USING SEMANTICALLY PAIRED IMAGES TO IMPROVE DOMAIN ADAPTATION FOR THE SEMANTIC SEGMENTATION OF AERIAL IMAGES

D. Gritzner[1,*], J. Ostermann[1]

[1] Institut für Informationsverarbeitung, Leibniz Universität Hannover, Germany - {gritzner|ostermann}@tnt.uni-hannover.de

KEY WORDS: Transfer Learning, Domain Adaptation, Semantic Segmentation, Aerial Images, Neural Networks, Deep Learning

ABSTRACT:

Modern machine learning, especially deep learning, which is used in a variety of applications, requires a lot of labelled data for model training. Having an insufficient amount of training examples leads to models which do not generalize well to new input instances. This is a particular significant problem for tasks involving aerial images: often training data is only available for a limited geographical area and a narrow time window, thus leading to models which perform poorly in different regions, at different times of day, or during different seasons. Domain adaptation can mitigate this issue by using labelled source domain training examples and unlabeled target domain images to train a model which performs well on both domains. Modern adversarial domain adaptation approaches use unpaired data. We propose using pairs of semantically similar images, i.e., whose segmentations are accurate predictions of each other, for improved model performance. In this paper we show that, as an upper limit based on ground truth, using semantically paired aerial images during training almost always increases model performance with an average improvement of 4.2% accuracy and .036 mean intersection-over-union (mIoU). Using a practical estimate of semantic similarity, we still achieve improvements in more than half of all cases, with average improvements of 2.5% accuracy and .017 mIoU in those cases.

1. INTRODUCTION

Many applications in a variety of domains rely on automatically extracting useful information from images: person identification, object detection and tracking, defect detection during production, semantic segmentation of images and many more. Semantic segmentation, i.e., assigning a semantically meaningful class label to each pixel in an image, is of particular interest for aerial images. Such a segmentation serves as the basis for applications such as creating and updating maps, tracking city growth over time, or tracking deforestation over time.

Most modern machine learning approaches for computing a semantic segmentation of an image use deep learning techniques. From a collection of training examples, i.e., pairs of images and their respective segmentations, a model is learned. Such models usually consist of tens of millions of learnable parameters, with some of the larger models even reaching into the hundreds of millions of parameters. In other domains, e.g., natural language processing, models with more than a billion parameters exist (Radford et al., 2019).

Training large models requires large, diverse sets of training examples. Too few training examples compared to the number of parameters lead to the model overfitting on the training data. The model merely memorizes all the segmentations used during training. However, for a model to be actually useful in a real application, it has to compute features which generalize well to new images. Furthermore, a model trained to compute a semantic segmentation of large cities such as London or Berlin into buildings, roads and trees will likely perform poorly when used on images from rural areas. Acquiring sufficiently large and diverse datasets is a very time- and labor-intensive process and thus it is expensive and often impractical.

To mitigate this issue, transfer learning, in particular domain adaptation, can be used. The goal of domain adaptation is to reuse knowledge learned from one dataset, the so-called source domain, which consists of sufficiently many training examples, and apply it to another dataset, the so-called target domain, for which little to no training examples are available, i.e., only a set of input images without segmentation is known.

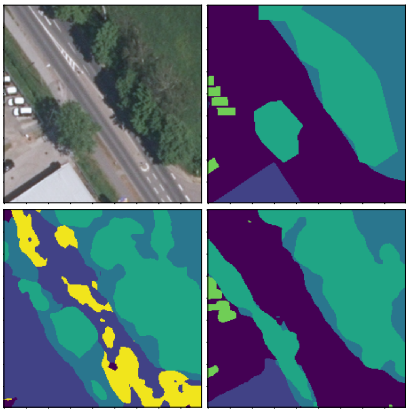


Figure 1. Using semantically paired images (bottom-right; our idea) enhances existing adaptation approaches (bottom-left) and brings them closer to the ground truth (top-right).

A lot of research on domain adaptation for semantic segmentation using deep learning has been published in recent years. However, the area of focus of most research is the segmentation of street scenes with the goal of improving the semantic data available to autonomous cars. The most popular domain adaptation setting is the transfer of knowledge from the SYNTHIA dataset (Ros et al., 2016) to the Cityscapes dataset (Cordts et al., 2016), both of which consist of street scenes from the point of view of a car. Both datasets have in common that there is a spatial prior for each class: buildings are most likely to appear in the top corners of an image, sky is most likely to appear in the top-center and the road is most likely to appear in the bottom

* Corresponding author

half of an image. These spatial priors, which are visualized for SYNTHIA in Fig. 2 of (Zou et al., 2018), make it more likely for the same pixel position in two random images from either dataset to belong to the same semantic class. Aerial images do not have such spatial priors due to their point of view. Thus, two random street scenes are likely to be semantically similar, while aerial images are not.

Inspired by (Kuhnke, Ostermann, 2019), using a model's prediction as pseudo ground truth, and this difference between street scenes and aerial images, we propose using semantically paired images for domain adaptation. In this paper we demonstrate that such a pairing improves the performance of domain adaptation, as shown in Fig. 1. Furthermore, we demonstrate an approach which approximates image pairing based on semantic similarity without using any knowledge of the ground truth segmentation of the target domain.

This paper is structured as follows: Sect. 2 explains the theoretical foundations, followed by related work in Sect. 3. Our contribution is then described in Sect. 4 and evaluated in Sect. 5. The paper finishes with a conclusion in Sect. 6.

2. DOMAIN ADAPTATION

This section explains the foundations necessary for understanding our contribution, while focusing on a deep learning context.

2.1 Foundations

The goal of *domain adaptation* is reusing knowledge learned from one dataset and applying it to another dataset. More precisely, a *domain* D is a tuple $(\mathcal{X}, P(X))$ where $\mathcal{X}$ is a feature space and $P(X)$ with $X = \{x_1, x_2, \ldots, x_n\}, x_i \in \mathcal{X}$, is a marginal probability distribution. In our case, $\mathcal{X}$ is the set of all aerial images and P specifies how likely each image is. Furthermore, there is a *task* $T = (Y, f : \mathcal{X} \rightarrow Y)$ in a domain adaptation setting, with Y being a label space and f being a function mapping instances $x_i \in \mathcal{X}$ to their label $y_i \in Y$. Again, in our case, Y is the set of semantic segmentations and f is the model being learned from a set of training examples (x_i, y_i).

