6 Supplementary Meterial

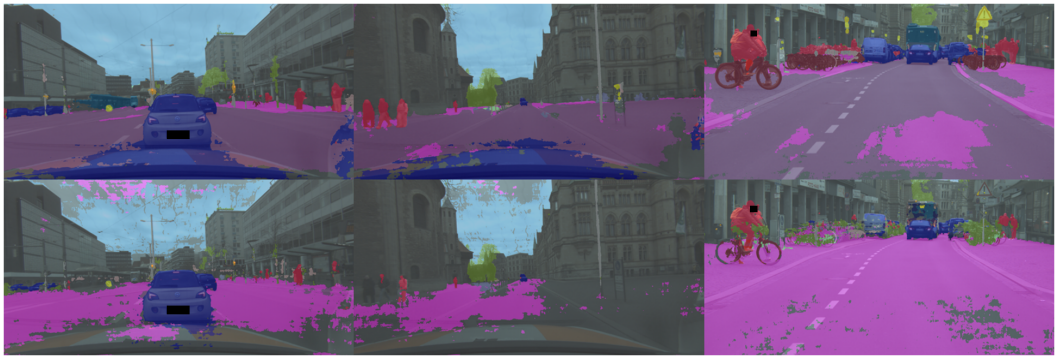

Figure 3: Qualitative evaluation of domain generalization on real data of our autonomous vehicle. Top: Trainig on GTA-5 dataset with adaptation to Cityscapes using EasyAdap; bottom: Trainig on GTA-5 dataset without domain adaptation

6.1 Semantic Clustering

Section 3.1 provides an overview to our self-supervision approach implementing a semantic clustering. In the following, we describe this method in detail.

Computation of class centroids. We partition the feature representations according to their classification in the output layer. Let $\mathbf{E}(f_j^s)$ be the average of the subset $f_j^s \subset f^s$ of feature representations that yield a prediction of the j-th class. In order to estimate the expected value of the class-specific feature representations, we compute a running average from each training batch to the next.

$$c_j \leftarrow c_j \cdot \alpha + \mathbf{E}(f_j^s) \cdot (1 - \alpha) \tag{1}$$

Similarity matrix between source- and target-domain features. To compute the cosine similarity between the class centroids c_j and the feature representations f^t of the target-domain images, we first normalize them regarding the L_2-norm.

$$\bar{c}_j := \frac{c_j}{\|c_j\|} \qquad and \qquad \bar{f}_i^t := \frac{f_i}{\|f_i\|} \quad for \quad f_i \in f^t \tag{2}$$

Our similarity matrix applies a soft-max per target-domain feature on the cosine similarity.

$$p_{i,j} = \frac{exp(\bar{c}_j^T \bar{f}_i^t / \tau)}{Z_i}, \tag{3}$$

where $1 \leq i \leq |f^t|$, $1 \leq j \leq K$, and K is the number of classes.

$$Z_i = \sum_{j=1}^{K} exp(\bar{c}_j^T \bar{f}_i^t / \tau). \tag{4}$$

The parameter τ controls the concentration degree of the clustering process.

Clustering loss function. The basic idea of our loss function is to minimize the entropy of the similarity matrix. Since we applied a soft-max over the columns of the similarity matrix, minimizing this loss function yields matrices that minimize the distance of each particular feature in f^t to exactly one class centroid c_j while maximizing its distance to the other class centroids. Hence, applying this loss function pulls the features in f^t to their nearest class centroid c_j.

Features that are not closest to the correct class centroid disturb the clustering quality. We observed that these features often yield an uncertain classification in the output layer. Hence, to mitigate the influence of wrongly assigned features, we introduce a weight factor in the loss function that scales

with the certainty of its prediction vector: $\mathrm{L}_{nc} = -\frac{1}{|B_t|}\sum_{i=1}^{|B_t|}\sum_{j=1}^{K} p_{i,j} log(p_{i,j}) \cdot \frac{1}{1+H_i}$ where $|B_t|$ is number of target-domain features and H_i is the entropy of the prediction vector corresponding to f_i^t. Note, that H_i must be considered constant regarding the back-propagation. Otherwise, minimizing the loss function could end up maximizing each entropy H_i.

6.2 EasyAdap Algorithm

For Algorithm 1, let K be the number of classes, let N_{adapt} be the number of steps of the adaptation loop, and let N_{train} be the number of epochs for training a model.

Algorithm 1 EasyAdap: a simple yet effective unsupervised domain adaption

```
Input:
    source-domain dataset S
    target-domain dataset T
    initialization model M
M_r ← M
repeat N_train:
    Ŝ ← prepare_sampling(S) # do random and class−uniform sampling
    M ← train_epoch(M, Ŝ) # supervised source−only training
repeat N_adapt:
    T ← create_pseudo_labels(T, M)
    M ← M_r # reinitialize weights
    repeat N_train:
        Ŝ ← prepare_sampling(S)
        T̂ ← prepare_sampling(T)
        for each batch B_s ∪ B_t in Ŝ ∪ T̂:
            B_s ∪ B_t ← augment(B_s ∪ B_t) # cropping uses provided polygon centroid if available
            L_seg ← M(B_s ∪ B_t) # segmentation loss
            # update class centroids on source domain
            f^s ← pre-logit features of M(B_s)
            f^t ← pre-logit features of M(B_t)
            for each 1 ≤ j ≤ K:
                f_j^s ← {f ∈ f^s | M : f ↦ j} # partition features, see Section 6.1
                c_j ← c_j · α + E(f_j^s) · (1 − α) # see Equation 1
            normalize each c_j and f_i ∈ f^t # see Equation 2
            p_{i,j} ← similarity_matrix({c_j}, f^t) # see Equation 3
            L_nc ← clustering_loss((p_{i,j})) # see Equation 6.1
            back_propagate(L_nc + L_seg) and update weights of model M
output M
```

6.3 Further Research Questions and Limitations

Does supervised training on richer datasets yield improved generalization capabilities? Instead of learning or memorizing the data as classical machine learning approaches such as nearest-neighbor classifiers do, deep learning approaches are superior because they find generalizing models for the training data (see Section 4.2 for an example). This generalization effect is even stronger for more diverse training datasets (which also causes the demand for large datasets in the area of deep learning). For this reason, a supervised training on the source and target domain achieves well generalizing models.

Does self-training improve generalization capabilities? The generalization capabilites of supervised training methods could explain the good generalization capabilities of self-training approaches mimicking such a supervised training on both domains. Here, providing high-quality pseudo-labels remains the crucial challenge to achieve this generalization effect through the target domain.

What causes EasyAdap's performance drop on the second domain change? According to Table 1, EasyAdap achieves near state-of-the-art performance on the GTA5 to Cityscapes domain change. However, Table 2 shows a performance drop of EasyAdap. We assume it is due to our

self-supervision's sensitivity to larger domain gaps; thus, the simplicity of our approach yields a trade-off. Improved future datasets of the synthetic domain might mitigate these impediments.

Does self-training always work? We observed that self-training requires an initial model with sufficient understanding of the target-domain data already. If this requirement is not satisfied, including self-training in the training process even decreases the overall quality of the resulting model. Hence, there seems to be a threshold of the initial model's quality for the usefulness of self-training.