Supplementary Materials for “Federated Multi-Target Domain Adaptation”

Chun-Han Yao[1] Boqing Gong[2] Hang Qi[2] Yin Cui[2] Yukun Zhu[2] Ming-Hsuan Yang[1,2,3]

[1]UC Merced [2]Google [3]Yonsei University

1. Overview

In this document, we present the implementation details, ablation studies, and additional results of our method.

2. Implementation Details

We implement our framework with TensorFlow [1] and train the models on Tesla P100 GPUs. The feature extractor and classifiers are parametrized with deep neural networks. We apply Batch Normalization [7] after each convolutional layer and use ReLU as the activation function of the middle layers. For local optimization on client devices, we update the model for one epoch of the target data and upload the model parameters to the server. Detailed hyper-parameters in our experiments are shown in Table 9.

2.1. Network architectures

Detailed network architecture for digit recognition is shown in Table 1. For image classification, we use ResNet101 [6] for feature extraction and a linear layer as the classifier. In the semantic segmentation tasks, we adopt DeepLabv3 [2] as our segmentation model and MobileNetv2 [14] with width multiplier $\alpha = 0.5$ as the network backbone.

Table 1. **Model Architecture for digit recognition (Digit-Five dataset [11]).** The feature extractor consists of the four convolutional layers and each classifier includes two fully-connected (FC) layers.

Operator	Input	Output	Kernel size	Stride	Padding	Activation
Feature extractor						
Conv2D	$32^2 \times 3$	$32^2 \times 64$	5	1	2	ReLU
MaxPool	$32^2 \times 64$	$16^2 \times 64$	2	-	-	-
Conv2D	$16^2 \times 64$	$16^2 \times 64$	5	1	2	ReLU
MaxPool	$16^2 \times 64$	$8^2 \times 64$	2	-	-	-
Conv2D	$8^2 \times 64$	$8^2 \times 64$	5	1	2	ReLU
MaxPool	$8^2 \times 64$	$4^2 \times 64$	2	-	-	-
Conv2D	$4^2 \times 64$	$4^2 \times 64$	5	1	2	ReLU
MaxPool	$4^2 \times 64$	$2^2 \times 64$	2	-	-	-
Classifier						
FC	256	64	-	-	-	ReLU
FC	64	10	-	-	-	-

2.2. Dataset statistics

We provide the dataset statistics in our experiments. In Tables 2 (Digit-Five [11]), 3 (Office-Caltech10 [5]), 4 (DomainNet [10]), 5 (GTA5 [12]-to-CrossCity [3]), and 6 (GTA5 [12]-to-BDD100K [15]), we report the number of training/testing examples in each dataset we use. Note that we do not use the testing data from source domains since the goal is to train a federated model that performs well on the target domains.

Table 2. **Number of training/testing examples in the Digit-Five [11] experiment.**

	Source domain	**Target domains**			
	MNIST	MNIST-M	SVHN	Synthetic	USPS
Train	25,000	25,000	25,000	25,000	7,348
Test	-	9,000	9,000	9,000	1,860

Table 3. **Number of examples in the Office-Caltech10 [5] experiment.**

	Amazon	Caltech	DSLR	Webcam
Total	958	1,123	157	295

Table 4. **Number of training/testing examples in the DomainNet [10] experiment.**

	Clipart	Infograph	Painting	Quickdraw	Real	Sketch
Train	34,019	37,087	52,867	120,750	122,563	49,115
Test	14,818	16,114	22,892	51,750	52,764	21,271

Table 5. **Number of training/testing examples in the GTA5 [12]-to-CrossCity [3] experiment.**

	Source domain	**Target domains**			
	GTA5	Rio	Rome	Taipei	Tokyo
Train	25,000	3,200	3,200	3,200	3,200
Test	-	100	100	100	100

Table 6. **Number of training/testing examples in the GTA5 [12]-to-BDD100K [15] experiment.**

	Source domain	**Target domains**			
	GTA5	Cloudy	Overcast	Rainy	Snowy
Train	25,000	4,535	8,143	4,855	5,307
Test	-	346	1,254	215	242

2.3. Calculations of communication and computational costs

In the manuscript, we report the server-client communication cost in terms of the number of model parameters that need to be transmitted per federated training round. For computational cost, we calculate the client-end FLOPs caused by a feedforward or backpropagation pass of a data sample per iteration. To compare the model performance fairly, the numbers should be multiplied by the number of training iterations/rounds until convergence. Specifically, the total communication cost during the training process can be expressed as:

$$(\text{upload cost} + \text{broadcast cost}) \times \text{number of clients} \times \text{number of federated rounds}. \tag{1}$$