In a common supervised machine learning setting there is only one domain D and one task T. However, in domain adaptation, there are two domains and two tasks: the *source domain* $D^S = (\mathcal{X}, P(X^S))$ with the *source task* $T^S = (Y, f^S)$ and the *target domain* $D^T = (\mathcal{X}, P(X^T))$ with the *target task* $T^T = (Y, f^T)$. The underlying feature space $\mathcal{X}$ of both domains is the same, e.g. RGB images, but the distribution P of instances differs for both domains. This difference is called *domain shift* or *domain gap*. Also, the same task shall be performed for both domains, e.g., assigning the correct semantic segmentation from Y to each instance x_i^S or x_i^T. However, the actual function f, i.e., the model, used to perform this mapping of instances to elements in Y may differ between domains. Fig. 2 shows a classification task (geometric shape) for two domains (color). It also demonstrates that a model trained on either domain is likely to perform poorly on the other domain.

There are two common domain adaptation settings: in the *semi-supervised* setting only a small number of training samples (x_i^T, y_i^T) in the target domain are known, while in the *unsupervised* setting, which is our focus, no target domain training samples are known at all. In both settings, a large amount of source domain training samples (x_i^S, y_i^S) are readily available.

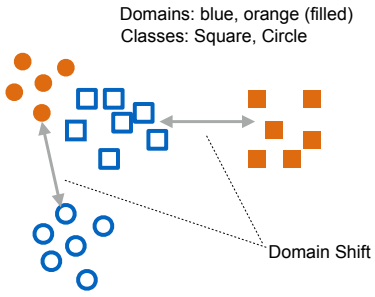


Figure 2. Two domains (color) in a two-dimensional instance space with a classification task (geometric shape)

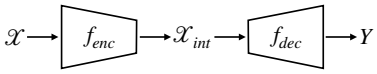


Figure 3. Visualization of an abstract deep learning model f

2.2 Deep Learning-based Domain Adaptation

Current deep learning models used for solving semantic segmentation tasks $T = (Y, f)$ have an encoder/decoder structure: the function f is subdivided into an encoder function f_{enc} and a decoder function f_{dec}, s.t. $y_i = f(x_i) = f_{dec}(f_{enc}(x_i))$, i.e., the function $f_{enc} : \mathcal{X} \rightarrow \mathcal{X}_{int}$ maps instances x_i to an intermediate feature space $\mathcal{X}_{int}$, while $f_{dec} : \mathcal{X}_{int} \rightarrow Y$ maps instances from the intermediate space to the label space. Fig. 3 visualizes this approach. Given a particular sequential deep learning model f, the ouput of any arbitrary model layer can be chosen as the desired intermediate feature space $\mathcal{X}_{int}$. A common choice for $\mathcal{X}_{int}$ is a bottleneck in the model, i.e., where the intermediate tensors $x_{i,int} \in \mathcal{X}_{int}$ have a low spatial resolution.

Deep learning-based domain adaptation approaches can be classified based on what part of the chain $y_i = f_{dec}(f_{enc}(x_i))$ they affect. *Appearance adaptation* or *instance transfer* is the transformation of source instances x_i^S s.t. they look like instances drawn from $P(X^T)$ or vice versa. We denote such transformed source instances as $x_i^{S \rightarrow T}$. In *feature level adaptation* the goal is to map instances x_i^S and x_i^T to a common intermediate feature space $\mathcal{X}_{int}$ shared by both domains. Domain adaptation through *parameter transfer* means reusing some of the, potentially transformed, parameters of the model f^S in the model f^T. Which parts of a model f are affected by the different kinds of domain adaptation approaches is shown in Fig. 4.

2.3 Adversarial Training

The currently most popular approaches for unsupervised domain adaptation for deep learning models are based on adversarial training. The idea comes from so called Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). In a GAN, a generator model attempts to create artificial data, e.g., images, from noise, while a discriminator model tries to distinguish between real data and artificial data. Properly training these two adversarial models results in a generator model

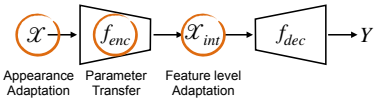


Figure 4. Parts of an abstract deep learning model f affected by different kinds of domain adaptation

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2020-483-2020 | © Authors 2020. CC BY 4.0 License.

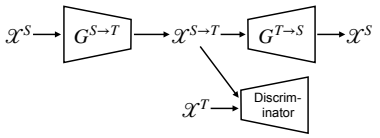


Figure 5. Partial Cycle-GAN approach for appearance adaptation. The generator $G^{S\to T}$ tries to fool the discriminator, i.e., ideally the set of generated instances $\mathcal{X}^{S\to T}$ looks indistinguishable from $\mathcal{X}^T$.

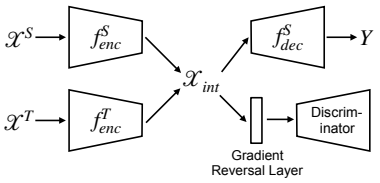


Figure 6. DANN approach for feature level domain adaptation

which is capable of transforming data drawn from one distribution (noise) into data drawn from another distribution (images).

In a Cycle-GAN (Zhu et al., 2017), visualized in Fig. 5, two generator models and two discriminator models are used for appearance adaptation. The generator $G^{S\to T}$ maps source instances x_i^S to the target domain's marginal probability distribution $P(X^T)$, while the generator $G^{T\to S}$ performs the mapping in the opposite direction. The generators are trained s.t. cycle consistency holds, i.e., $x_i^S = G^{T\to S}(G^{S\to T}(x_i^S))$ and $x_i^T = G^{S\to T}(G^{T\to S}(x_i^T))$. Additionally, the generators must generate outputs $x_i^{S\to T}$ and $x_i^{T\to S}$ which the two discriminators cannot distinguish from real data of the respective domain.

In the DANN approach (Ganin et al., 2016) for feature level adaptation a discriminator tries to distinguish instances from both domains in the intermediate feature space. The approach is shown in Fig. 6. A gradient reversal layer between the discriminator and the two encoder models f^S_{enc} and f^T_{enc} ensures that, while the discriminator gets increasingly better at distinguishing the instances, the encoder models learn to map to a shared intermediate feature space $\mathcal{X}_{int}$. Thus, eventually it becomes impossible to tell the instances from the two domains apart. When this point is reached, f^S_{dec} can be used as a decoder for both encoders, in particular, for f^T_{enc}.

