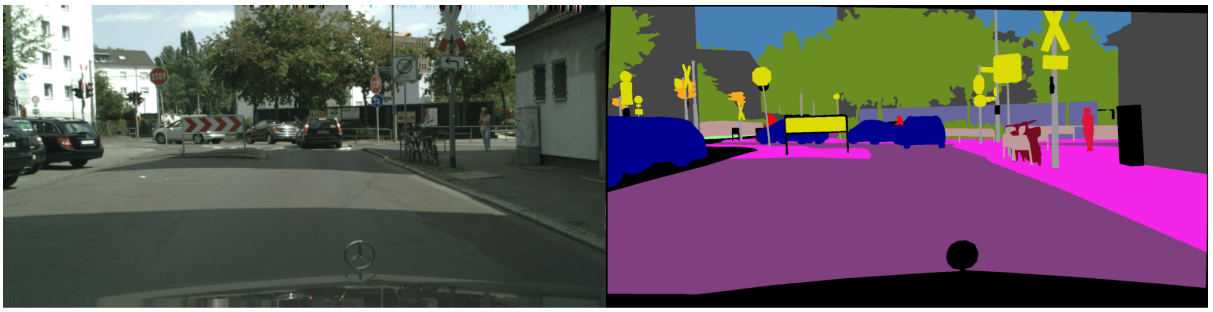
Federated Semantic Segmentation for self-driving cars

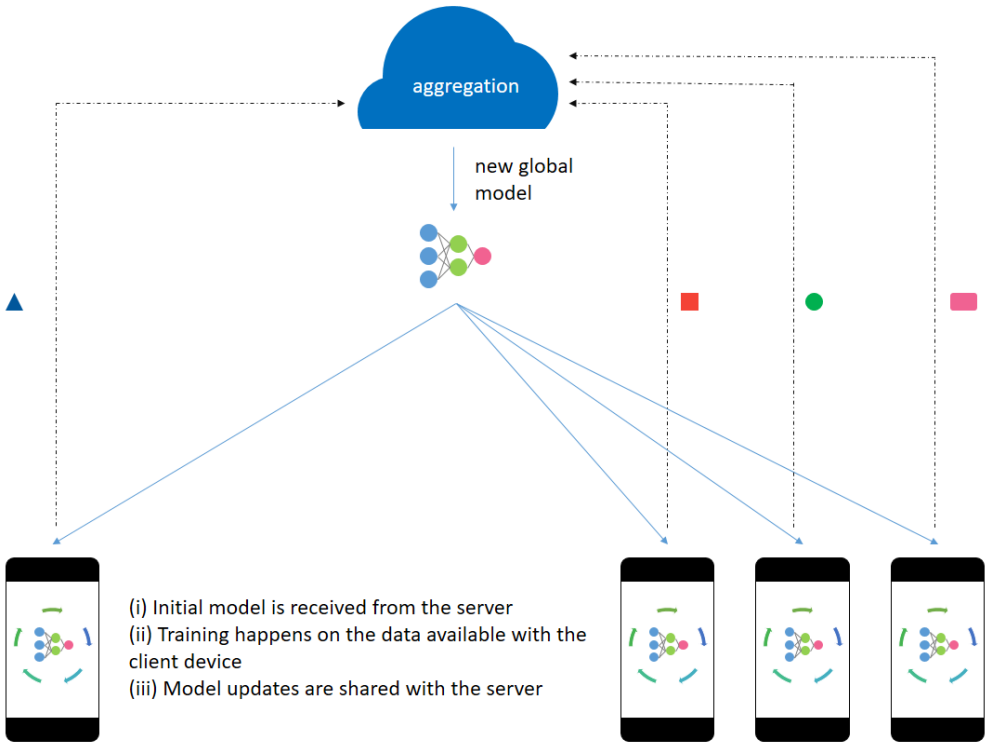
Eros Fanì (eros.fani@polito.it)

PROJECT OVERVIEW AND GOALS

Semantic Segmentation (SS) is essential to make self-driving vehicles autonomous, enabling them to understand their surroundings by assigning individual pixels to known categories. However, it operates on sensible data collected from the users' cars; thus, protecting the clients' privacy becomes a primary concern.



