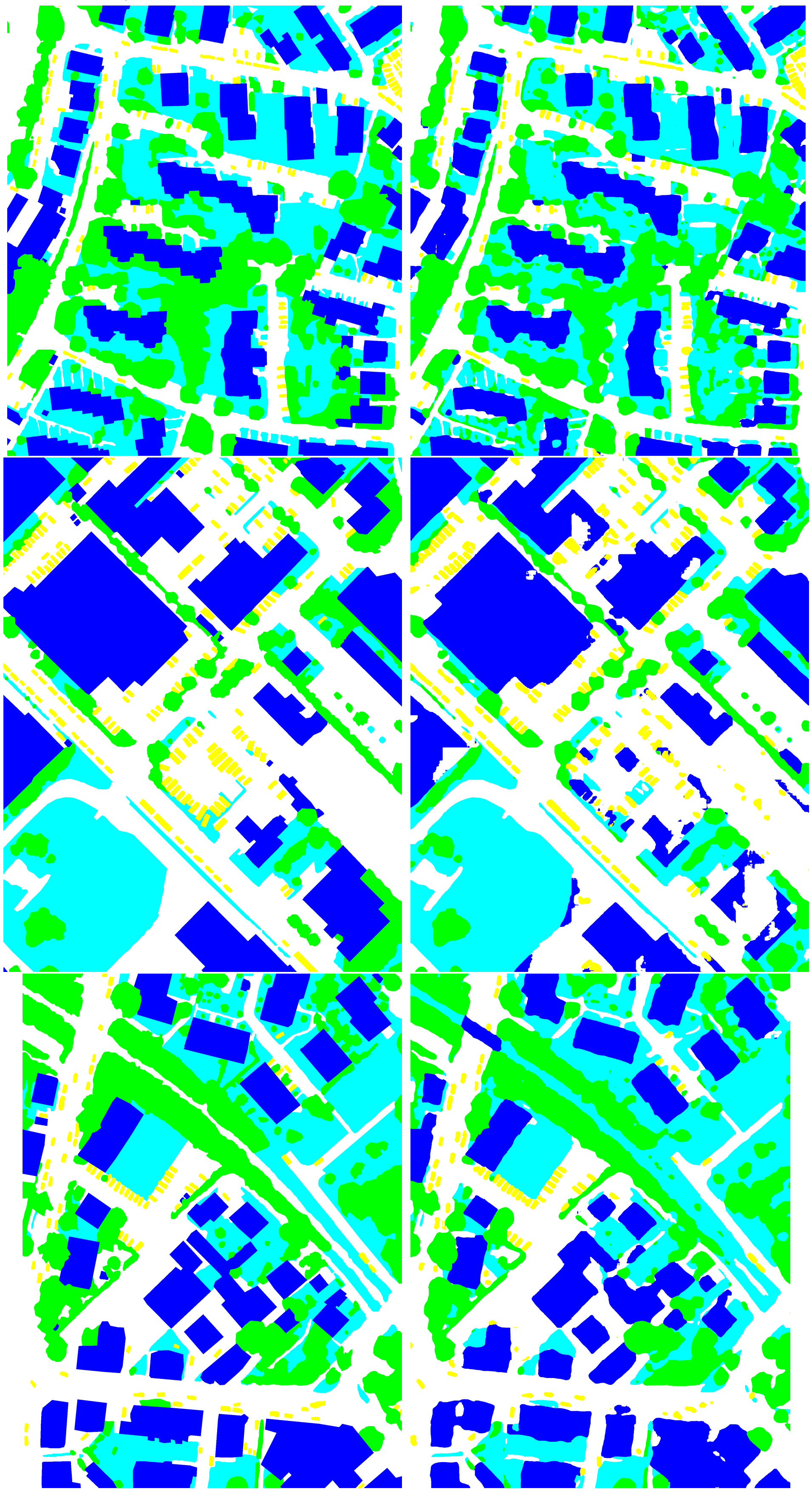
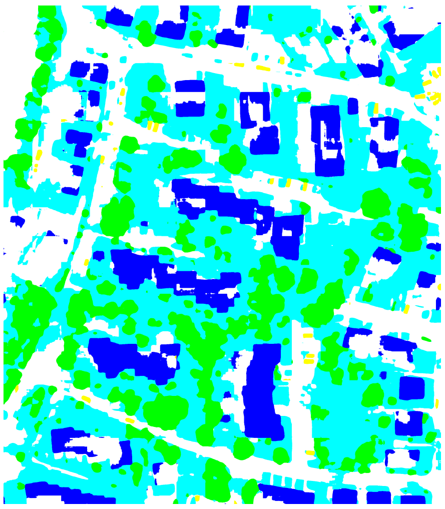
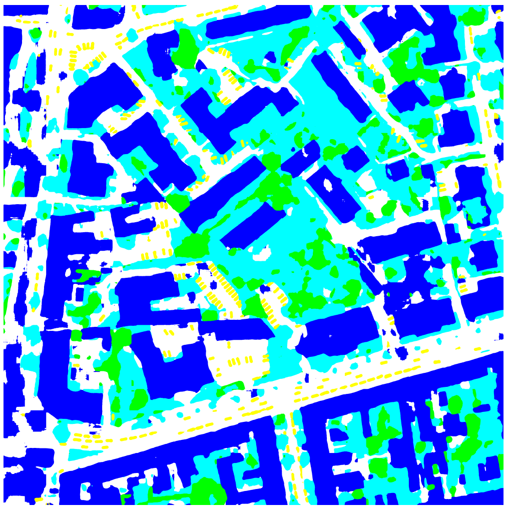