3. RELATED WORK

Modern domain adaptation approaches are based on adversarial training, usually combining the DANN approach (Ganin et al., 2016) and the Cycle-GAN approach (Zhu et al., 2017). Notable approaches combining the two are CyCADA (Hoffman et al., 2017) and CrDoCo (Chen et al., 2019b). Additionally, they enforce cross-domain consistency, i.e., the prediction for an input instance should be the same regardless of which domain said instance currently appears to belong to.

(Kuhnke, Ostermann, 2019) follows a similar approach to ours, but for head pose estimation, i.e., a regression task rather than a semantic segmentation task. They use a model pre-trained on the source domain to generate predictions for the target domain. Then, they sample from each domain during a DANN-like domain adaptation training s.t. the label distributions are similar for samples from both domains. We also propose using the prediction of a pre-trained model.

(Tsai et al., 2019) cluster their input images based on the source domain class distribution and use adversarial training to implicitly map each target image to a cluster. This auxiliary mapping task causes the encoders to learn features representing clusters instead of features representing domain-specific details. We propose using an explicit mapping to be able to compose mini-batches in a specific way. Also, our similarity metric is stronger than just comparing class distributions.

Other related works use the maximum mean discrepancy (MMD) instead of a discriminator for domain adaptation and related tasks such as source selection. Notable works include (Vogt et al., 2018), which does not rely on deep learning at all, but still is able to reach competitive adaptation performance in terms of the traditional machine learning model used for classification. (Long et al., 2018) use an MMD-based loss function instead of a discriminator in an otherwise DANN-like approach.

While parameter transfer, other than weight sharing, is rather rare in modern approaches, (Rozantsev et al., 2018) explicitly model the domain shift as a linear function mapping the weights of one domain's model to the other domain's model.

Other domain adaptation ideas include entropy minimization (Vu et al., 2019), adversarial output regularization (Tsai et al., 2018), separating domain-invariant structure from domain-specific texture (Chang et al., 2019), or using a maximum squares loss function (Chen et al., 2019a).

4. SEMANTIC SIMILARITY

This section formalizes our contribution, a metric for measuring the semantic similarity of two instances x_i and x_j and an approach for approximating this metric for instance pairs for which the ground truth segmentation is unknown for one of the instances. Our contribution is thoroughly evaluated in an empirical manner in Sect. 5.

4.1 Semantic Similarity Metric

As explained in the introduction, the class-wise spatial priors of the SYNTHIA and Cityscapes datasets means that two random images drawn from either dataset are likely to contain a high amount of pixel positions, which have the same semantic class in both images. This is what we intuitively understand as semantically similar images and we formalize this intuition as a metric.

Note: For brevity, we often write just x_i instead of (x_i, y_i). The context will make it clear whether we just use x_i or also its ground truth segmentation y_i.

Definition: Let x_i and x_j be two instances with their respective segmentations $y_i : L \to C$ and $y_j : L \to C$ with L being the common set of all spatial locations (pixel positions) in both instances and C being the common set of all semantic classes. The *semantic similarity* $SemSim$ of these instances is

$$SemSim(x_i, x_j) = \frac{|\{l \in L \mid y_i(l) = y_j(l)\}|}{|L|}. \quad (1)$$

This definition is the same definition as used for accuracy. In other words, by our definition two instances are semantically similar, if the segmentation of one instance is an accurate prediction of the segmentation of the other instance.

The semantic similarity is a metric which assigns values from the interval $[0, 1]$ to every pair of instances x_i and x_j. Higher values indicate a high similarity, with a value of 1 being assigned iff the segmentations y_i and y_j are identical. A high similarity also implies a high overlap in the class probability distributions of both segmentations. If a class is likely to appear in one segmentation of two semantically similar instances, then it must also be likely to appear in the other segmentation, otherwise the semantic similarity would be low. These class probability distributions are identical if the similarity is 1. However, the opposite is not true: two segmentations can have the same class probability distribution while their similarity is 0. Imagine a segmentation y_i which is partitioned into two segments of equal size, each with a different class. A segmentation with flipped labels will not be similar by our definition, but the class probability distribution will be identical.

4.2 Semantic Similarity Approximation

We will show in Sect. 5 that, in an unsupervised domain adaptation scenario, it is desirable to create pairs of instances x_i^S and x_j^T, s.t. $SemSim(x_i^S, x_j^T)$ is maximized over all $x_j^T \in \mathcal{X}^T$ for any given $x_i^S \in \mathcal{X}^S$. Using these pairs together in mini-batches to train a deep learning model will result in an improved model. However, to compute $SemSim(x_i^S, x_j^T)$, as defined in the previous subsection, their respective ground truth segmentations y_i^S and y_j^T must be known. As the latter is unknown, in fact, our very goal is to learn a model which can predict y_j^T well, we need an approximation of $SemSim(x_i^S, x_j^T)$ in our setting.

To approximate $SemSim$, we use a semantic segmentation model f^S learned using the source training samples (x_i^S, y_i^S), as inspired by (Kuhnke, Ostermann, 2019). The output tensor $\hat{y} = f^S(x)$ of the model is itself a function $\hat{y} : L \times C \to [0, 1]$ which assigns class probabilities to every location $l \in L$, s.t. $\sum_{c \in C} \hat{y}(l, c) = 1 \ \ \forall l \in L$. Some models require an additional Softmax layer, which does not have any parameters which need to be learned, to map class scores to class probabilities.

With the model f^S we compute the approximation $SemSim^*$ using the equation

$$SemSim^*(x_i^S, x_j^T, f^S) = \frac{1}{|L|} \sum_{l \in L} \hat{y}_j^T(l, y_i^S(l)) \qquad (2)$$

with $\hat{y}_j^T = f^S(x_j^T)$ and y_i^S being the ground truth segmentation of x_i^S. $SemSim^*$ is the mean of the probabilities assigned to the ground truth classes $y_i^S(l)$ by $\hat{y}_j^T \ \ \forall l \in L$. $SemSim^*$ is high if the model f^S is confident that $SemSim(x_i^S, x_j^T)$ is high.

