Federated Learning for Autonomous Driving: Enhancing Style Transfer Techniques for FFreeDA

Luca Agnese
Politecnico di Torino
DAUIN
l.agnese@studenti.polito.it

Fabio Rizzi
Politecnico di Torino
DAUIN
s308770@studenti.polito.it

Flavio Spuri
Politecnico di Torino
DAUIN
s303657@studenti.polito.it

Abstract

Semantic Segmentation (SS) is a task fundamental to enabling the diffusion of self-driving vehicles, providing them with a tool to precisely understand their surroundings by assigning a class to each pixel of an image. However, in this application it would rely on sensitive data gathered from users' vehicles, raising substantial privacy concerns. In response to this challenge, in this paper we explore the application of Federated Learning (FL). However, often FL is implemented under the unrealistic assumption of having labeled data at client side. Thus, we also explore the recently introduced task of FFreeDA, in which clients' data is unlabeled and the server accesses a source-labeled dataset for pre-training. Moreover, we expand on existing style-transfer techniques by introducing two novel approaches to perform Fourier Domain Adaptation (FDA). In one, we introduce a clustering scheme using fuzzy logic at inference time. In the other, we try to mimic the presence of unseen sub-styles in the target dataset, aiming to improve the capability of generalization of the model. Both seem to yield some promising results, showing improvements upon the existing methods. ***Source Code****: https://github.com/Flavios3/ESTT*

1. Introduction

Semantic segmentation (SS) involves assigning a class to each pixel in an image. This task is critical for numerous applications, particularly self-driving [1] [2] vehicles, where it aids in precisely identifying key components like pedestrians, road signs, and cars.

Nevertheless, the task of SS in autonomous driving poses certain complexities. Key among these is the issue of dealing with data that is sensitive to privacy and the scarcity of adequately labeled data for the training process.

Regarding the first issue, a solution comes from the relatively new field of Federated Learning (FL) [3]. FL is a learning paradigm in which the task is solved through a collaboration between several devices, called *clients*, coordinated by a central *server*. Each client works locally on its own data without the need of transmitting it. This allows the data to remain unseen by the server which solves the issue of respecting users' privacy.

The second issue concerning the lack of adequately labeled data in SS arises from the difficulty of manually annotating images for this task. Indeed, as reported in [4], densely annotating a single image requires on average more than 1.5 h, making manual annotation highly time-consuming. However, most of existing FL works assume the availability of labeled data on the client side. This assumption is clearly unrealistic. First, because of the aforementioned difficulty in annotating data in the context of SS. Second, as in real-world scenarios clients would continuously collect data, it would be highly challenging, if not impossible, to maintain an updated and adequately labeled dataset.

This made it necessary to introduce the novel task of Federated source-Free Domain Adaptation (FFreeDA) [5] where this unrealistic assumption is dropped. In FFreeDA the clients only access their unlabelled *target* dataset while the server is pre-trained on a labeled *source* dataset. In particular, in our application we considered a synthetic source dataset; in this way, we remove the burden of manually annotating data.

However, this creates a gap between the source and the target dataset. To reduce the discrepancy between the source and target distributions and make the pre-training more effective we employ a style transfer technique based on FDA (Fourier Domain Adaptation). Thus, we randomly load the target styles onto the source images to reduce the gap between the two distributions.
Finally, we introduce two novel techniques to perform style transfer. In the first, we introduce a clustering scheme similar to the one presented in [5], and use fuzzy logic at inference time. In the second we try to enhance the candidate target styles so as to mimic the presence of ones unseen in our dataset.

2. Related Work

Semantic Segmentation (SS). The field of SS has seen remarkable advancements with the use of deep neural networks. Generally, deep neural networks learn a mapping between a semantic label and its visual appearance by training the network with a large number of images and their corresponding pixel-wise labels. State-of-the-art approaches rely on encoder-decoder architectures based on CNNs or Transformers [6] [7].

Federated Learning (FL). The main purpose of FL consists in acquiring knowledge from distributed devices, which store data locally, and using that data to train a unified central model [3]. The objective is to achieve this learning task while ensuring that the device-generated data remains local to the *clients*, with periodic transmissions of intermediate updates to a central *server*. Recent developments specific to FL are paving the way toward efficient, privacy-preserving solutions.

Domain Adaptation (DA). To tackle the problem of the expensiveness of having annotated labels in SS, an increasing number of methods rely for their training on synthetic data, i.e. on datasets generated in virtual environments. However, models trained with this logic fail to generalize when applied to a different dataset, especially real-world ones, because of inherent shifts in the distributions. DA [8] is a particular form of transfer learning which aims to reduce this shift, training a model that can generalize well from a *source* domain to a different *target* domain. Of particular interest is the case in which the target dataset is unlabelled; in this case the task is called Unsupervised Domain Adaptation (UDA) [9].

