Domain-Agnostic Feature Modulation for Semi-Supervised Domain Generalization

Supplementary Material

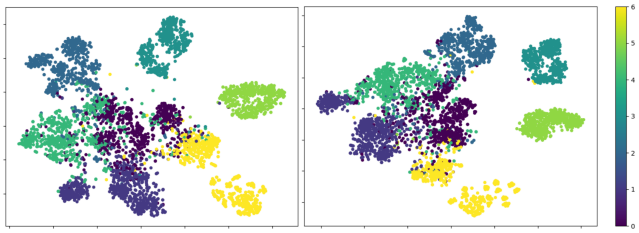

Figure 5. t-SNE visualization of features of three source domains (Cartoon, Photo, and Sketch) in the PACS dataset after training. The separation has gained improvements for class labels 0, 2, 4, and 5. (dog, giraffe, horse, and house respectively). **Left**: Features from feature extractor in FixMatch. **Right**: Modulated features in our method.

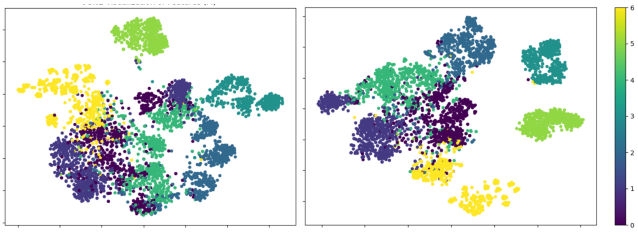

Figure 6. t-SNE visualization of features of three source domains (cartoon, photo, and sketch) in the PACS dataset at the beginning and end of the training. Overall the separation has gained improvements for all class labels. With the training iterations, the model has sufficiently removed the domain-wise separation in the above classes, achieving better domain generalization. **Left**: Modulated features at the beginning of training after the first iteration. **Right**: Modulated features at the end of training after the last iteration.

A. Comparison with SOTA

Table 6 provides a comparison of our method with SOTA methods of SSDG. The SSDG problem we address is unique as it does not require domain labels for training. We have effectively reduced the confidence threshold for pseudo-labels from 0.95 to 0.75 in order to achieve better unlabeled data utilization together with better average accuracy, as in Fig.2 . Loss scaling is one of the major contributions we introduce in our proposed method to leverage more unlabeled data for model training. FBCSA [11] uses domain-aware class prototypes, a variant of class prototypes related to each domain. In our work, we use class prototypes irrespective of the domain. The dimensions of $S_i, z, \mathbf{Z}, \mathbf{M}, \mathbf{Z}_m$ are $C \times C$, 1×512, $C \times 512$, 1×512, $C \times 512$, $C \times 512$ respectively.

B. Improvement in Class Separation

From the t-SNE visualization in Fig. 5, we can empirically show the improvement in class separation of the proposed method compared to the baseline, FixMatch [27] after all training iterations. The figure shows the features of all three source domains—cartoon, photo, and sketch—of the PACS [20] dataset when the target domain is art painting. For each class, the separation between three domains is visible, especially for class labels 0, 2, 3, and 4 in Fig. 5, Left. Our method in Fig. 5, Right, has reduced the domain-wise separation for better domain generalization. The features of class 0 overlap with its nearby classes in Fig. 5, Left. Our method has sufficiently reduced that overlapping between nearby classes, leading to better class separation.

Fig. 6 shows the learning of the proposed method with training iterations. Initially, the domain separation within each class is high, with clearly visible three clusters for each class (Fig. 6 Left). These separate clusters belong to separate three domains Our method in Fig. 6 Right, has sufficiently reduced this domain separation within classes to achieve better domain generalization. The closely located four classes (0, 1, 4, and 6) has achieved a notable separation in our method.

C. Computational complexity

We compared the training efficiency of our method with the existing SSDG methods [27, 45]. In StyleMatch [45], they transfer the styles of one domain to other domain images during the training and consider style-modified images as an augmentation method. This style augmentation is implemented using AdaIN. StyleMatch adds a 200% additional overhead to FixMatch due to these style augmentations but our method only adds a small overhead of 10%. We report the calculations using the average time computed for five seeds.