4.3 Mini-Batch Composition

During model training, we will not actually compute $SemSim$ or $SemSim^*$. Rather, we propose pre-computing a mapping $M : \mathcal{X}^S \to \mathcal{X}^T$. Then, when creating a mini-batch during training, for every $x_i^S \in \mathcal{X}^S$ we put into the mini-batch we also put $M(x_i^S) \in \mathcal{X}^T$ into the same mini-batch.

We compute the mapping $M : \mathcal{X}^S \to \mathcal{X}^T$ using the equation

$$M(x_i^S \in \mathcal{X}^S) = \underset{x_j^T \in \mathcal{X}^T}{\arg\max} \ SemSim^*(x_i^S, x_j^T, f^S). \qquad (3)$$

Using $SemSim^*$ is very similar to using the predictions $f^S(x_j^T \in \mathcal{X}^T)$ as pseudo ground truth segmentations, however, we explicitly take the model's confidence into account. This is done to avoid small changes in f^S or x_j^T, e.g., noise, causing drastic changes in $SemSim^*(x_i^S, x_j^T, f^S)$ due to uncertainty in large regions $L^* \subseteq L$ causing the labels in said regions to flip to or from the correct ground truth class.

Dataset	Vaihingen	Potsdam	3Cities
no. of images	33	38	1 per city
Image Resolution [pixels]	2336×1281 to 3816×2550	6000×6000	10000×10000
Ground Sampling Distance [cm/pixel]	9	5	20
channels	near-infrared, red, green, depth	near-infrared, red, green, blue, depth	near-infrared, red, green, blue, depth

Table 1. Dataset details; 3Cities consists of the Hannover, Buxtehude and Nienburg data.

5. EVALUATION

As a second part to our contribution, we evaluate our proposed metric and its approximation in an thorough empirical evaluation. As evaluation metrics, we use accuracy and mean Intersection-over-Union (mIoU). Though these are commonly used metrics used, we still included their definition as a reminder in the appendix.

5.1 Datasets

We used five datasets, each dataset associated with a different German city. We used the two ISPRS 2D Semantic Labeling Benchmark Challenge datasets consisting of images from Vaihingen and Potsdam. Furthermore, we used images of Hannover, Buxtehude and Nienburg. The images are true orthophotos of the respective cities. Tab. 1 contains details about each dataset. To make the datasets comparable in terms of sensor data we resampled Vaihingen and Potsdam to a ground sampling distance of 20cm/pixel. Furthermore, we did not use any depth information as it was unreliable for at least one domain (Hannover) and we did not use the blue channels as no such information is available for Vaihingen. From Vaihingen and Potsdam we only used the 16 images initially released as training set for the respective benchmark challenges. The images of Hannover, Buxtehude and Nienburg were divided into 16 patches of 2500×2500 pixels. One such Hannover patch was omitted due it containing almost exclusively one class (tree). The image or image patches of each dataset were randomly partitioned into a training set and a validation set by randomly drawing ten images as training images and using the remaining images as validation images. The ground truth segmentations of each dataset include six classes: impervious surfaces, buildings, low vegetation, tree, car, and clutter/background. Their distribution is shown in Fig. 7. Hannover, as the largest city out of the five, has more buildings than the others. The distribution of Vaihingen is closer to that of Potsdam, despite Vaihingen being closer to Buxtehude and Nienburg in size. In the latter two, low vegetation is more common than in the other datasets. While the classes car and clutter are rare in all datasets, Potsdam has significantly more clutter than the other datasets.

5.2 Test Setup

We created actual training sets for each dataset by randomly cropping 224×224 image patches from the training images. We

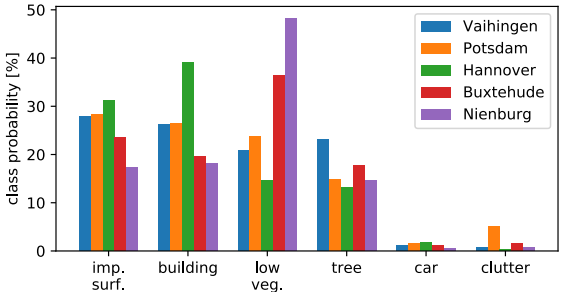


Figure 7. Class distribution of the datasets

used random translations, rotations and shearing for data augmentation. At image boundaries, we used reflection padding. We drew 800 image patches from each of the ten images, resulting in 8000 image patches as the training set for each dataset/domain. The validation image patches were created by sliding a 224×224 window with stride 224×224 over the validation images. At the right-hand and bottom boundaries of each image we moved the sliding window inwards s.t. it fits entirely inside the image, creating a small overlap region with the previous window. Thus, we avoided interpolation and padding for the validation image patches.

As explained in Sect. 4.3 we use mappings $M^{S\to T} : \mathcal{X}^S \to \mathcal{X}^T$ to compose mini-batches for training deep learning models. During any model training we only used the segmentations y_i^S of the source domain.

For every pair of domains we computed three mappings:

- *Random:* We mapped each x_i^S to a random x_j^T.
- *Ground Truth (GT):* We created a mapping based on Eq. 3 but used $SemSim$ instead of $SemSim^*$. For that, through our data augmentation process, we created 16000 image patches per target image instead of just 800, in order to find a better mapping.
- *Approximation:* This mapping is similar to the Ground Truth mapping, however, we actually used Eq. 3, i.e., no target ground truth was required.

All mappings were computed once and then stayed fixed throughout all our experiments.

For the domain pair Hannover (source) and Vaihingen (target) we created three additional mappings, two of which are based on the MMD. The MMD is a measure of how similar two probability distributions are. We used the estimate described in (Vogt et al., 2018) to compute it. The additional mappings are:

- *Shuffled Ground Truth:* Based on the Ground Truth mapping $M_{GT}^{S\to T}$, this mapping assigns a random $x_j^T \in M_{GT}^{S\to T}(\mathcal{X}^S)$ to each x_i^S, s.t. that $M_{GT}^{S\to T}(\mathcal{X}^S)$ is re-used in its entirety. The idea is to have a mapping with the same overall class distribution as the Ground Truth mapping, but with a randomized mini-batch composition.
- *Image space MMD:* For this mapping we treated each pixel in each image patch x_i^S or x_j^T as a sample in terms of the MMD. The mapping creates pairs which minimize the MMD between paired image patches.
- *Feature space MMD:* We used a model f^S, which was comprised of an encoder f_{enc}^S and a decoder f_{dec}^S, to create feature tensors $z_i^S = f_{enc}^S(x_i^S)$ and $z_j^T = f_{enc}^S(x_j^T)$. Treating each spatial location in each such tensor as a sample, we computed the MMD between feature tensors to create pairs $(x_i^S, x_j^T = M^{S\to T}(x_i^S))$, again minimizing the MMD between paired data.

