# Combining Semantic Self-Supervision and Self-Training for Domain Adaptation in Semantic Segmentation

Joshua Niemeijer[1] and Jörg P. Schäfer[2]

***Abstract*— This work presents a two-staged, unsupervised domain adaptation process for semantic segmentation models by combining a self-training and self-supervision strategy. Self-training (i. e., training a model on self-inferred *pseudo-labels*) yields competitive results for domain adaptation in recent research. However, self-training depends on high-quality pseudo-labels. On the other hand, self-supervision trains the model on a surrogate task and improves its performance on the target domain without further prerequisites.**

**Therefore, our approach improves the model's performance on the target domain with a novel surrogate task. To that, we continuously determine class centroids of the feature representations in the network's pre-logit layer on the source domain. Our surrogate task clusters the pre-logit feature representations on the target domain regarding these class centroids during both training stages. After the first stage, the resulting model delivers improved pseudo-labels for the additional self-training in the second stage. We evaluate our method on two different domain adaptions, a real-world domain change from Cityscapes to the Berkeley Deep Drive dataset and a synthetic to real-world domain change from GTA5 to the Cityscapes dataset. For the real-world domain change, the evaluation shows a significant improvement of the model from $46\%$ mIoU to $54\%$ mIoU on the target domain. For the synthetic to real-world domain change, we achieve an improvement from $38.8\%$ to $46.42\%$ on the real-world target domain.**

## I. INTRODUCTION

Autonomous vehicles require detailed information about the environment. In this context, image processing is of great importance. A semantic image segmentation provides a pixel-wise classification and hence creates detailed information about object and surface classes.

Training a network on a specific data domain (*source domain*) yields a model with good accuracy on that same domain. For example, a model trained on the Cityscapes dataset [3] performs well on the corresponding test data because it has the same properties as the training data regarding the sensor configuration and the perceived environment. Applying that model to a different data domain (*target domain*) with different properties results in a significant drop in accuracy. Performing this *domain adaptation* by manually creating new labels for the target domain and including these labels in the training process reduces the gap of performance between the source and the target domain. This approach, however, is expensive and thus not feasible, especially for the task of semantic segmentation.

[1]Joshua Niemeijer is with German Aerosapce Center (DLR) Joshua.Niemeijer@dlr.de

[2] Jörg P. Schäfer is with German Aerosapce Center (DLR) Joerg.Schaefer@dlr.de

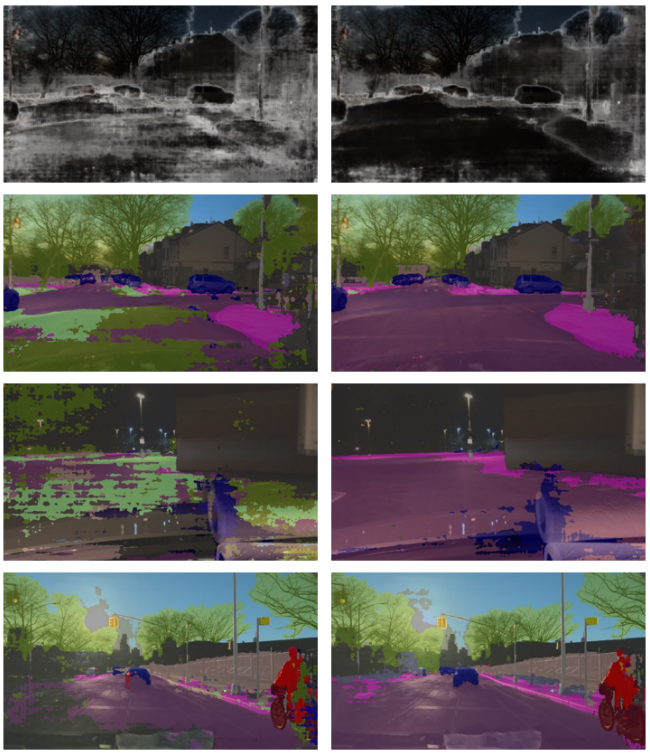

Fig. 1. The first and second column of images show the results before and after adaption, respectively. The adaptation process shows benefits on difficult conditions (e. g., glare, and on different road surfaces) and in the domain shift day to night. The top row shows the network's uncertainties, which is much smaller after the domain adaptation.

Several approaches to reduce the manual work for domain adaptation exist: Active learning approaches reduce the manual labeling overhead by recommending a subset of the target dataset that promises the most effective training results [13]. Unsupervised domain adaptation approaches even avoid manual labeling overhead altogether (cf. Section II).

In general, unsupervised domain adaptation methods aim to align the data distributions on the source domain and the target domain [1, 7, 15]. This alignment either happens in the input space (the images) or the neural network's feature space.

