Supplementary Material for "Generalized Semi-Supervised Learning via Self-Supervised Feature Adaptation"

In the Appendix, we provide algorithms of our SSFA in Section A. Section B presents the detailed experimental setting. Section C provides complete experimental results on CIFAR100, OFFICE-31 and OFFICE-HOME benchmarks. Finally, we conduct more ablation studies in Section D.

A Algorithm

Algorithm 1 summarizes the proposed SSFA method. For easy understanding, the framework 1 of SSFA is also shown.

Algorithm 1 Algorithm SSFA

Input: labeled batch $\{(x_b, y_b)\}_{b=1}^{B}$, unlabeled batch $\{(u_b)\}_{b=1}^{\mu B}$, the network parameter: θ_g for the shared encoder, θ_c for the main task head, and θ_s for the auxiliary task head, total number of training iterations T
for $t \leftarrow 1$ **to** T **do**
  for $i = 1$ to k **do**
    Update θ_g to $\theta_g^{'}$ for $\{u_b\}_{b=1}^{\mu B}$ based on $\mathcal{L}_{apt}$ in Equation 5
  end for
  Use $(\theta_g^{'}, \theta_c)$ to generate refined pseudo-label $\hat{q}_b'$ in Equation 6
  Use $\hat{q}_b'$ to calculate the unsupervised loss $\mathcal{L}_u$ in Equation 7
  Compute the supervised loss $\mathcal{L}_x$ in Equation 1
  Compute the auxiliary loss $\mathcal{L}_{aux}$ in Equation 3
  Compute the total loss $\mathcal{L}_{SSFA}$ and update $(\theta_g, \theta_c, \theta_s)$ in Equation 4
end for
Output: optimized parameters (θ_g, θ_c) for evaluation

B Experimental Setting

B.1 Datasets

CIFAR100-C. Using the method of generating ImageNet-C [5], we construct a corruption variant of CIFAR100 [6], CIFAR100-C. Let CIFAR100 denote the clean dataset and CIFAR100-C denote the corrupted dataset. The labeled samples are drawn from CIFAR100 without corruptions, while the unlabeled corrupted samples are drawn from the mixed dataset composed of CIFAR100 and CIFAR100-C.

In the test phase, we make evaluations on the distributions of *labeled* data and *unlabeled* data respectively. To further test the generalization of the model, we evaluate the distributions that have not been encountered during the training process, namely *unseen* distributions.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

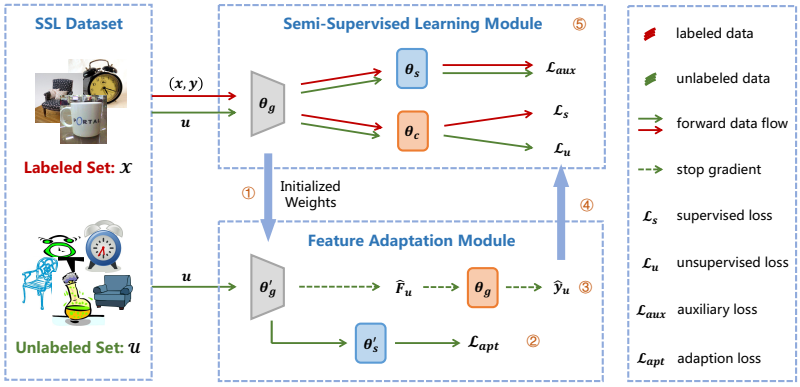

Figure 1: Architecture of SSFA. SSFA consists of two modules: a Semi-Supervised Learning Module that supplements traditional SSL learners with a self-supervised auxiliary task, and a Feature Adaptation Module that provides a higher-quality pseudo-label predictor through self-supervised feature adaptation.

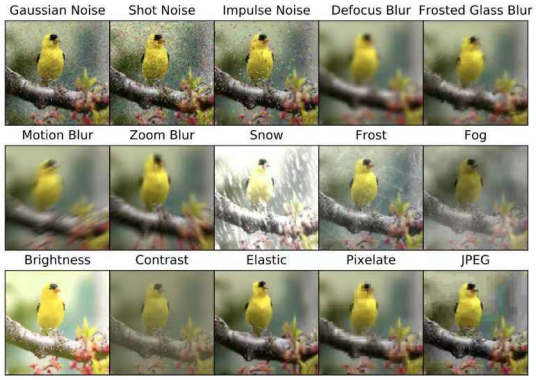

Figure 2: Visualization of distribution mismatch caused by image corruption. The labeled images are uncorrupted images, while the unlabeled data may suffer these corruptions.