These additional mappings were used to investigate whether $SemSim^*$ offers competitive performance and whether having a similar class distribution across the entire dataset is sufficient or a similar distribution across each mini-batch (Shuffled Ground Truth vs. regular Ground Truth mapping) is beneficial.

As a semantic segmentation model we used DeepLabv3+ (Chen et al., 2018) with MobileNetv2 (Sandler et al., 2018) as a backbone. We chose the same configuration for the spatial pyramid pooling (SPP) as used by the authors on their reference implementation. Furthermore, we used a large feature space, i.e., we only used an overall downsampling factor of 8 for each dimension. This model was also used for computing $SemSim^*$ and the feature space MMD mapping. As a feature space, we picked the output of the 1×1 convolution which combines the concatenated SPP branches into a single feature space with 256 channels per spatial location. To speed up training, we used weights from a pre-training on ImageNet. We then trained the model for an additional 80 epochs with 250 mini-batches, each of size 32, per epoch. We used Adadelta (Zeiler, 2012) with default parameters as optimizer.

As a DANN-like adversarial training approach (Ganin et al., 2016) for feature level adaptation, we took DeepLabv3+ and added a discriminator comprised of three inverted bottlenecks (Sandler et al., 2018). The discriminator assigned a class (source or target) to each spatial location of a feature tensor. We used a shared encoder f_{enc} for both domains. In each training iteration we first trained f_{enc} and f_{dec} to perform a semantic segmentation on the source domain, then we froze the weights f_{enc} to train the discriminator using cross-entropy loss. As a third and final step, we froze the discriminator weights and trained f_{enc} with flipped class labels (source became target and vice versa). This imitates having a gradient reversal layer. Again, we used the Adadelta optimizers (one for each step), trained for 80 epochs with 250 mini-batches each. We had to reduce the batch size to 16, using only a fixed subset of $\mathcal{X}^S$.

For appearance adaptation we used the Cycle-GAN architecture from (Zhu et al., 2017). However, as a final activation for the generators, we used ReLU instead of tanh. We also removed the Sigmoid activation from the discriminators and used mean squared error loss instead of cross entropy (Mao et al., 2017). Additionally, we performed identity loss training every ten mini-batches (Taigman et al., 2016). We used $\lambda = 10$ for the reconstruction/cycle-consistency loss weight and $\lambda = 1$ for all other loss weights. We trained the Cycle-GAN for 200 epochs with 250 mini-batches each. The mini-batch size was set to just

	Accuracy	mIoU
Vaihingen	81.29%	0.540
Potsdam	79.20%	0.593
Hannover	84.81%	0.561
Buxtehude	85.97%	0.728
Nienburg	85.10%	0.689

Table 2. Baseline performance of DeepLabv3+ using supervised training without any domain adaptation

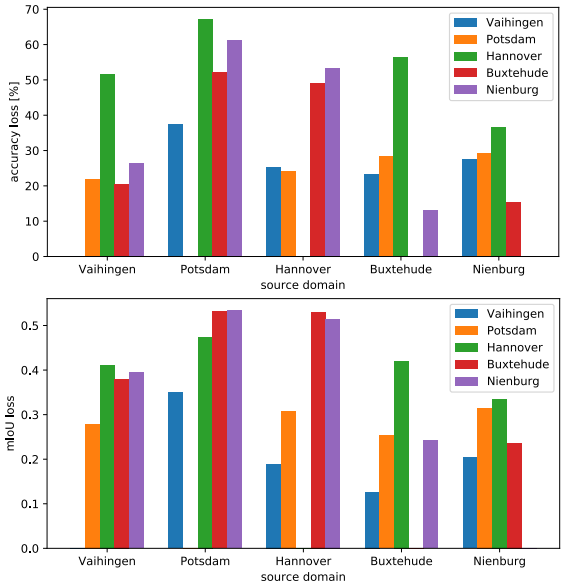


Figure 8. Accuracy and mIoU loss when using a model trained on one domain on another domain

10, again forcing us to use only a fixed subset of our datasets for training. Adam (Kingma, Ba, 2014) was used as optimizer for all parameters. After training the Cycle-GAN, we used the adapted source image patches $x_i^{S \to T}$ to train a semantic segmentation model. This model was then evaluated on the target validation image patches.

5.3 Results

We ran all experiments ten times. The reported numbers in this chapter are the mean values. Furthermore, the reported numbers were calculated on the target validation sets.

Table 2 shows the baseline performance of DeepLabv3+ on our datasets. The model performs well in terms of accuracy, however, the accuracy and mIoU together indicate that it performs better on abundant classes than it does perform on rare classes. Fig. 8 shows the performance loss when using a model trained on one domain to perform a segmentation of another domain as opposed to both domains being the same. 13% to 67% accuracy is lost in such cases. Notable domains are Potsdam, which performs poorly as a source domain, Hannover, which performs poorly as a target domain, and the pair Buxtehude and Nienburg, whose models, even without any domain adaptation, already work relatively well for the other domain.