This work contributes a new domain adaptation approach for combining self-supervision and self-training approaches (cf. Figure 2). In the first stage, our approach aligns the distributions of the source and target domain in the feature space via a self-supervision strategy that ties up the clusters of the feature representations, each of which represents one class. The resulting model generates pseudo-labels with

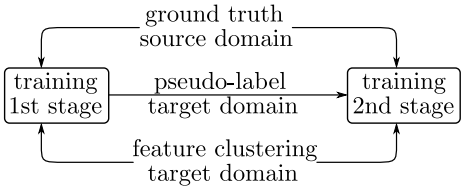


Fig. 2. Our two-staged training process.

improved quality on the target domain data. Additionally to the feature clustering and training on the source domain, the second stage trains the model on the target domain with the newly generated pseudo-labels.

Our evaluation addresses two important domain changes, a real-world domain change, and a synthetic to real-world domain change. We further show that the self-supervision is crucial for an effective self-training in the second stage. For the more important real-world domain change, the source domain data consists of very stable conditions, while the target domain data contains a large variety of weather, lighting, and sensor configurations. We also study our method's performance on the synthetic to real-world domain change.

## II. Related work

Most unsupervised domain adaptation approaches aim to align the distributions of the source and target domain. This alignment is either performed in the input space or the feature space of the neural network.

Some approaches for the former use style transfer to transform the source domain images to have similar textures as the target domain images [7, 9, 18].

This work, however, concentrates on the latter approaches, i. e., we focus on aligning the distributions of the features of the source and target domain. The following sections describe recent approaches of distribution alignment in the feature space using self-training, adverserial training, self-supervision, contrastive self-supervision, and semantic self-supervision.

### A. Self-Training

Recent self-training approaches achieved great success in domain adaptation [20, 21]. They use a model trained on the source domain to annotate images from the target domain. With these so-called *pseudo-labels*, the model further trains on the source domain and the target domain simultaneously. The challenge with self-training is mitigating the negative influence of wrongly inferred pseudo-labels.

Pan Zhang et al. manipulate the entries of the prediction vector on the target domain by weighting each class's probability with the similarity of its feature representation to the class's feature cluster centroid; thus, predictions on the target domain that are further away from the centroid are considered to be unlikely, and therefore weighted less in the training [20]. The proposed self-training presented in [21] alternates between selecting a subset of the most confident pseudo-labels on the target domain regarding the current model and training on the source and target subset data, i. e., after each update of the model, they choose another subset of the target domain for the training. Pan et. al. introduce the definition of inter- and intra-domain gaps, i. e., gaps between two domains and between easy and hard images within one domain, respectively [10]. They combine the self-training with the self-supervision strategy presented in [16] (cf. Section II-C).

The self-training approaches mentioned above apply to our work as well. However, we decided to implement a simple self-training strategy since we focus on the self-supervision methodology.

### B. Adverserial Training

Adversarial training is based on two networks, a feature extractor network and a domain discriminator network, classifying the feature space into source and target domain [1, 16]. The optimization goal is to generate a feature space that is not discriminable into source and target domain whilst containing relevant representations for the semantic segmentation. This approach, however, targets the problem of unsupervised learning from a different angle and is therefore not considered further in this work.

### C. Self-Supervision

Self-supervision approaches for domain adaptation rely on *pretext tasks* [6, 8, 15]. Pretext tasks require the prediction of annotations automatically generated on both the source and target domain. For example, rotating or flipping an image yields a pretext task that requires predicting the image's state regarding rotation and flipping. Well-chosen pretext tasks require similar features as the main task (i. e., semantic segmentation in this case), and therefore implicitly achieve a distribution alignment between the domains in the feature space. When the distributions of the source and target domains are aligned, a classifier trained on the source domain generalizes well to the target domain. The work presented in [15] uses this approach for domain adaptation in semantic segmentation. In contrast to the pretext tasks considered in [6, 8, 15], this work studies the clustering of feature representations.

Although not considered in this work, it is worth investigating the combination and synergies of contrastive self-supervision with our method.

### D. Contrastive Self-Supervision

Contrastive self-supervision methods consider pairs of samples of the same class *(positive pairs)* or different classes *(negative pairs)*. The training then aims to minimize the difference in the feature space between positive pairs and maximize the difference of negative pairs. Especially in image classification, this strategy has shown great success for pre-training [5].

A problem that arises for contrastive self-supervision is the trade-off between spatial invariance that is beneficial for classification and spatial sensitivity that is important for localization tasks like segmentation and detection. The method presented in [4] addresses this problem with a patch-wise contrastive self-supervision task.