Self Supervised Learning (SSL). Given the frequent reality of absent or insufficient labels in SS datasets, SSL emerges as a natural approach. SSL is a learning paradigm in-between supervised and unsupervised which enables models to discern an internal supervisory signal from the data itself, often represented as automatically generated pseudo-labels. These internally derived labels can then be employed in a subsequent learning stage, which might operate under supervised or unsupervised principles, leveraging the rich representations learned through SSL.

3. Methods

In this section, we present our proposed approach. We give an overview of the overarching methodology, as well as the details for each specific task.

3.1. Datasets

IDDA. The IDDA [10] dataset is a large synthetic dataset counting over one million images labeled for SS in the autonomous driving context. Such images are collected under more than 100 different scenarios, considering three axes: town, weather and illumination, and viewpoint (i.e. the car model). We can then refer to different domains as triads town_weather_car. To create our federated dataset, the selected images were divided into 24 train clients and 2 test clients. We also reduced the number of semantic classes from the original 24 to a subset of 16. Each train client contains 25 images taken from one of 12 possible scenarios. The two test clients were taken so that one has a similar latent distribution to the train clients, while the other one differs from them; specifically, the first test client contains 120 images taken across the same 12 scenarios used for the train clients, while the second test client contains 120 images taken from 6 scenarios different from the others. We will refer to them as *Test Same Dom* and *Test Diff Dom* respectively. We will instead refer to the concatenation of all the train clients as *Train*.

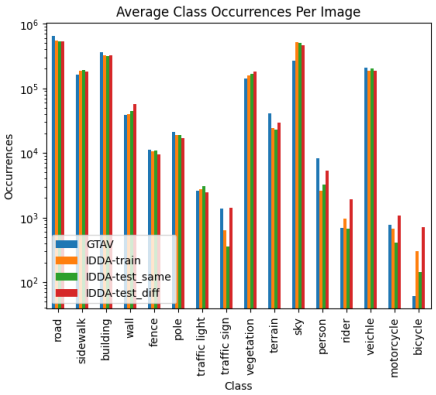


Figure 1. Distribution of the classes among the datasets: the graph reports the average number of occurrences for each of the 16 semantic classes in each of the 4 considered datasets, in a logarithmic scale.

GTAV. GTAV [11] is another synthetic dataset for SS in autonomous-driving applications, containing almost 25k labeled images rendered from the highly realistic video game Grand Theft Auto V. These images are all from the car perspective in the streets of American-style cities. In particular, we consider a subset of 500 images, and we align the original 19 semantic classes to our considered 16. We will refer to this dataset as *GTAV*.

In Figure 1, we can see the average occurrences of each of the 16 selected classes across *GTAV* and IDDA's *Train*, *Test Same Dom* and *Test Diff Dom*. As evidenced by the graph, there is some discrepancy in the distributions of different datasets. Specifically, the Train and Test Same Dom datasets exhibit consistent distributions, while the Test Diff Dom and the GTAV datasets present different characteristics.

3.2. Metrics

Pixel Accuracy (pAcc). pAcc is a metric used in SS that calculates the ratio of correctly classified pixels to the total number of pixels. While very straightforward and easy to

compute, it is unable to take into account class imbalance, possibly yielding misleading results.

Intersection-Over-Union (IoU). IoU is the most commonly used metric in SS. While remaining pretty straightforward, it overcomes the issue of treating unbalanced classes. For a given class c, it is defined as the ratio between the area of overlap and the area of union between prediction and ground truth:

$$IoU_c = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

where A is the set of pixels assigned with class c and B is the set of pixels with label class c. Our main evaluation metric is the mean IoU (mIoU) between all 16 semantic classes.

3.3. Network Architecture

While the training paradigm changes for different tasks, the network employed is kept the same. In particular, we use the DeepLabV3 [12] network with MobileNetV2 [13] as the backbone.

MobileNetV2. MobileNetV2 is a lightweight Convolutional Neural Network that balances accuracy and computational cost. It employs depth-wise convolutions and inverted residuals to achieve such balance.

DeepLabV3. DeepLabV3 is a state-of-the-art architecture developed by Google for SS. The model uses a technique called atrous convolution (or dilated convolution) to control its field-of-view and efficiently capture multi-scale information. This is coupled with a feature called atrous spatial pyramid pooling (ASPP) to capture objects and contexts at different scales.