Next, we investigated the pair Hannover (source domain) and Vaihingen (target domain) more closely for a comprehensive comparison of the mapping strategies using the DANN-like approach. The similarity scores for the different mapping strategies are in Table 3. Even the Ground Truth strategy often cannot guarantee a perfect match, however, it still has the highest score. The second highest score is achieved by our Approximation strategy, with the other strategies lagging behind. Fig. 9 shows our $SemSim^*$-based Approximation strategy (blue curve in all graphs) performing significantly better than Random or either of the MMD-based strategies. However, in almost all curves a slight downward trend, as model training goes on, can be observed. This implies that a short training period likely results in a better model, however, the exact epoch

	mean	std.
Random	.275	.137
Ground Truth	.706	.122
Shuffled GT	.291	.175
Approximation	.573	.167
Image space MMD	.210	.211
Feature space MMD	.424	.142

Table 3. $SemSim$ scores for the domain pair Hannover (source) and Vaihingen (target) using different mapping strategies

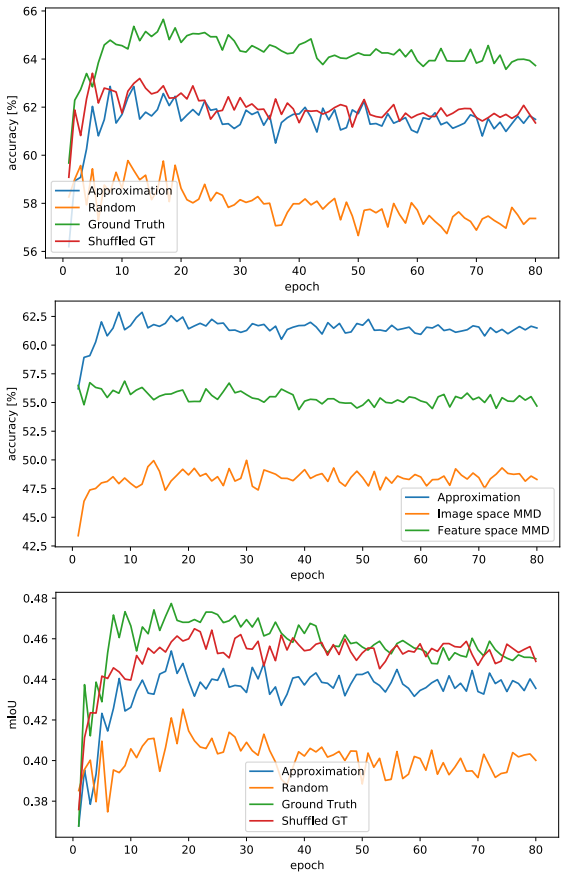


Figure 9. Comparing mapping strategies for DANN-like domain adaptation from Hannover (source) to Vaihingen (target)

after which to stop training cannot be determined in a real world application due to the lack of target ground truth.

Furthermore, we observe that using a better semantic pairing (GT strategy) results in even better performance and that class distribution similarity only (Shuffled GT strategy) still provides a benefit, but less so than semantic similarity. So while the Approximation strategy already increases performance, there is still room for improvement. A limitation of this experiment is that it cannot answer whether aligning the mini-batch class distribution is already sufficient or whether having the same classes at the same pixel locations, as $SemSim$ and $SemSim^*$ are designed to favor, brings an additional benefit.

While the GT strategy is better than Shuffled GT in terms of accuracy, they appear to be similar in terms of mIoU. As Table 4 shows, this is due to the GT strategy performing better on abundant classes such as impervious surfaces and buildings, while Shuffled GT performs significantly better on the rare class clutter. The table also shows that the semantic similarity based

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2020-483-2020 | © Authors 2020. CC BY 4.0 License.

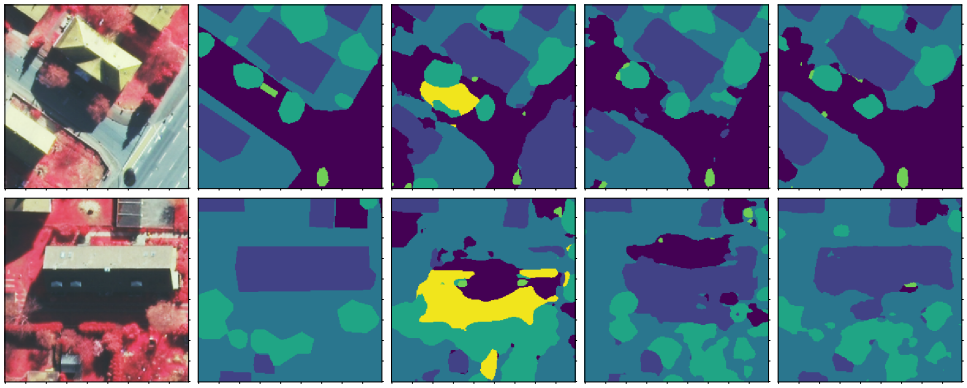


Figure 10. Domain adaptation example. Source domain: Buxtehude. Target domain: Nienburg. From left to right: input image patch, ground truth, using Random strategy, using Ground Truth strategy, using our Approximation strategy. Despite domain adaptation, there are still large regions which are misclassified. Our approach mitigates this issue partially, e.g., w.r.t. the class clutter (yellow).

	Approx.	**GT**	**Shuffled GT**
imp. surf.	.029	.072	.043
building	.061	.098	.070
low veg.	.025	.045	.033
tree	.059	.072	.021
car	.020	.046	.028
clutter	.018	-.033	.097

Table 4. Per-class IoU improvements for different mapping strategies compared against the random mapping strategy (DANN-like domain adaptation from Hannover to Vaihingen)

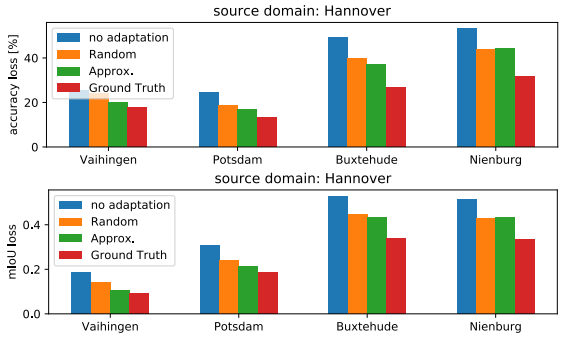


Figure 11. Accuracy and mIoU loss of DANN-like domain adaptation with Hannover as a source domain

strategies bring performance improvements for all classes in almost all cases. In the clutter case for the GT strategy, the recall actually decreases for that class, causing a decrease in IoU.

We performed DANN-like domain adaptation for all 20 possible source and target domain pairs using the strategies Random, GT and Approximation. Our Approximation strategy achieved a mean $SemSim$ score of .447 (std. $\sigma = .212$), lagging behind the Ground Truth strategy with a mean score of .667 ($\sigma = .13$) but being far ahead of the Random strategy which reached .24 ($\sigma = .125$). Examples of the accuracy and mIoU loss, compared to a model trained using the target domain ground truth, are shown in Fig. 11. Domain adaptation in general already mitigates some of the loss, with our Approximation strategy usually mitigating the loss even further. Sample images of actual segmentations are shown in Fig. 10.