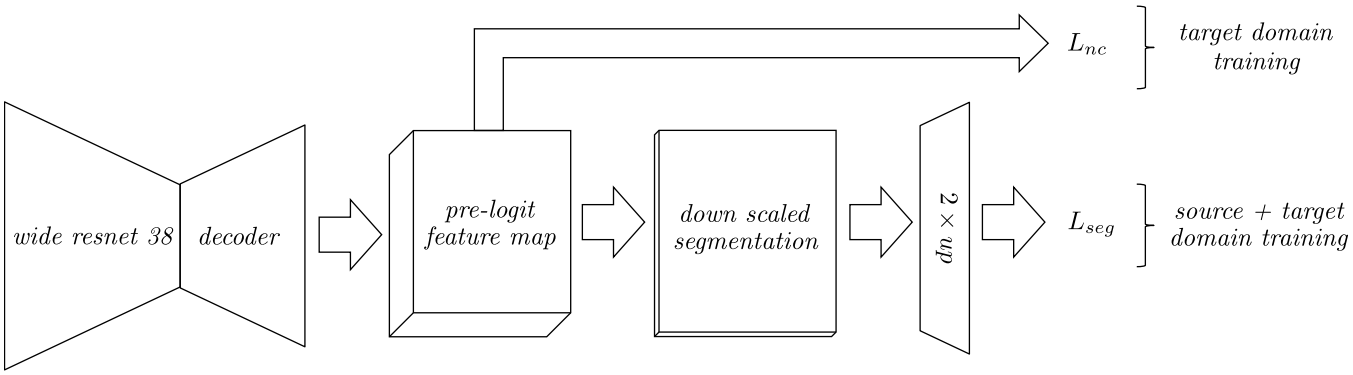


Fig. 3. This figure shows the integration of the domain adaptation into DeepLabV3+ [2]. The semantic segmentation is trained in a supervised manner on the source domain. The semantic clustering is applied on the target domain data. The self-supervised clustering is performed on the pre-logit feature map that is $1/4$ the size of the original image.

Domain adaptation problems cannot use target domain labels, making it challenging to find class-based positive and negative pairs between the source and target domain. For positive pairs, Shim and Kim address this problem with style transfer, i. e., they transform an image from the source domain to the style of the target domain, which yields a positive pair [14]. They sample negative pairs, however, from the same domain only.

### *E. Semantic Self-Supervision*

The goal of semantic self-supervision is a correct clustering of the pre-logit[1] feature space, i. e., feature representations that contribute to the same class should be in the same (class's) cluster [12]. Now, the assumption is that a classifier trained on the source domain generalizes well to the target domain when correct clustering was achieved.

Because there are no labels in the target domain, a key problem of semantic self-supervision is that it is unclear which cluster a feature representation belongs to. Saito et al. propose the computation of a similarity matrix from the feature representations in the target domain to the class's cluster centroids in the source domain [12]. Their objective is to minimize the distance of each feature representation to its nearest cluster centroid, respectively. This approach assumes that each feature representation in the target domain is usually closest to its correct cluster centroid. The challenge is to mitigate the influence of feature representations for which this assumption does not hold. To that, Saito et al. minimize the distance to the cluster centroids and a memory bank of target representations simultaneously. Their approach, however, was developed for classification and, as we experienced, is not feasible for semantic segmentation since in this case, the memory bank does not fit into memory.

Semantic self-supervision approaches bear a certain similarity to approaches that aim at minimizing the entropy of the class prediction vector, and hence increasing the probability of the class with the highest confidence even further [16]: Decreasing the entropy of the class prediction creates tighter clusters of the feature representations. In contrast to this approach, we aim at clustering the pre-logit feature space.

[1]The pre-logit layer precedes the classification layer of the network.

## III. Semantic clustering

As already suggested in Section II, we define *semantic self-supervision* as clustering the pre-logit feature space representations to *class prototypes*, i. e., to feature vectors representing the classes. To achieve such a clustering, a large amount of feature representations is required. Saito et al. define class prototypes as weight vectors $w = [w_1, w_2, w_3, ..., w_K]$ of the classification neurons for each particular class, respectively [12]. They normalize the elements of $w$, and they assume that the final classifier has no bias term for defining the classifier hyperplanes, both of which are necessary for the cosine similarity used for clustering.

However, the approach proposed by Saito et al. not only clusters the pre-logit feature representations towards class prototypes but additionally to the pre-logit feature representations of the target domain samples themselves. In other words, this approach aims to move feature representations towards known class prototype *and* neighboring pre-logit representations of the target domain.

Saito et al. implement their approach using a memory bank $F$ of feature representations and weights defined as the tuple

$$F = [f_1, \cdots, f_{N_t}, w_1, \cdots, w_K]$$

where $K$ represents the number of classes. They propose to keep track of these feature representations, such that it represents the network's current and past state in each iteration[2]. In contrast, we do not keep track of the history but only consider the pre-logit feature representations of the current target domain sample. In the case of a semantic segmentation architecture like the one shown in Figure 3, the number of pre-logit feature representations $N_t$ depends on the spatial size of the image. For example, the DeepLabV3+ architecture yields $N_t = {}^{width \times height}/_{4}$. Hence, keeping

[2]More specifically, Saito et al. keep track of a finite subset of past feature representations.

track of the past feature representations is not feasible due to memory limitations.