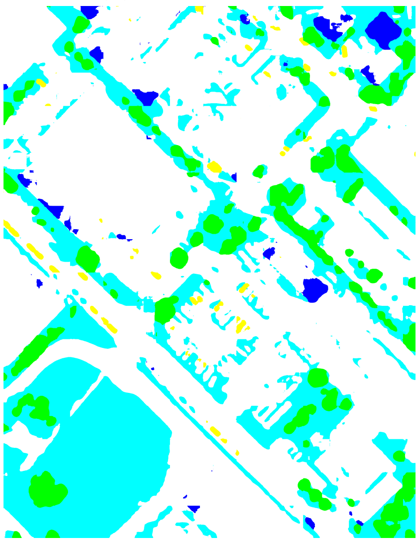
Figure 39: Vaihingen(Left: ground truth, right: prediction, top to bottom: patch 10, 22, 33)

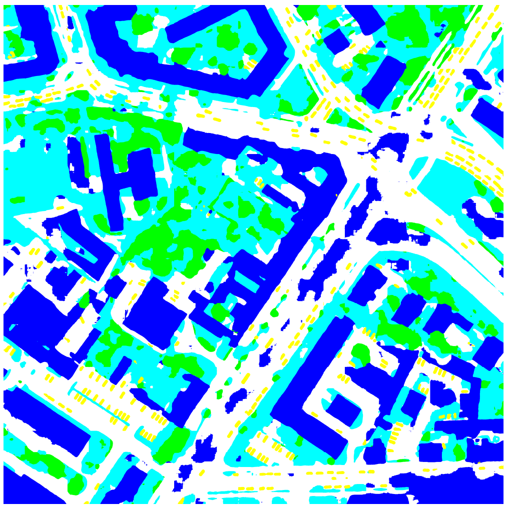


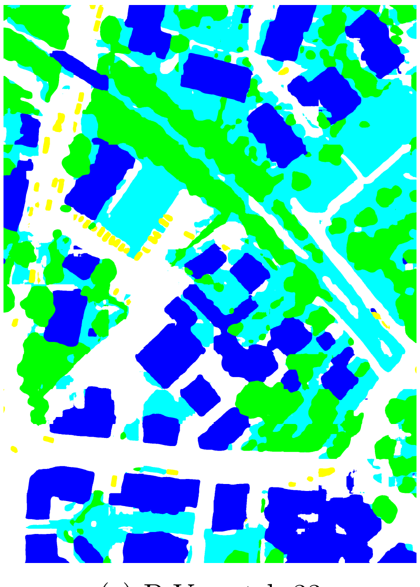
B.2. Predictions on the target domain without DA