Over all the 20 possible domain pairs, the GT strategy improves accuracy and mIoU in 19 instances when compared to the Random strategy. When there is an improvement, the average improvement is 4.2% accuracy and .036 mIoU. The Approximation strategy, again compared to the Random strategy, improves accuracy in 11 instances (average improvement 2.5%) and mIoU in 13 instances (average improvement .017). This proofs that semantically paired image patches improve domain adaptation performance. Our proposed approach already brings a measurable performance improvement while not quite reaching the full potential (GT strategy).

As Cycle-GAN-based domain adaptation is much slower, we only performed six experiments using it. We observed accuracy and mIoU improvements in only three cases. In those cases, the average improvements over the Random strategy were .88% accuracy and .009 mIoU (GT strategy) and .64% accuracy and .003 mIoU (Approximation strategy). These improvements are below the observed standard deviations for each experiment, thus our approach likely has no significant effect on appearance adaptation. However, the GT strategy achieved improvements of 1.7% accuracy and .016 mIoU in a case where Potsdam, which appeared to be a particularly problematic source domain, was used a source domain. Thus, there may still be a benefit for difficult domain adaptation problems. Since modern domain adaptation approaches are based on both, appearance adaptation and feature level adaptation, $SemSim$/$SemSim^*$-based mini-batch composition may still improve the performance of these approaches.

6. CONCLUSION

In this paper we introduced the concept of semantically similar images: two images are semantically similar, if the semantic segmentation of one image is an accurate prediction of the semantic segmentation of the other image. We furthermore proposed a way to estimate the similarity of an image pair if only the segmentation of one of the images is known. Additionally we demonstrated that adversarial domain adaptation methods show improved performance when composing mini-batches s.t. there is a semantically similar target domain image for each source domain image in the mini-batch. Using a DANN-based approach for domain adaptation we saw an improvement in 19 out of 20 instances (average improvement: 4.2% accuracy and .036 mIoU) when computing the similarity using the target domain ground truth segmentation. Using our estimate of the semantic similarity, we saw accuracy improvements in 11 out of 20 instances (average improvement in those instances: 2.5%)

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2020-483-2020 | © Authors 2020. CC BY 4.0 License.

and mIoU improvements in 13 out of 20 instances (average improvement in those instances: .017). Using a Cycle-GAN-based approach instead, we saw no significant improvements aside from one experiment, which used a particularly difficult source domain. This indicates that semantic image pairing helps especially with difficult domain adaptation problems. While the maximum performance increase (i.e., when using target ground truth) offered by semantic pairing is almost always positive, our estimate does not quite achieve this limit yet, however, it already is beneficial on average.

In the future, we want to investigate open questions such as whether the pairing also improves performance of combined approaches, e.g., CyCADA or CrDoCo, and whether having mini-batch class distributions that match for both domains is sufficient already. Another open question is whether the ideal point in time when to stop training a model can be estimated, as a downward trend in performance has been observed when a model is trained for too long. Furthermore, preliminary tests have shown that semantic pairing makes the DANN approach more robust w.r.t. discriminator complexity, which may mean that it improves hyperparameter setting robustness in general.

REFERENCES

Chang, W.-L., Wang, H.-P., Peng, W.-H., Chiu, W.-C., 2019. All about structure: Adapting structural information across domains for boosting semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1900–1909.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H., 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. *Proceedings of the European Conference on Computer Vision (ECCV)*, 801–818.

Chen, M., Xue, H., Cai, D., 2019a. Domain adaptation for semantic segmentation with maximum squares loss. *Proceedings of the IEEE International Conference on Computer Vision*, 2090–2099.

Chen, Y.-C., Lin, Y.-Y., Yang, M.-H., Huang, J.-B., 2019b. Crdoco: Pixel-level domain transfer with cross-domain consistency. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1791–1800.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2016. The cityscapes dataset for semantic urban scene understanding. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3213–3223.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V., 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1), 2096–2030.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2672–2680.

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A. A., Darrell, T., 2017. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*.

Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kuhnke, F., Ostermann, J., 2019. Deep head pose estimation using synthetic images and partial adversarial domain adaption for continuous label spaces. *IEEE International Conference on Computer Vision (ICCV)*.

Long, M., Cao, Y., Cao, Z., Wang, J., Jordan, M. I., 2018. Transferable representation learning with deep adaptation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(12), 3071–3085.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., Paul Smolley, S., 2017. Least squares generative adversarial networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2794–2802.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 9.

Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A. M., 2016. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3234–3243.

Rozantsev, A., Salzmann, M., Fua, P., 2018. Beyond sharing weights for deep domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(4), 801–814.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520.

Taigman, Y., Polyak, A., Wolf, L., 2016. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*.

Tsai, Y.-H., Hung, W.-C., Schulter, S., Sohn, K., Yang, M.-H., Chandraker, M., 2018. Learning to adapt structured output space for semantic segmentation. *CVPR*, 7472–7481.

Tsai, Y.-H., Sohn, K., Schulter, S., Chandraker, M., 2019. Domain adaptation for structured output via discriminative patch representations. *Proceedings of the IEEE International Conference on Computer Vision*, 1456–1465.

Vogt, K., Paul, A., Ostermann, J., Rottensteiner, F., Heipke, C., 2018. Unsupervised source selection for domain adaptation. *Photogrammetric Engineering & Remote Sensing*, 84(5), 249–261.

Vu, T.-H., Jain, H., Bucher, M., Cord, M., Pérez, P., 2019. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2517–2526.

Zeiler, M. D., 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhu, J.-Y., Park, T., Isola, P., Efros, A. A., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2223–2232.

Zou, Y., Yu, Z., Vijaya Kumar, B., Wang, J., 2018. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. *Proceedings of the European Conference on Computer Vision (ECCV)*, 289–305.

APPENDIX

Evaluation Metrics

We use two metrics to evaluate the performance of the models we trained. First, we are using the pixel level accuracy. Given a ground truth segmentation $y : L \rightarrow C$ mapping locations $l \in L$ to classes $c \in C$ and a prediction $\hat{y} : L \times C \rightarrow \mathbb{R}$ of the same segmentation, we first compute $\hat{y}^* : L \rightarrow C$ using the equation

$$\hat{y}^*(l) = \arg\max_{c \in C} \hat{y}(l, c). \tag{4}$$

The accuracy is then defined as

$$Acc(y, \hat{y}^*) = \frac{|\{l \in L \mid y(l) = \hat{y}^*(l)\}|}{|L|}, \tag{5}$$

i.e., the fraction of pixel position whose class has been predicted correctly.