Just like the elements of $w$, each entry $f_i$ is $L_2$ normalized. While the tuple $F$ concatenates both the memory bank and the class prototypes, the tuple

$$f = [f_1, \cdots, f_{N_t}]$$

contains only the feature representations $f_i$. Now, consider the weighted similarity matrix

$$p_{i,j} = \frac{exp(F_j^T f_i/\tau)}{Z_i}$$

between the entries of $F$ and $f$ using the cosine similarity, where the temperature parameter $\tau$ controls the distribution concentration degree. Again, due to memory limitations, we sample a subset of $1\%$ of $f$ to obtain the target domain representations $f_i$ we want to cluster. The factor

$$Z_i = \sum_{j=1, j\neq i}^{N_t+K} exp(F_j^T f_i/\tau)$$

normalizes the similarity values analogously to a soft-max function; thus, the sum of all $p_j$ equals 1. Note that we discard self-similarities (i. e., the similarities of every $f_i$ to itself) since they destroy the clustering to the nearest neighbor but instead yield a trivial solution, respectively. Finally, the loss function

$$L_{nc} = -\frac{1}{|B_t|} \sum_{i\in B_t} \sum_{j=1, j\neq i}^{N_t+K} p_{i,j} log(p_{i,j})$$

is the entropy of the similarity matrix where $|B_t|$ is the number of clustered feature representations. Minimizing the entropy enforces each point to either move closer to a pre-logit representation or a class prototype.

### A. Our Method

Our approach for clustering is inspired by the method presented in [12] and described above. However, we do not represent the class prototypes using the normalized weights of the classification neurons. We argue that these *classification normals* are strongly effected by outliers, which causes the normals to fluctuate a lot during the training process. Clustering towards a noisy cluster center makes it hard for the algorithm to converge. Eventually, we obtain misrepresented centroids of the class representations in the pre-logit layer. Instead, we propose to compute the centroids in a probabilistic way as follows.

In every training step, we first compute the pre-logit representations $f_s$ of the batch of source domain images $B_s$. Since we implemented our approach on the semantic segmentation architecture shown in Figure 3, every image yields a number of $width \times height/4$ pre-logit representations $f_s$ (cf. [2]). We partition the feature representations regarding the classification results of the model to compute the expected values $\mathbf{E}_K(f_s)$ for each of the $K$ classes as shown in Figure 4 (left). Now, each expected value $\mathbf{E}_k$ is a centroid for the cluster of class $k$, respectively. Why not use the ground truth instead of the classification results? Representations of wrongly classified pixels are within another centroids area instead of being close to their class prototype. Therefore, they corrupt their centroid's position by being an outlier.

Since the current Batch $B_s$ only represents a subset of the entire dataset, we iteratively compute a running average

$$c_k = c_k * \alpha + \mathbf{E}_k * (1 - \alpha)$$

of the class centroids. Intuitively, $\alpha$ affects the accuracy of seldom classes. Section IV discusses how to choose this hyper parameter $\alpha$.

Given the updated centroids $c_k$, we apply the clustering method presented in [12] for the samples in the target-domain Batch $B_t$. As suggested there, we first perform an $L_2$ normalization on the class centroids $c_k$ and the target-domain pre-logit feature representations $f^t$ that result from the images present in $B_t$. Hence, the similarity matrix $p_{i,j}$ for every $f_i \in f^t$ and class centroid $c_j$ is

$$p_{i,j} = \frac{exp(c_j^T f_i^t/\tau)}{Z_i}.$$

We set the neighborhood parameter $\tau$ to $0.05$ according to the best practice in [12]. Since $c_j$ and $f_i^t$ are $L_2$ normalized, computing the dot product yields the cosine similarity as illustrated in Figure 4 (center). The soft-max normalization parameter $Z_i$ is

$$Z_i = \sum_{j=1}^{K} exp(c_j^T f_i^t/\tau).$$

Finally, the loss function computes the overall entropy of all elements $p_{i,j}$:

$$L_{nc} = -\frac{1}{|B_t|} \sum_{i\in B_t} \sum_{j=1}^{K} p_{i,j} log(p_{i,j})$$

Figure 4 (right) helps to understand the effect of minimizing this loss function: Intuitively, each element of $f^t$ moves closer to the centroid that is most similar in terms of the cosine similarity computed in $p_{i,j}$.

### B. Uncertainty Weights

The loss function described above implicitly assumes that target samples are closer to the centroid of their respective ground truth class than to the others[3]. Since this assumption does not hold in every case, we extend the original method with a measure of uncertainty. A standard measure for the uncertainty of the network predictions is the entropy of the class confidences. We use this measure to weigh the loss function

$$L_{nc} = -\frac{1}{|B_t|} \sum_{i\in B_t} \sum_{j=1}^{K} p_{i,j} log(p_{i,j}) \cdot \frac{1}{1 + H_i},$$

where $H_i$ is the class confidences' entropy with respect to each $f_i$. The weight factor $1/1+H_i$ mitigates the influence of target samples that yield a high classification uncertainty.

[3]This work always considers similarity in terms of the cosine similarity.