3.4. Centralized Baseline

Our first step consists in training a centralized baseline on IDDA, using *Train* as our training dataset. To optimize the hyper-parameters, we have to compromise with our limited computational resources, as extensively running several experiments to perform hyper-parameter optimization is prohibitive. Thus:

- **i)** First, we perform a coarse grid search to get a sense of the problem. In it, we try several configurations of learning rate (lr) and weight decay (wd), both with the SGD and the ADAM optimizers, for a very limited number of epochs, and discard those that give a poor initialization.
- **ii)** Then, we train the best configurations for both optimizers for a larger number of epochs, thus selecting the best configuration of lr, wd, and optimizer.
- **iii)** Finally, we run for a smaller number of configurations and for a greater number of epochs, to select the best lr scheduler and data augmentation.

In the end, we obtain the set of hyper-parameters Θ_c, that can be seen in Table 1.

3.5. Supervised FL

Maintaining the set of hyper-parameters Θ_c, we now move toward a Federated framework leveraging the *FedAvg* algorithm [14]. For each round of communication $t \in [1, num\_rounds]$, the server selects a certain number of clients $clients\_per\_round$ within the 24 available train clients. The server then sends the current model parameters ω^{t-1} to the selected clients, and each client c performs a num_epochs number of local epochs, obtaining the updated parameters ω_c^t. At the end of each round, the server collects all the client-updated parameters $\{\omega_c^t | c = 1, ..., clients\_per\_round\}$ and aggregates them by computing their average, weighted by the clients' cardinalities. Note that this is equivalent to updating the central model using the average gradient computed across the selected clients.

3.6. Moving Towards FFreeDA

To now move toward the more realistic scenario of not having access to the ground truth labels, we want to consider the case in which the model is pre-trained on a labeled dataset. In particular, we use *GTAV* for the pre-training. First, we tune the hyper-parameters following a similar pipeline to 3.4, obtaining the set of hyper-parameters Θ_{pt} which can be seen in Table 2. During this step, we validate on IDDA *Train*.

We then try to reduce the discrepancy between target and source domains. To do so, we maintain the same set of hyper-parameters Θ_{pt} and, during the training, we apply the Fourier Domain Adaption (FDA) technique [15]. FDA transfers the style from an image in the target domain to an image in the source domain by replacing the low-frequency spectrum of the latter with the one from the former; this is controlled by a new hyper-parameter β, which determines the size of the window in the frequency domain that is replaced. Although it only involves calculating a Fourier transform and its inverse, it achieves state-of-the-art performances in SS tasks. In our particular application, we first extract the average style for each IDDA train client, and then, while training on *GTAV*, on each image we apply a random style from those extracted.

While we keep the set of hyper-parameters Θ_{pt}, we still need to optimize the value for the window's size β. Given our computational constraints, we test several values of β, training for a limited number of epochs. Then, we take the best-performing ones and train with them for longer, in order to determine the optimal choice.

3.7. Federated Self-Training using Pseudo-Labels

Finally, we move toward the SSL paradigm: from now on, we consider the IDDA *Train* dataset to be unlabeled. Our learning framework consists of two components, a *student* model and a *teacher* model, inspired by the two-way learning presented in [16]. In this arrangement, the teacher model

Table 1. Centralized Baseline Results. Refer to 4.1 for further details regarding hyper-parameters, especially the data augmentation pipeline.

Θ_c					Number of Training Epochs	Eval mIoU	Test Same Dom mIoU	Test Diff Dom mIoU
Lr	**Wd**	**Opt**	**Sched**	**Set of Transforms**				
0.1	0.0001	SGD(m=0.9)	PolyLR	advanced	100	0.5807 ± 0.0028	0.5087 ± 0.0018	0.3211 ± 0.0031

Table 2. Pre-training Results

Θ_{pt}					β	Training Epochs	GTAV mIoU	IDDA Train mIoU	Test Same Dom mIoU	Test Diff Dom mIoU
Lr	**Wd**	**Opt**	**Sched**	**Set of Transforms**						
0.01	0.0001	SGD(m=0.9)	PolyLR	advanced	n.a. 0.000001	100	0.5824 ± 0.0056 0.5680 ± 0.0030	0.2665 ± 0.0036 0.2708 ± 0.0046	0.2633 ± 0.0036 0.2772 ± 0.0043	0.2050 ± 0.0030 0.1954 ± 0.0028

generates the pseudo-labels upon which the student model learns. In particular, the training process can be outlined as follows:

- Both the teacher and the student model are initialized using the best performing model pre-trained on *GTAV*.
- The teacher model computes the pseudo-labels which are used as ground-truth to train the student model, still using the FedAvg algorithm.