To this end, **Federated Learning (FL)** constitutes a possible solution. FL is a relatively new machine learning paradigm aiming to learn a global model while preserving privacy and leveraging data on millions of remote devices.



However, most of the existing works on FL unrealistically assume labeled data in the remote clients. Thus, the new task of **Federated source-Free Domain Adaptation (FFreeDA)** has recently been introduced. In FFreeDA, the clients' data is unlabeled, and the server accesses a source-labeled dataset for pre-training only.

In this project, you will dive into the motivations and applications of **Federated Learning (FL)** to the **Semantic Segmentation (SS)** task applied to **self-driving cars**. The main objective of this project is to understand what are the FL framework and the SS task and become familiar with them. You will learn the motivations behind applying FL to SS in the context of autonomous vehicles and how to simulate experiments to resemble real-world scenarios using the **PyTorch library**. Moreover, you will understand the difficulties and challenges of this scenario and propose your solutions. Finally, you will be introduced to the new task of **Federated source-Free Domain Adaptation** applied to the Real-time Semantic Segmentation networks (**FFreeDA**).

BEFORE STARTING

Take your time to familiarize yourself with SS and FL, by reading the following papers. Before preceding, be sure to have understood these concepts. You can find the references at the end of this document.

- Semantic Segmentation: [1], [2], [3], [4], [5]
- Federated Learning: [6], [7]

Feel free to explore other online resources.
Moreover, you should be able to perform experiments using the PyTorch library. You might refer to this guide to get started if you need it.

RESOURCES

You can run your experiments on CoLab for free. Here you can find some suggestions to set up your working environment for this project:

- Download the datasets you will use from here and extract them. In the directory, you will find a subset of the Cityscapes dataset [3] and a subset of the GTA5 dataset [4]. In the following of this document, we will refer to these subsets of the original datasets simply as Cityscapes and GTA5
- Upload the dataset folder into your Google Drive account if you plan to perform your experiments on CoLab
- First, mount your Google Drive folder by executing the following code to access the dataset folder on CoLab:
  ```
  from google.colab import drive
  drive.mount('/content/drive')
  ```
- When executing on CoLab, remember to activate the GPU hardware accelerator: Runtime → Change runtime type → GPU

Refer to the following resources to take inspiration for the coding implementations and other details:

- FedDrive official website
- FedDrive official code
- LADD official code

N.B. For easier comprehension, follow the project steps first and explore these resources only after you have read and understood the associated paper.

N.B. The above codes exploit the DistributedDataParellel (DDP) package to work on multi-GPUs and the WandB platform for logging the results. If you are working on CoLab, you can drop the DDP package in your implementation. You may log the statistics of your experiments using WandB if you feel comfortable with it. However, this is not mandatory: any visualization tool is allowed (even text only. But plotting your curves is better for both you and us, to better understand what is happening in your experiments!).

N.B. Run your experiments for a reasonable number of epochs/rounds but consider that we are aware of the limited amount of resources at your disposal. Reduce the number of epochs/rounds if your experiments take too long, but be consistent with your choices in all the steps of the project.

1st STEP) PARTITION CITYSCAPES INTO TRAIN AND TEST

First, generate 4 different **dataloaders** for training and testing Cityscapes. You should **NOT** modify the dataset directory tree. On the contrary, you should programmatically generate two separate train/test partitions. If needed, refer to this guide for datasets and dataloaders in PyTorch.

1) Generate the first train/test partition **(A)** and corresponding dataset classes and dataloaders:
    - Test partition: 2 random images from each city
    - Train partition: the remaining images
2) Generate the second train/test partition **(B)** and corresponding dataset classes and dataloaders:
    - Test partition: the list of images in val.txt
    - Train partition: the list of images in train.txt

N.B. Once you have defined the A train/test partition, it must always be the same for all the following steps.

2nd STEP) CENTRALIZED BASELINE

In this step, you must implement a centralized baseline using the BiSeNet V2 network for the two A and B train/test partitions you built in the previous step. You can find a possible implementation of this model here. As loss function, use the CrossEntropy loss. Try some combinations of hyperparameters and transforms for the same amount of epochs, and choose the best combination according to the mIoU metric. If the best combinations for the two partitions do not match, choose one of the two or another combination you have tried as a reference for the following experiments. Justify your choice.