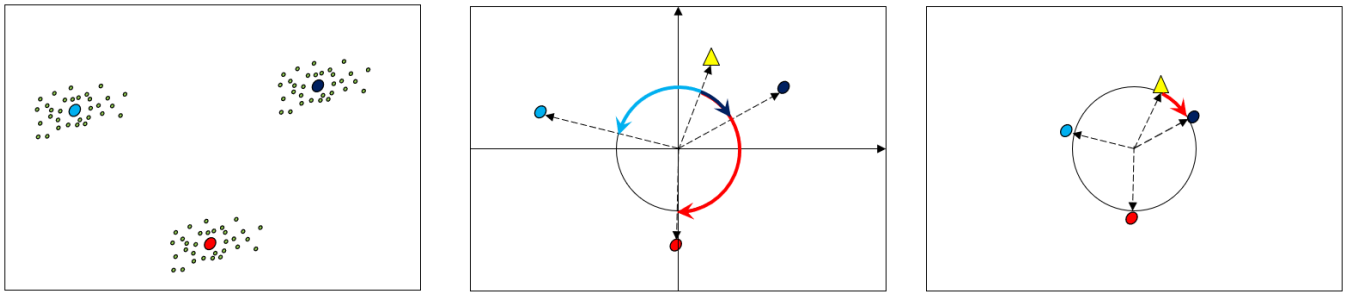

Fig. 4. This figure shows feature representations and class centroids of a source domain batch (left), the normalized similarity of a target feature representation (yellow triangle) to all centroids (center), and the effect of minimizing the entropy (cf. Section III-A), which causes the target representation to move closer to its closest centroid (right).

Our intuition is that the assumption above is more likely not to hold when this uncertainty is larger. Unfortunately, by introducing the entropy as a weighting factor, the loss function can now be minimized by increasing the term $H_i$. We address this problem by scaling down the gradients produced by this objective with a factor of $0.01$; thus, the objective to cluster the pre-logit representations around the most similar centroid dominates. We explicitly do not choose to zero this factor since the objective acts as an additional counter loss for those cases where the maximum similarity to a centroid is low.

### C. Training Schedule

Figure 3 helps to understand the training strategy. We train the semantic segmentation based on the pixel-wise cross-entropy loss function $L_{seg}$ on the source domain. Our DeepLabV3+ implementation performs the pixel-wise classification on a feature map $1/4$ of the size of the original image and subsequently up-scales the result by a bi-linear interpolation. Hence, the pre-logit feature space we want to cluster is $1/4$ of the size of the original image. We perform the clustering (and with it the computation of the loss function $L_{nc}$) on the target domain only. Our final loss function is

$$L = L_{seg} + \lambda L_{nc}$$

where $\lambda$ is a weight parameter that we empirically found $0.02$ to be a good value. Since the semantic self-supervision method requires a tremendous amount of memory, we first update the network according to $L_{nc}$ and subsequently $L_{seg}$. As shown in [15], this separate update scheme only has a limited negative influence on the network's performance.

### D. Self-Training

Multiple approaches for self-training exist that achieve competitive performance. Approaches like in [20] and [21] introduce methods for estimating the probability that a pseudo-label represents the correct class. We did not apply such methods because our main focus in this work is to show that our self-supervision method improves the initial quality of the pseudo-labels and the self-training process itself.

In the first stage, we adopt a semantic segmentation model to the new domain using supervised-training on the source domain and semantic self-supervised training, simultaneously (cf. Figure 2). In the second stage, we use the new model to compute the entropy of the pixel-wise class predictions $P_{i,j}$, serving as an uncertainty measure of the predictions.

$$H(P_i) = -1 \cdot \sum_{j=1}^{K} P_{i,j} log(P_{i,j})$$

We apply a threshold of $log(K)/4$ to this mask to filter out labels with high uncertainty, i. e., the training ignores all pixels exceeding this threshold. Half of the batch is chosen for the training from the source domain and half from the pseudo-labeled target domain.

We apply our self-supervision method in parallel to the self-training described above. We argue that the target and source domain distributions of the pre-logit feature space are closer aligned with self-training applied. That, in turn, inherently causes the centroids we determined on the source domain to better represent the target domain. Hence, it is more likely that the target-domain pre-logit representations are most similar to their ground truth class centroids.

## IV. Experiments

### A. Experimental Details

We study two different adaptation scenarios in our evaluation, i. e., a synthetic to real-world and a real-world to real-world adaptation.

In the first scenario, we adapt from the synthetic GTA5 [11] to the real-world Cityscapes dataset [3], both of which contain the same set of labels. The lack of variety in geometry, texture, and weather or lighting conditions in the simulated environment makes this delta especially challenging. However, we mostly pay attention to the real-world to real-world domain adaptation. In particular, we consider a delta from a real-world dataset recorded under constant conditions regarding the environment, weather, and the camera sensor to a real-world dataset consisting of various environmental and weather conditions and recorded with different cameras and camera positions. This delta is represented by the adaptation from the Cityscapes to the Berkeley Deep Drive dataset (BDD) [19], both of which contain the same set of labels.