As shown in Figure 2, the unlabeled samples are obtained from the mixture of ten corrupted distributions, including "gaussian_noise", "shot_noise", "impulse_noise", "glass_blur", "motion_blur", "frost", "fog", "elastic_transform", "pixelate" and "jpeg_compression". The unseen samples are generated from the mixture of five corrupted distributions that have not been seen during training, including "defocus_blur", "zoom_blur", "snow", "brightness" and "contrast".

In addition, we use a hyperparameter $ratio$ to control the proportion of corrupted and clean distributions in the unlabeled dataset. For example, $ratio$ 0.8 means the proportion of corrupted samples to the total unlabeled samples is set to 0.8.

OFFICE-31. OFFICE-31 [8] contains three domains: "Amazon" (A), "DSLR" (D), and "Webcam" (W). Each domain contains 31 classes. In the experiment, we use six combinations of "labeled data domain/unlabeled data domain/ unseen data domain" including "A/D/W", "A/W/D", "D/A/W", "D/W/A", "W/A/D", and "W/D/A". Throughout the experiment, we assess the performance of our model across three domains: labeled, unlabeled, and unseen. Take "A/D/W" task as an example, we test the model's accuracy on the labeled domain A, the unlabeled domain D, and the unseen domain W.

To further verify the general applicability of our method, we conduct experiments involving mixed distribution of unlabeled data fields, specifically "A/DW", "D/AW", and "W/AD". During the training phase, for each combination, we select 155 labeled instances from the labeled domain, *i.e.*, 5 labeled instances per class. Additionally, we sample 310 unlabeled instances from each unlabeled domain, *i.e.*, 10 unlabeled instances per class for each unlabeled domain. In cases where a class has fewer than 10 samples, we up-sampled the data to ensure an equal number of instances.

In the testing phase, we construct the labeled test set using the remaining instances from the labeled domain. Similarly, we create an unlabeled test set for each unlabeled domain using the remaining instances from their respective domains. Additionally, we form an unseen test set consisting of all instances from the unseen domain. This comprehensive testing setup allows us to evaluate our method across labeled, unlabeled, and unseen data distributions.

OFFICE-HOME. As shown in Figure 3, OFFICE-HOME [10] contains four domains: "Art" (A), "Clipart" (C), "Product" (P) and "RealWorld" (R), and each domain is composed of 65 classes. Similar to the OFFICE-31 experiments, we use four combinations for mixed test distributions of unlabeled data, including "A/CPR", "C/APR", "P/ACR" and "R/ACP". We also add samples from the labeled domain to the unlabeled data to explore the impact of the proportion of labeled domain data in the unlabeled data, namely "A/ACPR", "C/ACPR", "P/ACPR" and "R/ACPR". In the training phase, for each combination, we choose 325 labeled instances from the labeled domain, i.e., 5 labeled instances per class, and sample 1300 unlabeled instances from each unlabeled domain, i.e., 20 unlabeled instances per class. In the test phase, we construct the labeled test set and the unlabeled test set with the remaining instances respectively.

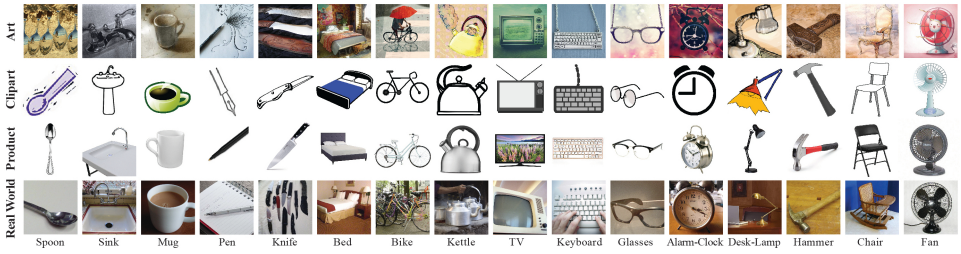

Figure 3: Visualization of distribution mismatch caused by style change. We select samples of a certain style as labeled data, while unlabeled data may consist of multiple samples of different styles. For example, selecting the "Art" samples as labeled data, unlabeled data can be composed of one or more of the following domains: "Art", "Clipart", "Product", or "Real World".