N.B. In SS, the transforms applied to the image should reflect on the corresponding label. In other words, you need to be sure to avoid misalignment between the pixels of the image and the pixels of the corresponding labels. We suggest using the custom Torchvision transforms you can find here, or the custom OpenCV transforms you can find here.

3rd STEP) FEDERATED SEMANTIC SEGMENTATION EXPERIMENTS

It is time to build your first FL + SS experiment. For the following experiments, use the hyperparameters and the transforms you have found in the previous step.

1) Read [8] and familiarize yourself with the FL+SS task and the code of FedDrive
2) Taking inspiration from the heterogeneous and uniform splits of FedDrive, build your uniform (**I**) and heterogeneous (**II**) splits starting from the train partitions A and B. Since you run experiments on a subset of Cityscapes, assign no more than 20 images to each client.
3) Run FL+SS supervised experiments in the following configurations, using the FedAvg algorithm:
    - train/test partition A, split A.I
    - train/test partition A, split A.II
    - train/test partition B, split B.I
    - train/test partition B, split B.II

4th STEP) MOVING TOWARDS FFreeDA - PRE-TRAINING PHASE

In this step, you will finally move towards a more realistic scenario. In the real world, self-driving cars do not have access to ground-truth labels. If they had ground-truth pseudo-labels, a SS model would not be necessary. Moreover, manually labeling the images is a costly process, and it is not realistic to assume that the clients have access to the labels associated with the images they collect. However, it is reasonable to assume that the model is pre-trained on an open-source dataset as GTA5.

1) Read [9], [10], and [11] to take a cue on how to deal with the Domain Adaptation task in FL for SS, what are the main challenges, and the real-world motivations behind it.
2) Use the images listed in the train.txt file of the GTA5 dataset to train the network from scratch. Pay attention to the semantic classes. You have to select just the 19 in common with Cityscapes. The JSON file provided within the dataset directory indicates the correct mapping between GTA5 and Cityscapes. While training on GTA5, evaluate your model on the test partitions A and B of Cityscapes. Tune your hyperparameters and transforms again., Evaluate the model several times while training for every experiment (every t rounds). Save the checkpoint of your best model among all the evaluations of all the experiments you performed (i.e. not necessarily the model you obtained at the end of the last epoch). But don't be greedy! There is a lot of randomicity in the training procedure (especially if you do not fix the random seed), thus oscillations in the results are the norm. Choose the best model carefully by observing the trend in your performance curves, and justify your choices.
3) Read [12] to familiarize yourself with the FDA technique.
4) Apply FDA as in [11]: extract the average style from each client of the Cityscapes train partition A and form a bank of target styles. Then, using the hyperparameters you found on the previous point, train the network from scratch again, as in point 4.2. However, now substitute every source image style (GTA5 dataset) with a random one from the bank of target styles during the training (you can think of this operation as a new transform). Evaluate your results on the corresponding test partitions of Cityscapes. Test some different FDA window sizes. Repeat for the B train partition, and save the best checkpoints for both partitions A and B.

5th STEP) FEDERATED SELF-TRAINING USING PSEUDO-LABELS

Pseudo-labels can be used as an unsupervised (or semi-supervised) self-training technique to improve performance. In short, to get the pseudo-labels from a batch of images, make a forward pass through a *teacher model*, normalize the predictions and generate a label image for each input image. This label will be used as a ground-truth label by a *student model* during the training (it can also be done in real-time, i.e. without physically storing the pseudo-labels). For each pixel of an image, the teacher model will predict an array of confidence probabilities. If the highest probability is above a certain threshold, the final label is the one with the highest probability. Otherwise, if none of the predictions is above the threshold, the pixel is labeled as *don't care*. Hint: take inspiration from this part of the LADD code for this step.

1) Read [13] to familiarize yourself with pseudo-labels

2) For each partition (A, B) and split (I, II), load the corresponding best checkpoint from step 4.2 and perform a FedAvg training on Cityscapes using pseudo-labels as ground-truth labels for your model, without using the original labels for training:
    - Before the training starts, set student model = pre-trained model, teacher model = pre-trained model
    - Perform FedAvg on the student model using the pseudo-labels generated by the teacher model as ground truth labels. Use the cross-entropy loss. Test the following three strategies to update the teacher model:
        i) teacher model never updated
        ii) teacher model = server model at the beginning of each round
        iii) teacher model = server model every T>1 rounds (the value of T is your choice. You can also test more than one possible T)
3) Repeat point 2, but now use the best checkpoints from 4.4

N.B. Use the correct test partition for each experiment.