TABLE I
CITYSCAPES TO BDD

| | road | sidewalk | building | wall | fence | pole | traffic light | traffic sign | vegetation | terrain | sky | person | rider | car | truck | bus | train | motorbike | bicycle | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| source | 88.56 | 49.25 | 70.57 | 10.24 | 22.61 | 37.86 | 41.96 | 42.75 | 73.29 | 36.52 | 89.59 | 58.36 | 34.32 | 84.28 | 24.27 | 26.35 | 0.01 | 46.51 | 41.65 | 46.26 |
| memory bank | 84.75 | 39.51 | 74.64 | 14.26 | 17.93 | 33.37 | 31.66 | 34.61 | 76.98 | 33.76 | 87.85 | 51.77 | 26.56 | 75.94 | 17.02 | 41.77 | 0.0 | 26.43 | 21.42 | 41.59 |
| prototype | 89.96 | 48.27 | 76.68 | 17.86 | 26.74 | 41.44 | 42.91 | 44.17 | 80.31 | 26.74 | 90.05 | 58.69 | 41.37 | 84.53 | 22.87 | 37.95 | 1.01 | 46.4 | 37.28 | 48.17 |
| semantic $\alpha 0.1$ | 90.21 | 49.75 | 74.45 | 14.65 | 24.68 | 37.61 | 42.91 | 46.17 | 79.02 | 31.07 | 90.27 | 61.11 | 48.97 | 85.1 | 31.08 | 30.64 | 0.06 | 55.31 | 53.89 | 49.84 |
| semantic $\alpha 0.9918$ | 90.97 | 52.21 | **78.48** | 17.87 | 30.34 | 42.7 | 45.07 | 44.04 | 82.6 | 37.79 | 90.81 | 56.93 | 36.39 | 81.55 | 30.54 | 47.12 | 0.1 | 38.79 | 32.4 | 49.3 |
| semantic $\alpha 0.9$ | 90.85 | 51.44 | 77.04 | 14.89 | 25.98 | 38.63 | 43.63 | 46.19 | 81.14 | 36.72 | 90.65 | 61.34 | 48.27 | 84.63 | 31.39 | 49.29 | 0.6 | **57.31** | 49.83 | 51.54 |
| self-training | 92.13 | 55.33 | 78.28 | **18.45** | **34.53** | 42.44 | 45.64 | 46.1 | 81.68 | 40.69 | 91.13 | 61.54 | 47.13 | 87.13 | 37.1 | 43.73 | 0.01 | 51.46 | 44.38 | 52.58 |
| 1. semantic self-training | **93.06** | **57.01** | 77.98 | 17.85 | 34.43 | **42.76** | **48.37** | 47.48 | **82.25** | **41.17** | 91.69 | 63.66 | **54.02** | 87.01 | **38.08** | 50.73 | 0.08 | 52.69 | 49.45 | 54.2 |
| 2. semantic self-training | 92.37 | 56.54 | 78.12 | 15.02 | 33.26 | 41.95 | 48.06 | **48.53** | 81.64 | 40.69 | **91.72** | **64.25** | 52.58 | **87.48** | 37.76 | **56.65** | **1.79** | 55.99 | **55.11** | **54.71** |
| Oracle | 94.24 | 61.45 | 84.54 | 31.13 | 49.47 | 50.49 | 54.92 | 51.06 | 86.17 | 47.48 | 94.54 | 66.0 | 32.51 | 89.33 | 47.9 | 74.41 | 0.026 | 53.64 | 45.86 | 58.72 |

All experiments performed for this paper are based on a particular implementation of DeepLabV3+[4] with a WideResNet38 [17] backbone. In order to achieve a class-balanced training, the implementation employs a uniform sampling over the classes for 50% of the source domain images in a training epoch. We scale the images of the source domain to obtain equally sized images as in the target domain. Then, we crop these images randomly to a size of $400 \times 400$ pixels. Every batch consists of 24 source and, when a domain adaptation strategy is applied, 24 target images. We trained the neural net with a stochastic gradient descent optimizer for 180 and 45 epochs in the real-world to real-world and synthetic to real-world scenario, respectively. The initial learning rate is $0.007$ and decays each epoch according to $1-{epoch}/{epoch_{max}}$. We employ a regularizing weight decay of $1e-4$ and a momentum to the optimizer of $0.9$. In contrast to the training, the evaluation uses the images in their original size.

### B. Adaptation of the Related Work to Semantic Segmentation

In addition to the method presented in Section III-A, we performed experiments implementing the method introduced in [12] and adapted it for semantic segmentation. We evaluated its performance on the domain change from Cityscapes to BDD as shown in Table III. Building the memory bank based on the current sample's feature representations only (as described in III) leads to worse results than even achieved with a source-only training, which illustrates the necessity of additionally holding feature representations of other samples in the memory bank.

By leaving out the target feature representations and only putting the class prototypes into the memory bank, we improved the source-only training of $1.91\%$ in terms of mIoU and $15.33\%$ relative to the gap ($12.46\%$) between source-only and source+target training. We conclude that an improved way of constructing the memory bank will positively influence the overall performance since, as described in [12], a structured feature space is more discriminable.

### C. Fine-Tuning the Centroid Updates

The parameter $\alpha$, introduced in Section III-A, controls the update of the centroids representing the classes. As a value between $0$ and $1$, it weights the old state of the centroids while $1-\alpha$ weights the update. We tested different values for $\alpha$, as shown in Table I. Our empirical study shows that $\alpha = 0.9$ promises the best results. We assume that a low value like $0.1$ causes the centroids to adapt strongly to the updates, which introduces noise into the clustering process, and therefore makes it hard for the optimization process to converge. If $\alpha$, on the other hand, is chosen too high (e. g., $\alpha = 0.9918$), then the centroids do not represent the current state of the network, in which case the data is clustered to the wrong centroids. Hence, a value of $\alpha = 0.9$ seems to be a good trade-off.