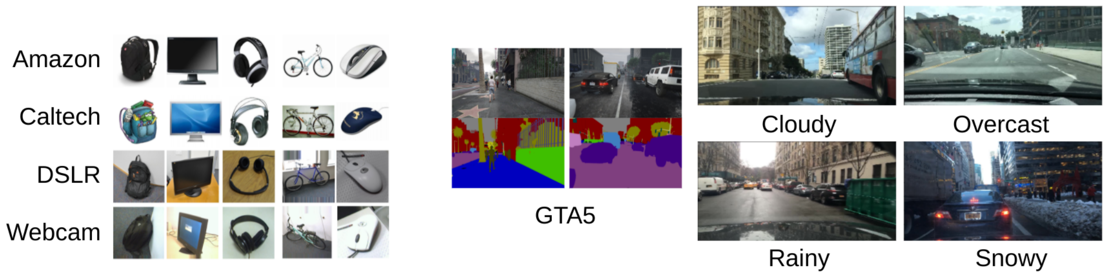
The overall computational cost for a single client can be given by:

$$\text{FLOPs per data sample} \times \text{size of client data} \times \text{number of local epochs} \times \text{number of federated rounds}. \tag{2}$$

In our experiments, the number of clients, size of client data, and number of local epochs are fixed. The client models are trained on local data for one epoch per federated round. We observe that our method generally has a similar convergence rate as the competing methods (all converge within 80-100 federated rounds). Therefore, one can see the raw communication and computational costs per example per iteration as comparable metrics.

3. Additional Results

As stated in the manuscript, we perform two other DA tasks on the Office-Caltech10 and BDD100K datasets. We show the additional results here to demonstrate that our method performs well on diverse image classification and semantic segmentation datasets. Some examples in these datasets are shown in Figure 1.



(a) Office-Caltech10 dataset **(b) GTA5 (left) and BDD100k (right) datasets**

Figure 1. **Examples in the Office-Caltech10, GTA5, and BDD100K datasets.**

3.1. Image classification: Office-Caltech10

In the Office-Caltech10 experiment, we take turn using one domain as source and the rest as target domains. Although the domain gaps in this dataset are not as significant as the others, the limited amount of target data still makes it challenging to achieve a large performance gain from the baselines. We show the quantitative results in Table 7.

Table 7. **Quantitative evaluations on the Office-Caltech dataset.**

Method	A $\rightarrow$ C, D, W	C $\rightarrow$ A, D, W	D $\rightarrow$ A, C, W	W $\rightarrow$ A, C, D	Avg
Source only	79.5	78.3	71.9	72.3	75.5
Cent-MCD [13]	81.2	81.8	74.7	75.0	78.2
Fed-oracle	80.4	81.0	74.1	74.0	77.4
Fed-DAN [9]	79.3	78.6	72.0	72.1	75.5
Fed-DANN [4]	79.7	79.2	71.9	72.4	75.8
Fed-MCD [13]	79.4	79.5	72.7	72.4	76.0
DualAdapt (ours)	**80.2**	**80.6**	**73.5**	**74.0**	**77.1**

3.2. Semantic segmentation: GTA5-to-BDD100K (cross-weather adaptation)

We adapt the GTA5 [12] models to street scenes taken in different weather conditions from the BDD100K dataset [15]. We use the dataset processed by Liu *et al.* [8], where the images with dense segmentation annotations are grouped into four domains: cloudy, overcast, rainy, and snowy given their weather labels. Each of the four clients owns the training images from one weather domain, and accuracy is measured by averaging over their testing images. We adopt the same model architecture and input resolution as in the CrossCity experiment but with 19 classes. Table 8 shows quantitative results. Despite the large inter-client domain gaps, DualAdapt achieves higher accuracy than the baselines and approaches the oracle performance.

Table 8. **Quantitative evaluations on GTA5-to-BDD100K (cross-weather adaptation).**

Method	Cloudy	Overcast	Rainy	Snowy	Average
Source only	26.1	25.4	21.4	21.5	23.6
Cent-MCD [13]	29.4	27.8	23.3	22.7	25.8
Fed-oracle	27.9	27.3	22.8	22.6	25.2
Fed-DAN [9]	26.2	25.0	21.5	21.3	23.5
Fed-DANN [4]	26.5	25.1	21.8	21.9	23.8
Fed-MCD [13]	26.7	25.8	21.4	21.7	23.9
DualAdapt (ours)	**27.1**	**26.8**	**22.7**	**22.3**	**24.7**

4. Ablation Studies

4.1. Hyper-parameters

In our framework, the hyper-parameters include self-training loss weight λ_{st}, GMM feature dimension d, and number of GMM components N_c. We report the hyper-parameters used in our experiments in Table 9 and their sensitivity analyses in Tables 10, 11, and 12, respectively. We observe that the model performance is not sensitive to the hyper-parameters. Increasing the values of d and N_c results in slightly higher accuracy but requires more communication overhead and client computational cost.