We also considered three different strategies to update the teacher model. In the first, it is never updated. In the second, the teacher model's parameters are set to be the ones of the current student model at the beginning of each FL round. In the third, we only perform this update of the teacher model's parameters every $T > 1$ FL rounds.

3.8. Ensemble Learning

Based on the work done in [5], we now try to explore a different framework for our network: we introduce an ensemble model based on a newly introduced clustering scheme.

Specifically, the clustering scheme is introduced following the one proposed in [5]. As a first step, the average styles of the train clients are grouped based on the clustering algorithm as described in the original work. I.e., for varying values of the number of clusters K, we perform K-means aiming to minimize the intra-cluster distance; among the best candidate for each K, we then select as our final clustering the one with the best silhouette score.

We then use the same training strategy as described in 3.6, applying the FDA technique. However, we train a separate model for each of the identified clusters. These models are trained by applying only the styles from their respective clusters to the training data.

At inference time, we adopt a fuzzy logic approach to clustering. Instead of assigning each test image to a single, definitive cluster based on its style, we calculate a series of scores, each representing the degree of similarity between the style of the test image and that of each cluster. In this way, rather than assigning a test image to a single cluster, we provide a graded similarity score for each cluster. In particular, such similarity scores are computed as the normalized distances between the style of the image and the centroids of the clusters. For simplicity, from now on we will purposely confuse the clusters and their corresponding models.

These similarity scores are used to determine the weight with which the outputs of the models are aggregated. In particular, we consider two weighting schemes. In the first, denominated *standard*, we simply take the normalized similarities as they are. In the second, we instead try to further penalize the outputs from the clusters that are farther from the test image style; to do so, we cube the similarity scores and re-normalize them. Clearly, this operation inherently amplifies the larger values and diminishes the smaller ones. We refer to this second weighting scheme as *skewed*.

The final output will then be a weighted aggregation of the outputs from each of the K trained models, based on one of the weighting schemes. We consider several types of aggregation[1].

- **Max**. We take as output the one from the cluster that is closer to the test image style; we note that this is equivalent to the approach proposed in [5], although with the difference that in the original work a distinction was made between global and local model parameters.
- **Mean**. This method computes the weighted average value across the outputs. The idea is to balance out any outlier prediction and provide a more stable output.
- **Median**. This approach works similarly to Mean but considers the median of the predictions instead of their average.
- **Majority**. We aggregate using a majority vote system, choosing the label depending on which class receives the most votes. Note that this doesn't take into consid-

[1]To avoid excessive wordiness, in the descriptions of the aggregation methods we intentionally do not distinguish at which level the aggregation happens. While it should be immediate from the type of aggregation, we note that those may happen at the level of the whole image label, of the pixel-wise labels, or of the likelihood vectors associated with each pixel.

eration the similarity scores.

- **Random by output**. We randomly select one of the outputs weighted by the considered weighting scheme.
- **Random by pixel**. We randomly select the pixel-wise label weighted by the considered weighting scheme.

3.9. Simulating Unseen Styles

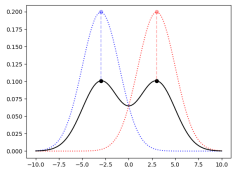
Finally, we present the methodology for a new approach to performing style transfer, where we aim to augment our model's ability to recognize "unseen" styles. Previously, we performed FDA by applying a random style from those extracted from *Train*. While this should bring tangible benefits when considering *Test Same Dom* we do not expect to work too well for *Test Diff Dom*, as we are transferring styles from a different subset of the IDDA dataset.

In this extension, we sought to enrich the model's capacity to adapt to new and previously unseen styles. To effectively achieve this goal, we suppose the simulated styles have to adhere to two crucial properties: 1) They should bear enough resemblance to the real styles, ensuring that they wouldn't originate from a completely different distribution. 2) They should diverge sufficiently from the original styles to foster better generalization in the model, rather than further strengthening its capability on the already observed ones.

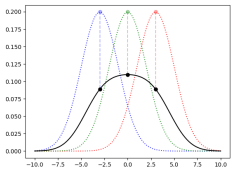
We have devised a strategy to create new styles by interpolating between couples of observed ones. Recalling that the styles can be thought of as matrices, given two observed styles S_1 and S_2 one can intuitively consider generating a new style $\hat{S}$ by averaging the two, i.e. $\hat{S} = wS_1 + (1-w)S_2, 0 < w < 1$. While this would suffice in verifying the first property, to also verify the second we further imagine that the in-between style is actually one of the two observed ones, and thus, inverting the previous formula, we generate $\hat{S} = \frac{1}{w}S_2 - \frac{1-w}{w}S_1$ (note that by randomly extracting S_1 and S_2, their role as the other external style is interchangeable).