### D. Real-World Domain Change

Table I shows the results when adapting from the real-world Cityscapes dataset to the real-world Berkeley Deep Drive dataset (BDD). The challenge, in this case, lies within the variety of weather conditions (rain, snow, sunny), lighting conditions (normal sunlight, direct sunlight causing glaring, twilight, and night), camera types, camera positions, and locations (rural, urban) of the BDD data. On the other hand, the Cityscapes dataset only consists of images from urban parts of Germany recorded under constant weather conditions with a constant camera setup.

These deltas between the datasets cause the performance of a model trained on the Cityscapes dataset to drop when evaluated on the validation set of the BDD dataset. Table I shows that the mIoU value decreases from $58.72\%$ when trained on both datasets to $46.26\%$ when trained on the source domain only. In terms of absolute mIoU values, this is a difference of $12.46\%$. For example, Figure 1 shows that the source-only model has poor quality on twilight and night images. Applying our self-supervision method presented in Section III-A eliminates $43.06\%$ of this gap, i. e., the method achieves an *mIoU* value of $51.54\%$. These improvements are visible in Figure 1. Based on these results, we assume that our method outperforms the source-only training for the deltas introduced by glaring, twilight, and nighttime in general. The uncertainty maps (i. e., the entropy of the class predictions per pixel) in the top row of Figure 1 show that our method reduces the overall uncertainty as expected with clustering to the class centroids. The improved results and the more interpretable uncertainty maps are, in turn, a good state for the generation of high-quality pseudo-labels.

[4]We used the implementation available at the following URL: `https://github.com/NVIDIA/semantic-segmentation/tree/sdcnet`

TABLE II
GTA5 TO CITYSCAPES

| | road | sidewalk | building | wall | fence | pole | traffic light | traffic sign | vegetation | terrain | sky | person | rider | car | truck | bus | train | motorbike | bicycle | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| source | 58.39 | 25.44 | 68.01 | 25.55 | 26.77 | 40.25 | 44.62 | 19.32 | 84.37 | 30.78 | 56.56 | 69.17 | 36.64 | 74.29 | 24.13 | 10.86 | 0.9 | 29.34 | 21.07 | 38.8 |
| Pan et. al. [10] | **90.6** | 37.1 | **82.6** | 30.1 | 19.1 | 29.5 | 32.4 | 20.6 | 85.7 | **40.5** | 79.7 | 58.7 | 31.1 | 86.3 | **31.5** | **48.3** | 0.0 | **30.2** | **35.8** | 46.3 |
| semantic $\alpha$0.9 | 72.18 | 30.79 | 75.58 | 26.43 | 21.25 | 37.16 | 37.38 | 18.72 | 85.36 | 30.93 | 77.44 | 64.5 | 27.86 | 85.65 | 28.63 | 27.68 | **5.54** | 16.82 | 14.9 | 42.34 |
| 2. semantic self-training | 82.45 | **43.94** | 76.42 | **31.68** | **24.74** | **45.24** | **45.6** | **22.46** | **87.09** | 30.91 | **82.63** | **71.04** | **41.77** | **86.52** | 28.02 | 27.7 | 0.01 | 25.54 | 27.26 | **46.42** |
| target | 98.01 | 84.41 | 92.07 | 49.66 | 59.69 | 64.43 | 68.76 | 78.22 | 92.36 | 63.49 | 94.3 | 82.17 | 62.3 | 94.82 | 80.36 | 85.76 | 79.74 | 65.99 | 76.93 | 77.55 |

We generated pseudo-labels for the training split of the BDD with the best model obtained from the adaptation through self-supervision. We aim to find a reliable surrogate metric to estimate the model performance without using a labeled validation set in future work. E. g., in our experience, the class accuracy on the source domain validation set indicates the target domain mIoU quiet well. We discover the performance of the best possible configuration by training with pseudo-labels generated with the best model selected from the validation set.

As described in Section III-D, we evaluate two configurations of self-training for the second training stage. Table I shows that the mIoU on the target domain increases by $1.04\%$ with simple self-training applied. The additional introduction of our self-supervision loss into the optimization process gains the improvement even further. Expressed in absolute terms, we improve the mIoU by $3.17\%$ in comparison to the self-supervised training; and by $2.13\%$ in comparison to the simple self-training. We close $64.45\%$ of the original gap of $12.46\%$ mIoU. Our two methods (1. and 2. semantic self-training) achieve the best results in 13 out of 19 classes and are even better than the combined training on the source and target domain for the classes *train*, *bicycle*, *rider*, and *motorbike*.

We decide between the two methods based on the following reason. As shown in Table I, we evaluate two versions of the self-supervision assisted self-training, the first of which applies the uncertainty weights introduced in Section III-B (cf. Table I, 2. semantic self-training), and the second of which does not use these weights (cf. Table I, 1. semantic self-training). We observe a gain of $0.51\%$ in terms of mIoU with the usage of the uncertainty weights. This indicates that introducing uncertainty weights can mitigate the influence of cases where the pre-logit target domain representations are clustered to the wrong centroids. We even achieved slightly better results than the method presented in [10] which is close to our method in some aspects.[5] However, there still is room for improvement.