6th STEP) IMPROVEMENTS (optional)

Take inspiration from [8] and [11] to improve your work! We propose a list of possible ideas, but feel free to test yours. **Before starting, contact me to explain your ideas and intentions first, even if your idea is to pick one of the following ones listed below.** When not explicitly specified, you can find the references of the following bullet points within [8] and [11].

Possible ideas:

- Implement Domain Adaptation techniques based on the Batch Normalization layers to address the Statistical Heterogeneity problem (see SiloBN, FedBN)
- Propose other partitions and splits. Justify your choices
- Repeat steps 4 and 5 using a subset of the IDDA dataset as the target dataset. You can ask for the download of the dataset here. Ask for version 3 (FedDrive). You are free to decide the subset of the dataset you will use or test another dataset. As always, you should be able to justify your choices
- Try a clustering scheme as in LADD
- Implement regularization techniques as in LADD
- Test FL algorithms other than FedAvg (e.g. FedProx [14], SCAFFOLD [15], your personalized aggregation scheme…)
- Test with some server optimizers [16]. First, verify the equivalence of SGD with momentum = 0 and learning rate = 1 with FedAvg.
- Repeat some steps with another lightweight network, e.g. Mobilenet V2 with a Deeplab V3 head (look here for a possible implementation)
- Explore Satellite image datasets online (some possible datasets can be found here). Download two satellite image datasets for SS and decide which is the

Source Dataset and which is the Target Dataset. Then, define an adequate class mapping between the two. Finally, repeat some of the previous steps with the selected datasets.

AT THE END

- Deliver PyTorch scripts for all the required steps
- Write a complete PDF report (in paper style). The report should contain a brief introduction, a related work section, a methodological section describing the algorithm that you have used, an experimental section with all the results and discussions, and a final brief conclusion. Follow this **link** to open and create the template for the report

EXAMPLE OF POSSIBLE QUESTIONS YOU SHOULD BE ABLE TO ANSWER AT THE END OF THE PROJECT

- What is Semantic Segmentation?
- What is Federated Learning?
- What is Domain Adaptation?
- What is Source Free Domain Adaptation?
- What is Federated Source Free Domain Adaptation?
- Can you justify this result?
- Describe BiSeNet V2
- What is Domain Generalization?
- What is Statistical Heterogeneity?
- How do pseudo-labels work?
- How does FDA work?
- What are the differences between the A and B train/test partitions?

REFERENCES

[1] Survey on SS: A Brief Survey on Semantic Segmentation with Deep Learning
[2] BiSeNet V2 network: BiSeNet V2: Bilateral Network with Guided Aggregation for Real-time Semantic Segmentation
[3] Cityscapes dataset: The Cityscapes Dataset for Semantic Urban Scene Understanding
[4] GTA5 dataset: Playing for Data: Ground Truth from Computer Games
[5] SS metrics: Metrics to Evaluate your Semantic Segmentation Model
[6] Federated Averaging (FedAvg): Communication-efficient learning of deep networks from decentralized data
[7] Survey on FL: Federated learning: Challenges, methods, and future directions
[8] FedDrive: FedDrive: Generalizing Federated Learning to Semantic Segmentation in Autonomous Driving
[9] Survey on DA: A Review of Single-Source Deep Unsupervised Visual Domain Adaptation
[10] Source-Free Domain Adaptation for Semantic Segmentation
[11] LADD: Learning Across Domains and Devices: Style-Driven Source-Free Domain Adaptation in Clustered Federated Learning

[12] FDA: FDA: Fourier Domain Adaptation for Semantic Segmentation
[13] Pseudo-labels: Bidirectional Learning for Domain Adaptation of Semantic Segmentation
[14] FedProx: Federated Optimization in Heterogeneous Networks
[15] SCAFFOLD: SCAFFOLD: Stochastic Controlled Averaging for Federated Learning
[16] Server optimizers: Adaptive Federated Optimization