Moreover, in order to not introduce too much confusion in the model, we employ two further expedients. Instead of using the average style for each train client as our bank of observed styles, we further group and average them based on the clustering obtained in 3.8. Indeed, the number of possible generated styles from an original bank of dimension n, for a fixed w, is $n^2 - n$, so reducing n quadratically reduces the number of possible originated styles. Moreover, we want to assure that the model is still leaning toward the observed styles, and thus we only transfer a generated style with probability $p, 0 < p < 1$.

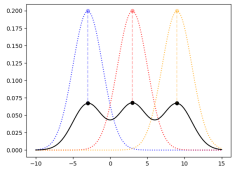
For a better visualization of our approach, let's conceptualize the cluster of styles as Gaussian distributions. In this view, each style is a data point extracted from the distribution, and the average style corresponds to the mean of the distribution. Thus, we can envision the underlying distribution of all the styles present in the IDDA dataset as the mixture of all the Gaussian distributions, encompassing both seen and unseen styles. The considered styles in our previous methodology are then simply the components' means, as illustrated in Figure 2a. We can then see that simply generating a new style in-between two observed ones would merely concentrate the mixture around its mean (Figure 2b). Instead, generating "external" styles would actually disperse the distribution more broadly. This introduces a range of component means that lie well beyond the observed ones, as depicted in Figure 2c. The comparison of these three potential mixtures can be seen in Figure 2d.



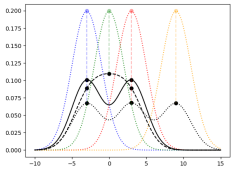
(a) Original mixture of styles



(b) Generation of an in-between unseen style



(c) Generation of an external style



(d) Comparison of possible mixtures

Figure 2. Depiction of the mixtures for the generation of an unseen style. The blue and red bell curves represent two observed distributions, while the green and orange ones represent a style simulated in-between or externally, respectively

4. Experiments and Results

In this section, we present the results obtained for each of the tasks described in the previous section. The experiments were run using Google Colab and Google Colab Pro, leveraging a T4 GPU with 15GB of dedicated memory.

4.1. Centralized Baseline

We report in Table 1 the result for the Centralized Baseline. For the sake of brevity, we only report the best configuration as obtained from the steps described in 3.4, trained on a large number of epochs, although intentionally limited to ensure comparability with the subsequent experiments. We note that the tested hyper-parameters, albeit not in all their possible combinations, were:

- $lr \in \{0.1, 0.01, 0.001, 0.0001\}$
- $wd \in \{0.0001, 0.00001, 0\}$
- $optimizers \in \{SGD(momentum = 0.9),$
  $Adam(\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}),$
  $Adam(\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-1}),$
  $Adam(\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-5})\}$
- $LRdecay \in \{StepLR, PolyLR\}$

- $DataAugmentation$:
  - $basic = \{RandomResizedCrop\}$
  - $advanced = \{RandomResizedCrop, RandomHorizontalFlip, ColorJitter\}$

We obtain some good results on *Test Same Dom*, comparable to those on the Eval set. However, there is a pretty significant reduction in the performances on *Test Diff Dom*. This is consistent with the gap between the datasets highlighted in 3.1.

4.2. Supervised FL

To analyze the performance of our FL algorithm, we run 3 sets of experiments.

In a realistic scenario, the computational effort would be moved as much as possible onto the client devices, i.e. keeping the number of communications as low as possible while running as long as possible on each client. Thus, our first set of experiments aims at analyzing how the number of local computations changes the final performance. In particular, for each $clients\_per\_round \in \{2, 4, 8\}$, we fix num_rounds and run experiments for changing $local\_epochs \in \{1, 3, 6\}$. As it can be seen in Table 3, there is an obvious improvement when increasing either $clients\_per\_round$ or $local\_epochs$, as they imply an overall increase of training steps. However, the increased performance does not always seem to justify the increased computational effort: indeed, while increasing the $local\_epochs$ from 1 to 3 yields some notable improvements, further increasing it to 6 fails to yield significant additional gains, especially on the test datasets. This is even more evident when increasing $clients\_per\_round$, and thus the number of communications: while one might expect a substantial impact, the actual improvement appears to be marginal; we will further analyze this last observed behavior in the third set of experiments.