### E. Synthetic to Real-world Domain Change

We developed our methods on the real-world domain change described in the previous section. For further impressions about the performance, we analyzed the resulting best models on the synthetic to real-world domain change (i. e., from GTA5 to Cityscapes). However, we did no special fine-tuning of the hyperparameters to achieve the results shown in Table II.

The source-only training achieves an mIoU of $38.8\%$ which is $38.7\%$ lower than training on the target domain only (which yields $77.55\%$). The magnitude of the performance gap indicates a big domain gap between the simulated data and real-world data. Applying our self-supervision method (with an $\alpha$ value of $0.9$) for domain adaptation as described in Section III-A improves the performance on the target domain by $3.45\%$ mIoU. Again, as described in the Section IV-D, we choose this model to generate pseudo-labels for the self-training on the Cityscapes dataset. For the self-training, we again choose the best configuration obtained from the experiments on the real-world to real-world domain change. We hence apply the self-supervision method with the uncertainty weights and therefore gain an improvement of $4.08\%$ mIoU over the mere self-supervised training and an improvement of $7.62\%$ over the source-only training.

## V. CONCLUSION

We have developed a semantic self-supervision method for domain adaptation in semantic segmentation. We view self-supervision as an important building block for the more general task of domain adaptation. As shown in this work, our self-supervision method can be applied in parallel to other domain adaptation approaches like self-training.

We want to investigate further surrogate measures for the target-domain performance. This way, one can select the best model for the target domain without acquiring an expensive validation set. Concerning our method, we want to deal with class imbalance on the target domain since the random sampling of target domain images introduces a bias towards more frequent classes.


## ACKNOWLEDGEMENT

The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "KI Delta Learning – Development of methods and tools for the efficient expansion and transformation of existing AI modules of autonomous vehicles to new domains. The authors would like to thank the consortium for the successful cooperation."


[5] Remark that they configured their experiments differently, i. e., they used different learning rates, network architectures, etc.

## References


[1] Jan-Aike Bolte et al. “Unsupervised Domain Adaptation to Improve Image Segmentation Quality Both in the Source and Target Domain”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2019.

[2] Liang-Chieh Chen et al. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”. In: *CoRR* abs/1802.02611 (2018).

[3] Marius Cordts et al. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *CoRR* abs/1604.01685 (2016).

[4] Jian Ding et al. “Unsupervised Pretraining for Object Detection by Patch Reidentification”. In: *CoRR* abs/2103.04814 (2021).

[5] Wouter Van Gansbeke et al. “Learning To Classify Images Without Labels”. In: *CoRR* abs/2005.12320 (2020).

[6] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. “Unsupervised Representation Learning by Predicting Image Rotations”. In: *CoRR* abs/1803.07728 (2018).

[7] Judy Hoffman et al. “CyCADA: Cycle-Consistent Adversarial Domain Adaptation”. In: *CoRR* abs/1711.03213 (2017).

[8] Longlong Jing and Yingli Tian. “Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[9] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. “Bidirectional Learning for Domain Adaptation of Semantic Segmentation”. In: *CoRR* abs/1904.10620 (2019).

[10] Fei Pan et al. “Unsupervised Intra-Domain Adaptation for Semantic Segmentation Through Self-Supervision”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[11] Stephan R. Richter et al. “Playing for Data: Ground Truth from Computer Games”. In: *European Conference on Computer Vision (ECCV)*. Ed. by Bastian Leibe et al. Vol. 9906. LNCS. Springer International Publishing, 2016, pp. 102–118.

[12] Kuniaki Saito et al. “Universal Domain Adaptation through Self Supervision”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 16282–16292.

[13] Burr Settles. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison, 2009.

[14] Dongseok Shim and H. Jin Kim. “Learning a Domain-Agnostic Visual Representation for Autonomous Driving via Contrastive Loss”. In: *CoRR* abs/2103.05902 (2021).

[15] Yu Sun et al. “Unsupervised Domain Adaptation through Self-Supervision”. In: *CoRR* abs/1909.11825 (2019).

[16] Tuan-Hung Vu et al. “ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[17] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. “Wider or Deeper: Revisiting the ResNet Model for Visual Recognition”. In: *Pattern Recognition* 90 (2019), pp. 119–133. ISSN: 0031-3203.

[18] Yanchao Yang and Stefano Soatto. “FDA: Fourier Domain Adaptation for Semantic Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[19] Fisher Yu et al. “BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling”. In: *CoRR* abs/1805.04687 (2018).

[20] Pan Zhang et al. “Prototypical Pseudo Label Denoising and Target Structure Learning for Domain Adaptive Semantic Segmentation”. In: *CoRR* abs/2101.10979 (2021).

[21] Yang Zou et al. “Unsupervised Domain Adaptation for Semantic Segmentation via Class-Balanced Self-Training”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.