B.2 Implementation Details

CIFAR-100-C. In the experiment, we adjust the ratio of unlabeled samples from labeled distributions for training. We use WRN-28-8 [11] as the backbone except for [2] where we use WRN-28-2 to prevent training collapse. All of the methods share the same set of hyper-parameters. The threshold τ is set to 0.95. We use SGD with a momentum of 0.9 as the optimizer. The initial learning rate is set to 0.03 with a cosine learning rate annealing scheduler. The batch size of labeled data is set to 64 and the batch size of unlabeled data is set to 128. The model is trained for 100 epochs from scratch, with 1024 iterations per epoch. We use the exponential moving average with the momentum of 0.999 of the training model to make predictions on the test dataset.

OFFICE-31 and OFFICE-HOME. We use ResNet-50 [4] pre-trained on ImageNet [3] as the backbone. The thresholds τ on OFFICE-31 and OFFICE-HOME are set at 0.80 and 0.95 respectively. We use SGD with a momentum of 0.9 and weight decay of $1e^{-3}$ as the optimizer. The initial learning rates are set to $5e^{-4}$ on OFFICE-31 and $3e^{-3}$ on OFFICE-HOME with the cosine learning rate annealing scheduler. The batch size of both labeled and unlabeled data are set to 64. The experiments are set to run 5000 iterations in total.

C Main Results

To assess the impact of feature distribution mismatch between labeled and unlabeled images, we consider two factors: image corruption and style change. In the case of the CIFAR100 dataset, the unlabeled images demonstrate corruption, while in the OFFICE-31 and OFFICE-HOME datasets, the unlabeled data showcases style change. Due to space constraints, we presented only partial experimental results in Table 3 of the main paper. In this section, we provide the complete experimental results in Tables 1 and Table 2.

Table 1: Comparison of accuracy (%) for Feature Distribution Mismatch SSL on OFFICE-31.

(a) Single unlabeled domain

Method	A/D			A/W			D/A		
	L	UL	US	L	UL	US	L	UL	US
supervised	65.9	57.1	52.0	65.9	47.8	61.0	**93.0**	56.8	84.9
DANN	53.9	17.9	20.1	15.3	7.2	4.6	74.6	4.8	16.7
CDAN	2.9	5.6	2.6	2.9	2.3	4.2	4.7	2.9	2.6
FixMatch	58.4	46.9	53.3	56.3	9.5	19.3	82.8	4.7	50.6
FM-Rot	60.0	53.1	57.1	54.6	27.4	19.3	70.9	29.5	50.6
FM-SSFA	**67.5**	**61.7**	**63.0**	**66.4**	**56.5**	**62.5**	**93.0**	**59.3**	**85.9**

Method	D/W			W/A			W/D		
	L	UL	US	L	UL	US	L	UL	US
supervised	93.0	84.1	57.0	**93.6**	56.2	92.4	**93.6**	**91.9**	**56.1**
DANN [1]	4.7	2.3	2.9	2.5	2.9	4.2	80.9	16.8	26.8
CDAN [7]	4.7	2.3	2.9	41.3	7.7	18.5	1.7	3.1	1.3
FixMatch [9]	88.9	84.3	49.3	78.3	9.4	62.7	87.7	91.4	50.1
FM-Rot	89.5	**89.7**	56.4	75.9	17.9	66.9	86.7	86.8	48.7
FM-SSFA	**93.3**	86.2	**58.5**	93.0	**56.4**	**92.8**	93.4	**91.9**	54.7

(b) Multi unlabeled domains

Method	A/DW		D/AW		W/AD	
	L	UL	L	UL	L	UL
supervised	68.0	51.8	**93.3**	60.7	**93.4**	**57.8**
DANN [1]	28.1	7.6	4.7	2.8	66.7	12.4
CDAN [7]	2.9	3.2	45.5	10.7	2.5	3.1
FixMatch [9]	55.0	48.8	80.2	40.3	72.7	31.2
FM-Rot	54.0	40.1	63.0	26.9	70.6	30.9
FM-SSFA	**69.1**	**60.8**	92.4	**61.8**	92.2	57.4

Table 2: Comparison of accuracy (%) for Feature Distribution Mismatch SSL on OFFICE-HOME.