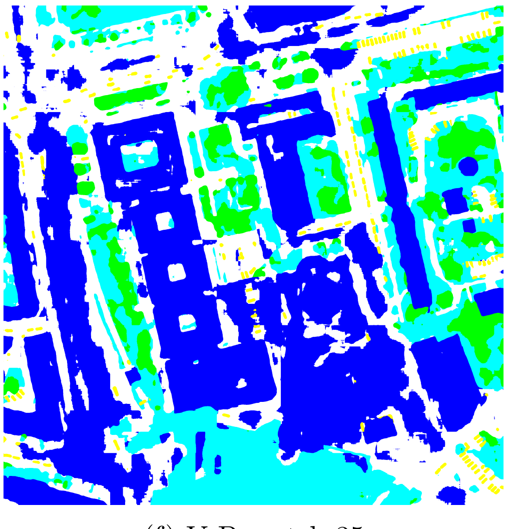

(a) P-V: patch 10


(b) V-P: patch 14

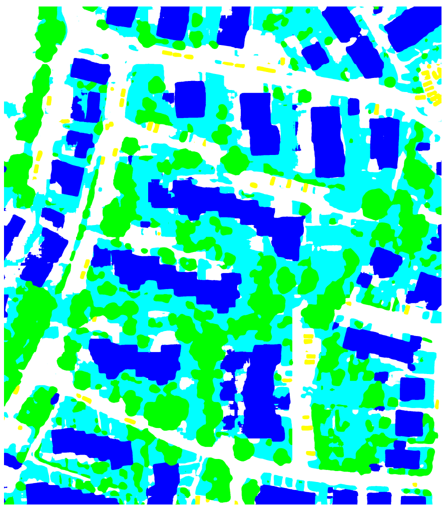

(c) P-V: patch 22

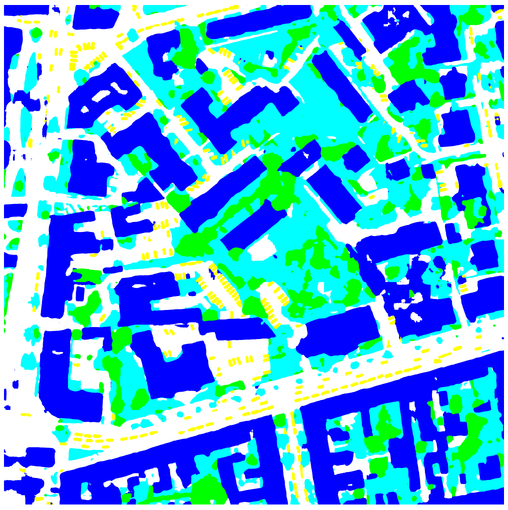

(d) V-P: patch 22

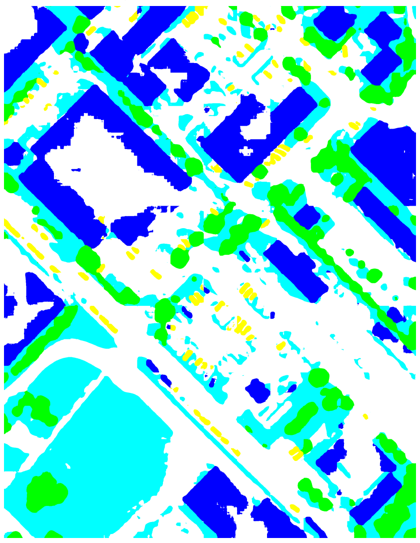

(e) P-V: patch 33

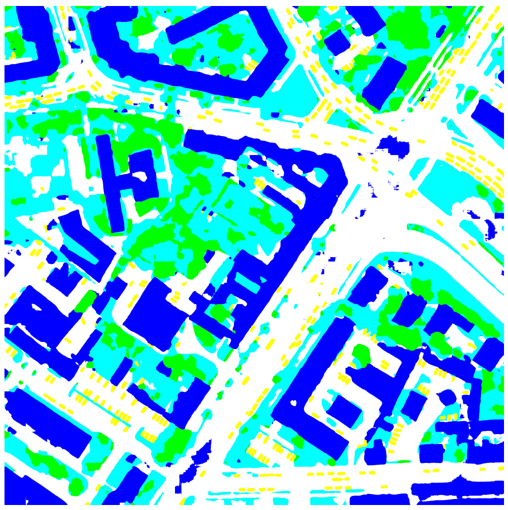

(f) V-P: patch 35

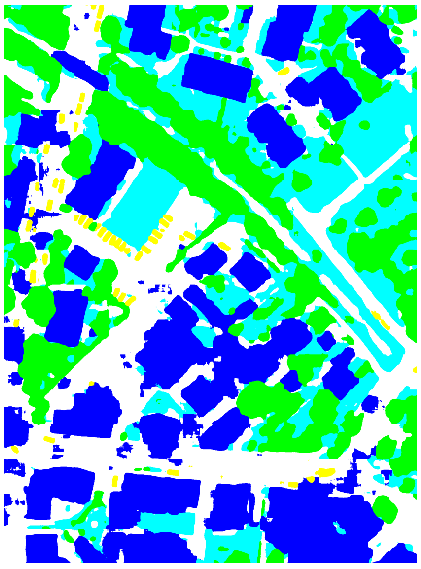
B.3. Predictions after deep Coral adaptation on l7