We run a second set of experiments where we compare the performances of the two configurations with the most different setups when closer to convergence. I.e., we fix $local\_epochs$ to 3, as it gave the best trade-off in the previous set of experiments, and run the experiments with $clients\_per\_round$ equal to 2 and 8 for 100 rounds. Results can be seen in Table 4. It's worth noting that although these two experiments involve different quantities of communications, if we consider all client computations occurring in parallel in each round they would require the same duration. In an application in which the primary concern is time constraints, rather than limitations on computational and communication resources, it is clearly better to select a larger number of clients in each round.

Finally, we focus on the trade-off between performance and the number of communication rounds. Specifically, we now establish a target performance of $mIoU = 0.38$ and, fixing again $local\_epochs = 3$, we run the experiments for each $clients\_per\_round$ until the target mIoU is reached; note that the number of communications is the number of rounds taken to reach the target performance multiplied by the number of clients per round. The results, displayed in Table 5, make it evident that the increase in the accuracy is not proportional to the augment of the computational cost. The most efficient approach appears to be keeping the $clients\_per\_round$ low and executing a larger number of rounds.

Despite FL seems to achieve good performances we note a not insignificant dip of about 6 percentage points compared to the centralized baseline.

4.3. Moving Towards FFreeDA

Similarly to 4.1, for the case without FDA we select the best hyper-parameter configuration, as determined when near convergence. We report its results when used in an experiment of 100 epochs, to keep the various experiments comparable. These results can be seen in the first row of Table 2. The possible values considered for the hyper-parameters are the same as 4.2, with the only difference that we don't consider the variations of the optimizer Adam, but instead try a new lr decay schedule in *cosine annealing*, which still performs worse then LRPoly.

For what regards the application of FDA, we first tune the hyper-parameter β. To obtain a sensible set of values to try, we empirically look at how they affect the source image when applied. In particular, we observe that past a certain threshold, which we identify to be around 0.05, the source image with the transferred style started showing some "hallucination"; such effect can be seen in Figure 3. Thus, we tested several values under this threshold, and only a few values over it. To do so, we limit the number of epochs to 20: while FDA is a very lightweight approach to performing style transfer, it still makes the computation more demanding, thus we have to further limit the number of epochs performed. The results for some of the tested values of β can be seen in Table 6. As it is evident from the reported results, we achieve some good performances for smaller values of β; instead, when growing closer to the threshold, the performance already starts to deteriorate. We then take as our hyper-parameter $\beta = 0.000001$.

Finally, we report the result for the experiment run with the selected β for a larger number of epochs, which can be seen in Table 2. Specifically, we select the number of epochs to be 100, which is the most we could afford given our setup. Being this the most demanding experiment, we report the results for the other experiments with this number of epochs as well, so as to keep them comparable.

Comparing the results with and without the application of FDA, we note a behavior consistent with our expectations. When using FDA, the performance on the source train dataset

Table 3. Federated Results - Fixed Clients per Round

Clients per round	Number of rounds	Local Epochs	Eval mIoU (train partition)	Test Same Dom mIoU	Test Diff Dom mIoU
2	30	1	0.3109 ± 0.0109	0.2835 ± 0.0158	0.1953 ± 0.0317
		3	0.3585 ± 0.0052	0.3416 ± 0.0023	0.2470 ± 0.0085
		6	0.3868 ± 0.0032	0.3500 ± 0.0024	0.2556 ± 0.0017
4	30	1	0.3300 ± 0.0053	0.3077 ± 0.0114	0.2192 ± 0.0205
		3	0.3725 ± 0.0029	0.3472 ± 0.0057	0.2588 ± 0.0122
		6	0.3795 ± 0.0074	0.3455 ± 0.0094	0.2647 ± 0.0160
8	30	1	0.3426 ± 0.0031	0.3202 ± 0.0053	0.2382 ± 0.0095
		3	0.3773 ± 0.0020	0.3542 ± 0.0035	0.2543 ± 0.0126
		6	0.4035 ± 0.0031	0.3715 ± 0.0030	0.2682 ± 0.0089

Table 4. Federated Results - Long Experiments

Clients per round	Number of rounds	Local Epochs	Eval mIoU (train partition)	Test Same Dom mIoU	Test Diff Dom mIoU
2	100	3	0.4434 ± 0.0071	0.3962 ± 0.0093	0.2667 ± 0.0178
8	100	3	0.4901 ± 0.0041	0.4485 ± 0.0038	0.2946 ± 0.0045

Table 5. Federated Results - Target mIoU

Local Epochs	Target mIoU	Client per Round	# Communications
3	0.38	**2**	**82**
		4	144
		8	256

Table 6. Beta Tuning

β	Eval mIoU
0.0000001	0.2539
0.000001	0.2679
0.00001	0.2612
0.01	0.2598

Figure 3. Visual effects of applying styles for different values of β

is slightly lower, as the images used in training get slightly modified. However, the performance on the target dataset increases, which ultimately was our aim in this task. The performance on the *Test Diff Dom* still sees a small decrease: while this might be explainable due to random fluctuations, it is likely also due to the fact that we are only transferring styles from the scenarios seen in *Train* and *Test Same Dom*.