As a second metric we compute the mean intersection-over-union (mIoU). The intersection-over-union (IoU) for a given class $c \in C$ is defined as

$$IoU(y, \hat{y}^*, c) = \frac{|\{l \in L \mid y(l) = c \wedge \hat{y}^*(l) = c\}|}{|\{l \in L \mid y(l) = c \vee \hat{y}^*(l) = c\}|}, \tag{6}$$

i.e., it is the number of pixel positions $l \in L$, which both segmentations assign to class c (intersection), over the number of pixel positions, which at least one segmentation assigns to class c (union). The mIoU is then defined as

$$mIoU(y, \hat{y}^*) = \frac{1}{|C|} \sum_{c \in C} IoU(y, \hat{y}^*, c), \tag{7}$$

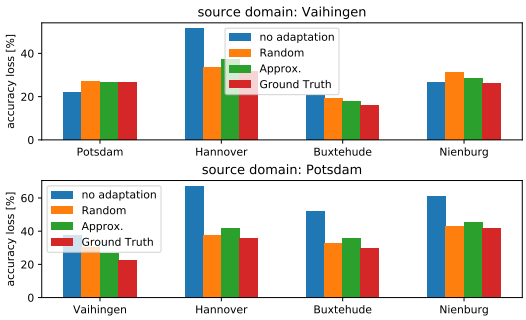
i.e., the mean IoU over all classes $c \in C$.

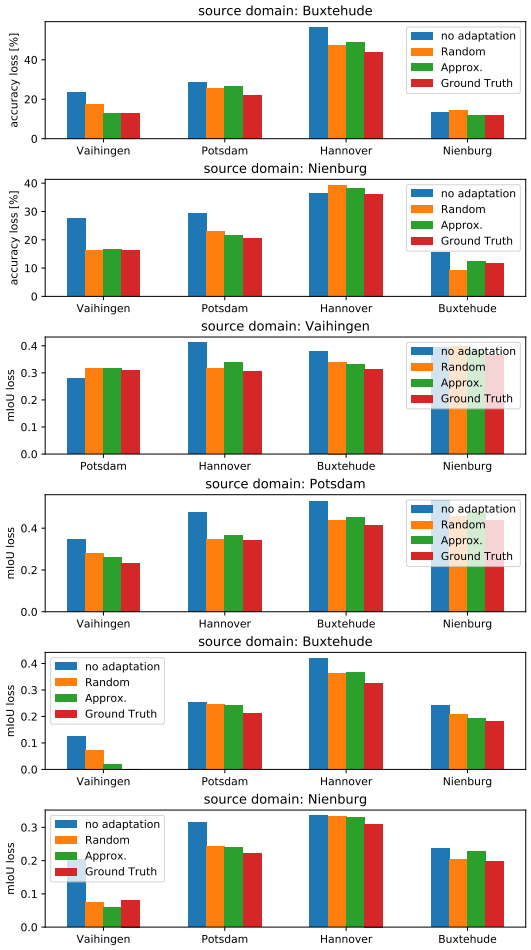
Both metrics assign values in $[0, 1]$ to every pair of segmentations, with higher values meaning that the two segmentations are more alike. This, in turn, means values as close to 1 as possible are desirable as that means that the predicted segmentation is close to the ground truth segmentation.

Note: We do not use eroded boundaries when computing these metrics as opposed to metrics used by some other researchers, e.g., as (partially) used in the benchmark results of the ISPRS Vaihingen 2D Semantic Labeling Test.

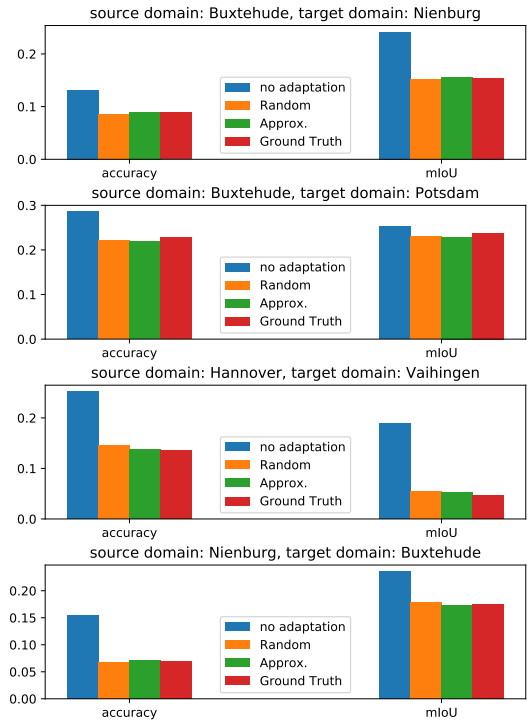
Accuracy and mIoU Loss Details

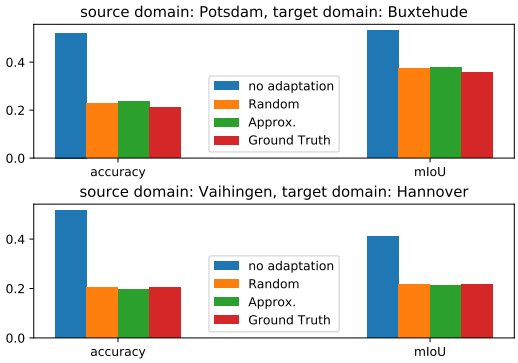
The following graphs show the loss in accuracy and mIoU for different mapping strategies and source domains when using a DANN-like approach for domain adaptation.





The following graphs show the loss in accuracy and mIoU for different mapping strategies, source domains and target domains when using a Cycle-GAN for domain adaptation. *Note:* Accuracies are not normalized to 100% in these graphs.




source domain: Potsdam, target domain: Buxtehude
0.4
0.2
0.0
no adaptation
Random
Approx.
Ground Truth
accuracy
mIoU

source domain: Vaihingen, target domain: Hannover
0.4
0.2
0.0
no adaptation
Random
Approx.
Ground Truth
accuracy
mIoU