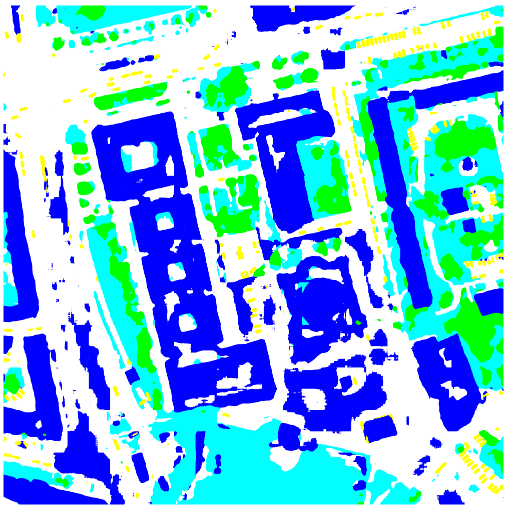

(a) P-V: patch 10


(b) V-P: patch 14

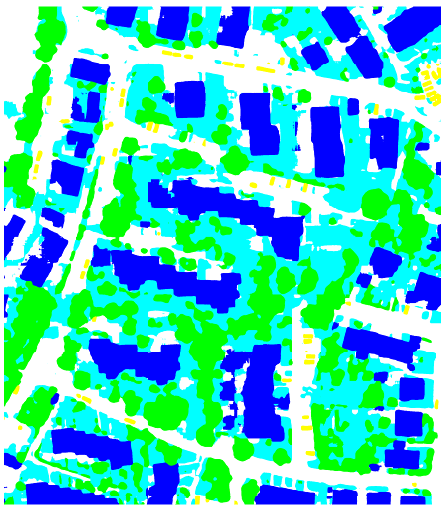

(c) P-V: patch 22

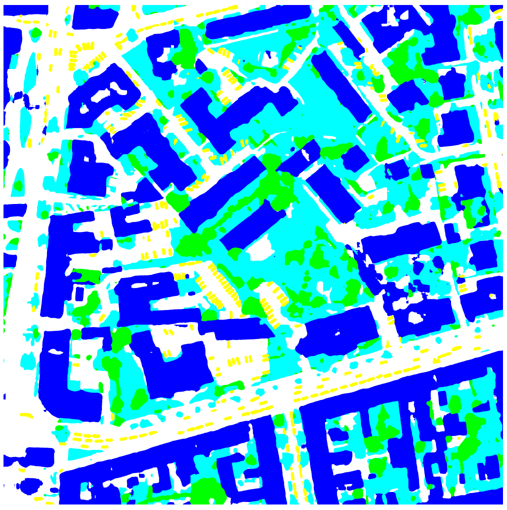

(d) V-P: patch 22

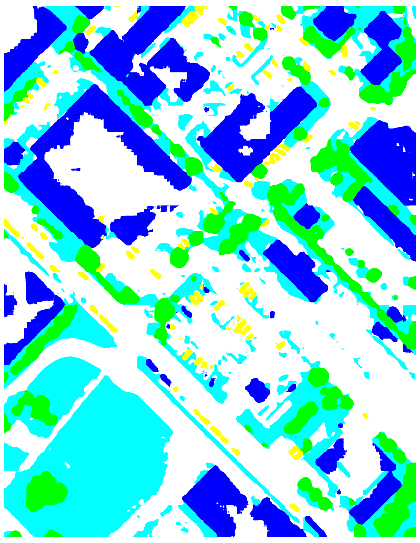

(e) P-V: patch 33

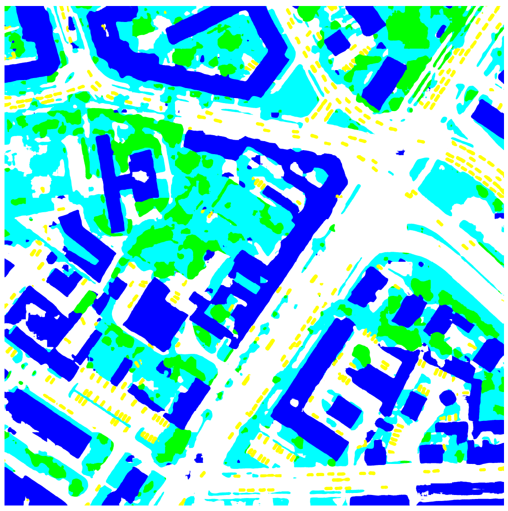

(f) V-P: patch 35

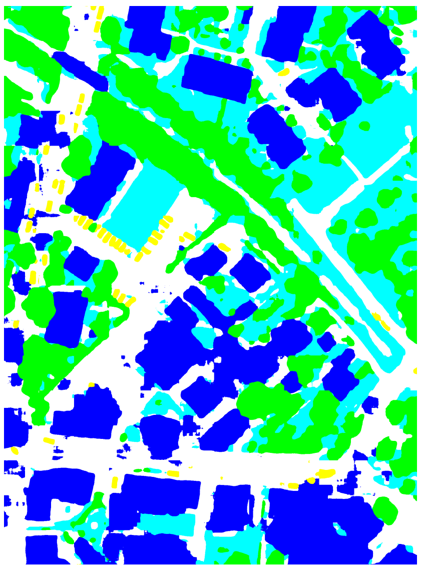
B.4. Predictions after deep Coral adaptation on l7 and logits