4.4. Federated Self-Training using Pseudo-Labels

To test the performance achieved in this task, we consider the experiments run as following.

- As our pre-trained model, we consider both the best-performing model without using FDA and the best-performing one using FDA.
- For the training, we use the set of hyper-parameters Θ_c. Moreover, we set the number of local epochs to 1 and consider $clients\_per\_round \in \{2, 8\}$.
- We test all three teacher update strategies described in 3.7. For the third strategy, we consider two values of T, specifically $T = 5$ and $T = 15$.

Note that we identify a teacher update strategy with its corresponding T, i.e. we will identify them as $T = \infty, T = 1, T = 5$ and $T = 15$.

By running the experiment using Θ_c, we notice that the performance decay very quickly. We identify the problem to be a too-large value of $lr = 0.1$. We then try to tune again the value of lr, which we eventually set to $lr = 0.00001$.

A small part of the obtained results can be seen in Table 7. For the sake of conciseness, we don't report the results for all the performed experiments. However, we report how across our experiments this method doesn't bring any significant improvement, generally being detrimental to the performances, as both the teacher and the student make their prediction using the same exact model weight. Thus, the pseudo-label and the predictions are almost the same except for some small randomicity, which leads to what we expect to just be random fluctuations that eventually degrade the model performance.

Table 7. Results with Pseudo-Labels

Clients per round	T	lr	FDA	Train mIoU	Test Same Dom mIoU	Test Diff Dom mIoU
2	$+\infty$	0.00001	X	0.3202	0.2904	0.2206
			✓	0.3350	0.2964	0.2026
8	5	0.1	X	0.0066	0.0061	0.0058
			✓	0.0160	0.0164	0.0167

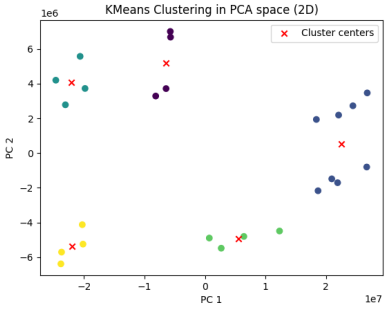


Figure 4. Best clustering of the average styles

4.5. Ensemble Learning

Performing the clustering algorithm described in 3.8, we obtained that the optimal number of clusters is $K = 5$. The resulting clustering can be seen in Image 4.

The results obtained from the various aggregation methods can be seen in Table 8. The most notable result is that we find three aggregation methods - Mean, Majority, and Median - that outperform Max. In particular, we have that Majority is the best performing on *Test Same Dom*, whereas Mean is the most effective on *Test Diff Dom*. Overall, Mean, Majority, and Median all surpass Max on both test datasets. Moreover, we notice that the *skewed* weighting scheme typically underperforms.

Table 8. Results for Ensemble Learning

Aggregation	Weighting Scheme	Test Same Dom	Test Diff Dom
Max	n.a	0.2524	0.2008
Mean	Standard	0.2688	0.2094
	Skewed	0.2595	0.2056
Median	Standard	0.2675	0.1936
	Skewed	0.2614	0.1674
Majority	n.a	0.2713	0.2059
Random by Output	Standard	0.2609	0.1880
	Skewed	0.2547	0.1943
Random by Pixel	Standard	0.2552	0.1887
	Skewed	0.2531	0.1949

The observations we have made seem to suggest that the optimal approach for the ensembling of models trained on different clusters involves trying to produce a weighted yet balanced prediction from the generated outputs. Indeed, the *standard* weighting scheme performs generally better, and the aggregation schemes that yield the highest performance are those that result in a balanced prediction.

4.6. Simulating Unseen Styles

To assess the effectiveness of the methodology introduced in 3.9, we run two experiments using different values of p, specifically $p = 0.2$ and $p = 0.5$. The results of this experiments can be found in Table 9 under the name *interpolation*. For comparison, we also report the results obtained in the best-performing configuration of the standard use of FDA as discussed in 3.6. These results are reported under the name Default.

As we can see, we have a slight dip in performance on *GTAV*, an outcome we expected given the additional modifications to the training images. However, this in turn yields a good improvement in the performances on IDDA. Notably, we observe a significant improvement of around 2.5 percentage points on *Train* and *Test Same Dom*, coupled with another noteworthy increase of 1.5 percentage points on *Test Diff Dom*, which is our ultimate goal in this extension.