Method	A/CPR			C/APR			P/ACR			R/ACP		
	L	UL	UU	L	UL	UU	L	UL	UU	L	UL	UU
supervised	48.4	39.9	-	40.9	31.2	-	68.4	36.5	-	60.6	37.2	-
DANN [1]	45.0	37.8	-	31.8	7.7	-	53.9	25.6	-	41.7	24.6	-
CDAN [7]	30.4	22.8	-	1.2	1.8	-	63.0	34.3	-	52.2	35.9	-
FixMatch [9]	27.2	17.7	19.3	28.1	19.6	19.4	55.6	24.9	30.6	23.9	15.0	17.5
FM-Pre	21.2	15.0	15.4	28.6	22.9	25.4	51.9	22.0	26.3	2.3	1.8	0.0
FM-Rot	1.7	2.1	0.0	33.4	20.1	0.7	43.4	14.6	17.2	33.7	20.7	24.1
FM-SSFA	**50.7**	**44.0**	**69.0**	**45.1**	**37.6**	**66.2**	**70.6**	**41.9**	**70.5**	**62.4**	**43.2**	**78.0**

Method	A/ACPR			C/ACPR			P/ACPR			R/ACPR		
	L	UL	UU	L	UL	UU	L	UL	UU	L	UL	UU
supervised	49.3	43.0	-	41.0	36.3	-	70.0	48.0	-	61.6	46.8	-
DANN [1]	49.3	42.0	-	30.5	20.2	-	50.7	31.2	-	44.5	32.1	-
CDAN [7]	4.4	4.2	-	10.5	6.7	-	2.3	2.1	-	13.5	10.2	-
FixMatch [9]	32.4	23.0	31.0	36.9	30.6	36.0	52.9	32.6	37.7	42.2	31.5	35.4
FM-Pre	30.6	20.1	24.8	37.9	33.8	41.7	61.7	40.9	51.0	33.4	23.8	24.9
FM-Rot	25.6	17.7	21.3	37.3	31.0	33.2	55.3	33.2	35.8	45.6	34.7	42.3
FM-SSFA	**55.0**	**45.5**	**73.2**	**44.7**	**41.7**	**66.0**	**71.8**	**52.6**	**78.6**	**64.8**	**52.7**	**80.3**

Table 3: The effectiveness of Feature Adaptation Module on CIFAR100.

(a) mix 10-$ratio$ 1.0-level 5

Method	L	UL
FixMatch[9]	15.7	3.5
FM-Rot	18.4	2.9
FM-SSFA	**25.7**	**22.2**

(b) mix 10-$ratio$ 0.5-level 5

Method	L	UL
FixMatch[9]	25.8	4.7
FM-Rot	**37.9**	5.0
FM-SSFA	37.0	**23.2**

Table 4: The impact of the parameters optimized in Feature Adaptation Module on OFFICE-HOME.

Method	A/CPR			C/APR			P/ACR			R/ACP		
	L	UL	UU	L	UL	UU	L	UL	UU	L	UL	UU
FM-shared	48.9	39.9	**70.8**	**46.1**	**39.5**	**69.1**	70.4	**42.0**	67.3	61.0	40.8	67.0
FM-all	**50.7**	**44.0**	69.0	45.1	37.6	66.2	**70.6**	41.9	**70.5**	**62.4**	**43.2**	**78.0**

Method	A/ACPR			C/ACPR			P/ACPR			R/ACPR		
	L	UL	UU	L	UL	UU	L	UL	UU	L	UL	UU
FM-shared	51.2	43.7	**73.9**	**47.2**	**43.6**	**73.0**	70.8	**52.9**	**79.2**	63.1	50.9	77.1
FM-all	**55.0**	**45.5**	73.2	44.7	41.7	66.0	**71.8**	52.6	78.6	**64.8**	**52.7**	**80.3**

D Ablation Study

The effectiveness of Feature Adaptation Module. To evaluate the effectiveness of the Feature Adaptation Module in the SSFA framework, we compare FM-SSFA with a baseline method FM-Rot, where we remove the Feature Adaptation Module and only add the self-supervised rotation prediction task. As shown in Table 3, FM-SSFA largely outperforms FM-Rot on OFFICE-31 and OFFICE-HOME, especially in the UL evaluation metric. The superiority of FM-SSFA highlights that the Feature Adaptation Module helps the model to adapt to unlabeled samples from different distributions, resulting in better generalization on unlabeled domain. Moreover, we can observe the FM-Rot is only marginally better than FixMatch and, in some cases, may even perform worse. These results suggest that simply integrating the self-supervised task into SSL methods only brings limited performance gains and may even be detrimental in some scenarios.