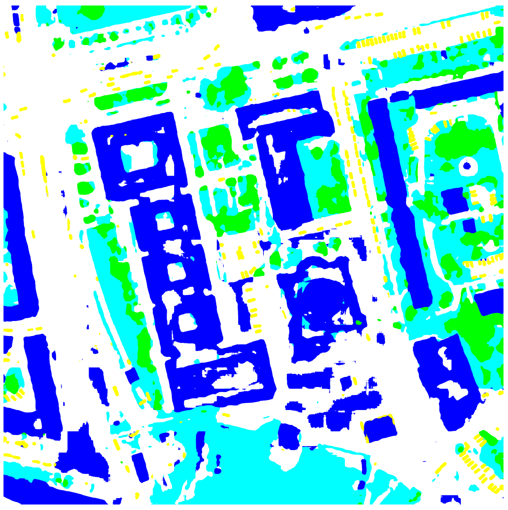

(a) P-V: patch 10


(b) V-P: patch 14

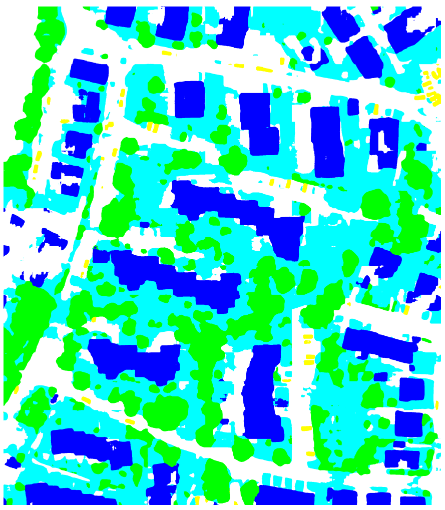

(c) P-V: patch 22

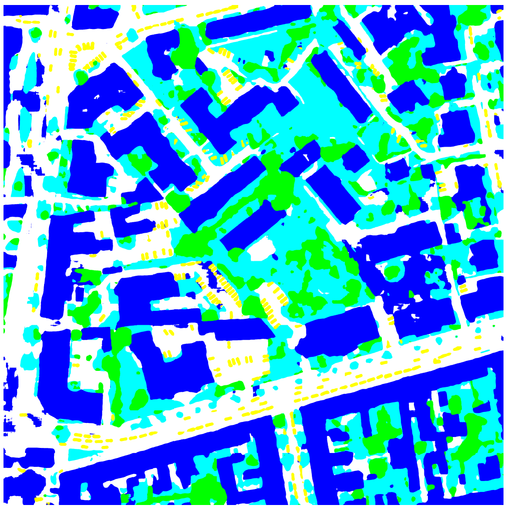

(d) V-P: patch 22

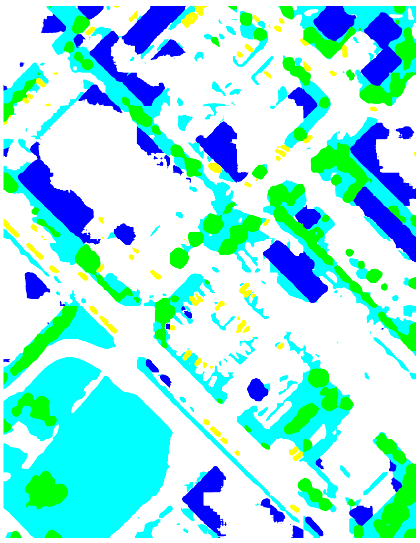

(e) P-V: patch 33