To discern if the observed changes in performance stem from our new methodology's ability to simulate unseen styles or if they are merely a byproduct of increased style ambiguity, we also introduce a control experiment. In it, we select styles randomly from the cluster averages and add Gaussian noise to them. The results, identified as *noise*, seem to indicate that indeed the increased ambiguity contributes to the model's enhanced generalization ability to the IDDA dataset. This is evidenced by the improvement on *Train* and *Test Same Dom*, which is comparable to that obtained using *interpolation*. However, the improvements obtained on *Test Diff Dom* do not seem to measure up to those produced by our proposed methodology. While we couldn't account for possible fluctuations caused by different initializations, we still believe it is possible to conclude that, while the improvement on the "same domain" datasets is explainable by the added ambiguity, and thus by thinking of FDA as an additional data augmentation step, our proposed methodology still seems able to capture some unseen style distributions, as the improvement on *Test Diff Dom* is nearly triple when using *interpolation* instead of *noise*.

5. Conclusions

In this study, we were able to explore the efficacy of FL as a paradigm within the context of SS for autonomous driving. Although we observed a dip from the centralized results, FL was still able to achieve good performances despite our limited computational abilities.

Moreover, we explored the possible application of SSL within this context, specifically using pseudo-labels to continue training a pre-trained model without further access to any label. However, we mainly focused our discussion in this regard noting how employing a naive approach in this setting doesn't result in any particular benefit.

Lastly, we focused on the FDA technique to perform style transfer from a target to a source dataset. While we introduced it within the pre-training phase of our model on the source dataset, we delved deeper by investigating two innovative methods for its application, with an aim to improve the performances.

As evidenced by the aforementioned results, we achieved two main contributions. Firstly, we found that in the process

Table 9. Results from the Simulation of Unseen Styles

Type	p	GTAV mIoU	IDDA Train mIoU	Test Same Dom mIoU	Test Diff Dom mIoU
default	n.a.	0.4315 ± 0.0015	0.2352 ± 0.0066	0.2421 ± 0.0068	0.1746 ± 0.0079
interpolation	0.2	0.4299 ± 0.0006	0.2620 ± 0.0031	0.2689 ± 0.0030	0.1909 ± 0.0032
interpolation	0.5	0.4230 ± 0.0018	0.2604 ± 0.0061	0.2639 ± 0.0059	0.1907 ± 0.0068
noise	n.a.	0.4307 ± 0.0016	0.2605 ± 0.0049	0.2648 ± 0.0049	0.1812 ± 0.0057

of assembling models trained under a clustering scheme, the most successful aggregations appeared to be those favoring balanced predictions, rather than those inclined entirely towards the closest cluster. Even more significantly, we were able to devise a technique that seems to improve the generalization to "unseen" styles by simulating them as an interpolation of the observed ones.

References

[1] Weichao Xu, Baojun Li, Shun Liu, and Wei Qiu. Real-time object detection and semantic segmentation for autonomous driving. page 44, 02 2018. 1

[2] Yu-Ho Tseng and Shau-Shiun Jan. Combination of computer vision detection and segmentation for autonomous driving. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1047–1052, 2018. 1

[3] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *CoRR*, abs/1908.07873, 2019. 1, 2

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016. 1

[5] Donald Shenaj, Eros Fanì, Marco Toldo, Debora Caldarola, Antonio Tavera, Umberto Michieli, Marco Ciccone, Pietro Zanuttigh, and Barbara Caputo. Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning, 2022. 1, 4

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. 2

[7] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *CoRR*, abs/2105.15203, 2021. 2

[8] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. A brief review of domain adaptation. *CoRR*, abs/2010.03978, 2020. 2

[9] Sicheng Zhao, Xiangyu Yue, Shanghang Zhang, Bo Li, Han Zhao, Bichen Wu, Ravi Krishna, Joseph E. Gonzalez, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia, and Kurt Keutzer. A review of single-source deep unsupervised visual domain adaptation. *CoRR*, abs/2009.00155, 2020. 2

[10] Emanuele Alberti, Antonio Tavera, Carlo Masone, and Barbara Caputo. IDDA: a large-scale multi-domain dataset for autonomous driving. *CoRR*, abs/2004.08298, 2020. 2

[11] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. *CoRR*, abs/1608.02192, 2016. 2

[12] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017. 3

[13] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 3

[14] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016. 3

[15] Yanchao Yang and Stefano Soatto. FDA: fourier domain adaptation for semantic segmentation. *CoRR*, abs/2004.05498, 2020. 3

[16] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. *CoRR*, abs/1904.10620, 2019. 3