Table 9. **Hyper-parameters used in our experiments.**

Parameter	Notation	Digit-Five	Office-Caltech	DomainNet	CrossCity	BDD100K
ST loss weight	λ_{st}	0.1	0.1	0.1	0.1	0.1
GMM dimension	d	16	64	64	64	64
# GMM components	N_c	20	20	345	26	38
Client learning rate	-	0.01	0.01	0.001	0.001	0.001
Server learning rate	-	0.001	0.001	0.001	0.001	0.001

Table 10. **Analysis on the self-training (ST) loss weight λ_{st} in the Digit-Five experiment.**

λ_{st}	MNIST-M	SVHN	Synthetic	USPS	Average
0.01	**27.7**	11.1	27.4	67.9	33.6
0.1	**27.7**	**11.9**	**28.0**	68.9	**34.1**
1.0	27.3	11.6	26.7	**69.3**	33.7

Table 11. **Analysis on the reduced feature dimension d of GMM in the Digit-Five experiment.**

d	MNIST-M	SVHN	Synthetic	USPS	Average
8	27.0	10.8	27.4	68.1	33.3
16	27.7	**11.9**	28.0	68.9	34.1
32	**28.0**	11.7	**28.6**	**69.2**	**34.4**

Table 12. **Analysis on the number of GMM components N_c in the Digit-Five experiment.**

N_c	MNIST-M	SVHN	Synthetic	USPS	Average
10	27.5	11.8	27.7	68.2	33.8
20	27.7	**11.9**	**28.0**	68.9	34.1
40	**28.0**	**11.9**	27.6	**69.4**	**34.2**

4.2. Ablation studies on CrossCity

In addition to the ablation studies on Digit-Five, we conduct a similar evaluation on the CrossCity dataset to demonstrate the effectiveness of each component in DualAdapt. We show the ablative results in Table 13.

Table 13. **Ablation studies on the GTA5-to-CrossCity experiment.** In the Fed-MCD baseline [13], each client updates the feature extractor on both target and source data, resulting in two forward and two backward passes. Our full method requires only one forward passes of the feature extractor on client devices, which significantly reduces the communication and computational costs.

Method	Client	Server	Accuracy (mIoU)	Computation (FLOPS)	Communication (# parameters)
Fed-MCD [13]	MCD target	-	27.6	32.6B	46.1M + 46.1M
DualAdapt	MCD target	MCD mixup	28.1	**8.4B**	**1.5M + 46.1M**
DualAdapt	MCD target + ST	MCD mixup	28.4	**8.4B**	1.5M + 46.2M
DualAdapt	MCD target + ST	MCD mixup + GMM	**28.9**	**8.4B**	1.6M + 46.2M
Fed-oracle	MCD target + ST	MCD target	29.4	8.4B	1.5M + 46.1M

4.3. Various amount of target data

In Table 14, we report the classification accuracy using different amount of target data for training. With sufficient training data per target domain, the one-to-one adaptation approach performs competitively. However, in a practical scenario where each client possesses limited data, the one-to-multiple setting achieves higher accuracy by exploiting data from different domains. It justifies our FMTDA setting and demonstrates the effectiveness of DualAdapt in this data-limited scenario.

Table 14. **Target accuracy using various amount of training data per domain in the Digit-Five experiment.**

Method	DA setting	100% (25k)	10% (2.5k)	1% (250)
Source only	-	30.3	30.3	30.3
Cent-MCD [13]	one-to-one	**40.0**	33.5	30.4
Cent-MCD [13]	one-to-combined	38.2	34.0	30.8
Cent-MCD [13]	one-to-multiple	39.7	**35.2**	**31.0**
Fed-DAN [9]	one-to-multiple	33.1	30.9	29.8
Fed-DANN [4]	one-to-multiple	33.5	31.3	30.3
Fed-MCD [13]	one-to-multiple	34.2	31.7	30.4
DualAdapt (ours)	one-to-multiple	**37.0**	**34.1**	**30.6**

4.4. Inference strategies

To investigate the effectiveness of our inference strategy, we report the accuracy of global classifier, local classifier, and the ensemble of both. As shown in Table 15, the ensemble strategy performs the best since it allows customized model prediction for each client while constrained by the global model.

Table 15. **Target accuracy using different inference strategy in the Digit-Five experiment.**

Global classifier	Local classifier	Ensemble
33.6	33.3	**34.1**

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016. 1

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 1

[3] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *ICCV*, pages 1992–2001, 2017. 1, 2

[4] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189, 2015. 3, 5

[5] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, pages 2066–2073, 2012. 1, 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015. 1

[8] Ziwei Liu, Zhongqi Miao, Xingang Pan, Xiaohang Zhan, Dahua Lin, Stella X Yu, and Boqing Gong. Open compound domain adaptation. In *CVPR*, pages 12406–12415, 2020. 3

[9] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, pages 2208–2217, 2017. 3, 5

[10] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, pages 1406–1415, 2019. 1, 2

[11] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. *arXiv preprint arXiv:1911.02054*, 2019. 1, 2

[12] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, pages 102–118, 2016. 1, 2, 3

[13] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, pages 3723–3732, 2018. 3, 5

[14] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. 1

[15] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, pages 2636–2645, 2020. 1, 2, 3