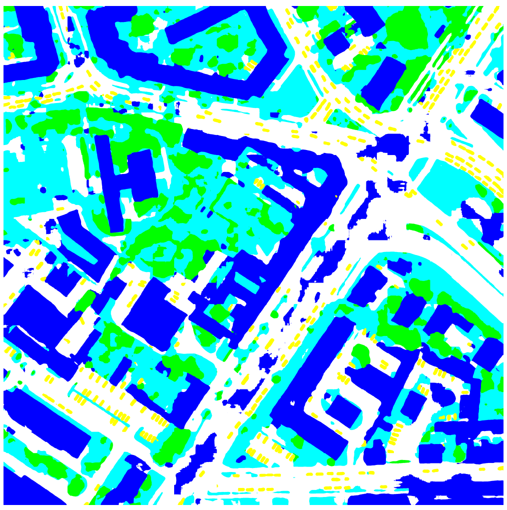

(f) V-P: patch 35

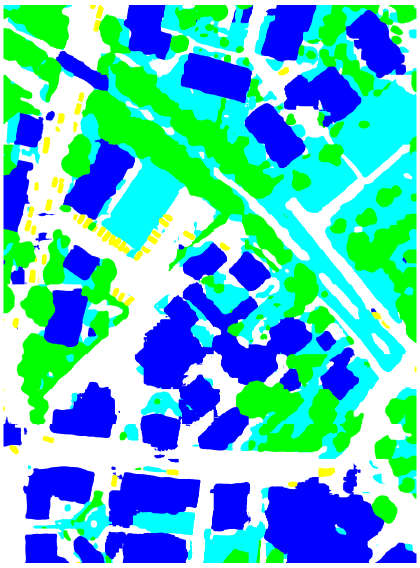
B.5. Predictions after MMD adaptation on l7

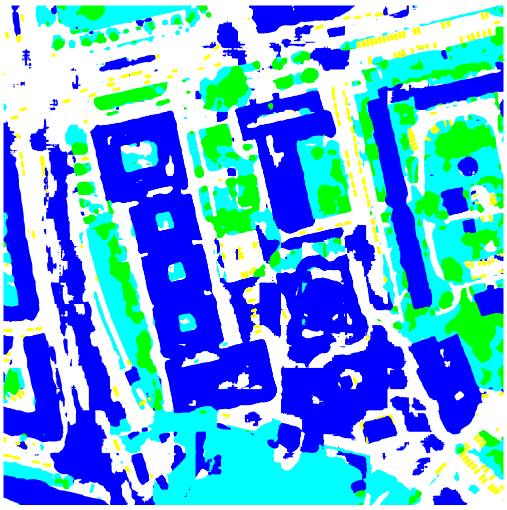

(a) P-V: patch 10


(b) V-P: patch 14


(c) P-V: patch 22


(d) V-P: patch 22


(e) P-V: patch 33


(f) V-P: patch 35