The impact of the parameters optimized in the Feature Adaptation Module. We analyze the impact of the parameters optimized in the Feature Adaptation Module. In Table 4, "FM-shared" indicates that only the shared parameters (θ_g) are updated and "all" indicates that the whole parameters of the auxiliary task (θ_g, θ_s) are updated. As shown in Table 4, the differences between updating the shared parameters and updating the whole parameters can be ignored.

Table 5: The impact of shared layers between auxiliary and main task on OFFICE-HOME. "X layers" denotes the number of shared layers for the feature extractor.

Method	A/CPR			C/APR			P/ACR			R/ACP		
	L	UL	UU	L	UL	UU	L	UL	UU	L	UL	UU
FM-SSFA(2 layers)	**48.9**	**39.9**	70.8	**46.1**	**39.5**	69.1	70.4	42.0	67.3	61.0	40.8	67.0
FM-SSFA(3 layers)	47.5	39.8	**71.0**	45.4	39.0	**77.6**	**72.4**	**45.4**	**74.4**	**63.4**	**43.9**	**76.8**
FM-SSFA(4 layers)	44.4	39.1	57.1	1.3	2.4	0.0	57.9	33.4	59.6	40.3	28.7	54.3

Method	A/ACPR			C/ACPR			P/ACPR			R/ACPR		
	L	UL	UU	L	UL	UU	L	UL	UU	L	UL	UU
FM-SSFA(2 layers)	51.2	43.7	73.9	**47.2**	43.6	**73.0**	70.8	52.9	79.2	63.1	50.9	77.1
FM-SSFA(3 layers)	**52.9**	**44.9**	**80.4**	46.2	40.6	68.7	**75.0**	**54.3**	**85.2**	**64.2**	**52.2**	**82.8**
FM-SSFA(4 layers)	51.1	44.1	71.8	46.9	**44.3**	63.2	2.3	2.1	0.0	2.3	2.1	0.0

The number of shared layers between auxiliary and main task. As shown in Table 5, the performance difference is negligible between sharing 2 and 3 layers in the feature extractor, while a significant performance degradation happens if we share all layers (4 layers) of the feature extractor in some cases. We argue that the main reason is that the excessive sharing of parameters in the feature extractor can lead to over-adaptation and compromise the performance of the main task, leading to more erroneous predictions of pseudo-labels. Self-supervised learning leverages structured supervisory information to train the network and learn valuable representations for downstream tasks. Typically, the shallow layers of the feature extractor capture low-level information that is independent of the specific task, while the top layers extract high-level semantic information about the specific task. Therefore, sharing proper layers of the feature extractor can facilitate the extraction of richer feature representation. On the contrary, if the entire feature extractor is shared between the two tasks, the extracted features mainly work for the auxiliary task rather than the main task. This can lead to inaccurate predictions of pseudo-labels and compromise the performance of the main task.

More optimization steps in Feature Adaptation Module. We further explore multi-step adaptation during the optimization process in feature adaptation stage. As shown in the Table 6, only one step of adaptation can bring significant improvements over the baseline. As the number of adaptation steps increases, the performance will be further improved, but with greater computational costs. Considering the trade-off between performance and computational cost, we perform one-step optimization in our paper.

Table 6: The impact of optimization steps in Feature Adaptation Module on CIFAR100.

Method	L	UL	US
FixMatch [9]	15.7	3.5	8.5
FM-SSFA (1 step)	25.7	**22.2**	22.5
FM-SSFA (5 steps)	26.2	14.9	16.0
FM-SSFA (10 steps)	**41.1**	18.7	**33.7**

References

[1] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv*, 2014.

[2] David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[5] Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

[6] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Totonto, 2009.

[7] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2018.

[8] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, 2010.

[9] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020.

[10] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017.

[11] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.