As shown in Tab. 10, the distribution of learnable parameters across different modules indicates that the feature extractor ($\mathcal{F}$) accounts for the majority of parameters, while the feature modulator ($\mathcal{M}$) introduces only a small overhead relative to the total size of the model. The classifier ($\mathcal{C}$) scales with the number of classes in each dataset, but its parameter count remains relatively modest compared to the feature extractor. This analysis highlights that the proposed feature modulation adds minimal computational cost while providing the intended benefits in suppressing domain-specific information.

	FixMatch	StyleMatch	FBCSA	DGWM	Ours
Domain labels used	✗	✔	✔	✔	✗
Confidence threshold	0.95	0.95	0.95	0.95	0.75
Loss scaling used	✗	✗	✗	✗	✔
Prototypes used	✗	✗	Domain-aware class prototypes	✗	Class prototypes

Table 6. Comparison of our method with SOTA.

Domain	# of Samples	Ratio	Highest/Lowest
Caltech	809	62	13.0
LabelMe	1124	39	28.9
Pascal	1394	307	4.6
SUN	1175	19	61.9

Table 7. Sample distribution and ratio statistics across domains in VLCS.

Method	OfficeHome			PACS		
	Accuracy	Keep Rate	PL Acc.	Accuracy	Keep Rate	PL Acc.
PL without FM	60.7	66.4	89.8	78.3	93.9	**93.2**
FM (Ours)	**61.0**	**67.6**	**90.4**	**78.7**	**94.3**	93.0

Table 8. Impact of feature modulator during pseudo labeling on OfficeHome and PACS datasets.

D. Description of Fig.1 - Feature Modulation & SAR generation

The Fig.1 **A** shows the feature space with class prototypes for each class separated by boundary lines. **B** shows how SARs are generated. Each class prototype contributes to the SAR depending on how far it is. The glow associated with each class prototype shows the cosine similarity of that class to the class c. The closest class prototypes have the highest glow intensity. **C** is the feature space with all the SARs denoted by green squares. The extracted features are shown as the instance-specific features, z. In **D**, we modulate z toward every SAR by a weight that is learnt during training. We represent these modulated features in gray color stars.

E. Why are domain labels a luxury?

The field of semi-supervised learning has emerged due to the practical challenge of obtaining completely labeled data, as acquiring class labels is often a luxury. Similarly, the presence of domain labels is a luxury because the process of getting domain labels is equally difficult. A t-SNE or a clustering of domain generalization datasets does not separate domains clearly. As shown in 5, t-SNE does not achieve clear domain separation for some classes, and domain mixing further complicates separation. Hence, we consider the presence of domain labels as a luxury, which most of the SOTA SSDG methods rely on [11, 12]. We propose an SSDG method which does not require domain labels.

Algorithm 1 Pseudo-code

Require: Labeled data (x_i , y_i) , Unlabeled data (u_i,), Weakly augmented $\alpha(x_i)$, Strongly augmented $\mathcal{A}(u_i)$, Confidence threshold τ, Total epochs E, Variance init. $\mathbf{V}$ (Sec.3.1.2), Loss scaling function: $\mathrm{Q}(x) = \exp(x^3 - 1)$, Feature extractor: f, Feature modulator: $\mathcal{M}$, Classifier: $\mathcal{C}$, Model: $F = \mathcal{C}(\mathcal{M}(f))$

Initialize $\mathbf{M} = \mathbf{1} - \frac{\mathbf{V} - \min(\mathbf{V})}{\max(\mathbf{V}) - \min(\mathbf{V})}$, Modulating matrix
Define Feature Modulator $\mathcal{M}(z, \mathbf{R})$
$\mathbf{Z}$ = Replicate z by number of classes C
$\mathbf{Z}_m = \mathcal{M}(\mathbf{Z}) = \mathbf{M} \odot \mathbf{Z} + (\mathbf{1} - \mathbf{M}) \odot \mathbf{R}$
for epoch = 1 to E **do**
    $\mathbf{P}$ = Compute class prototypes using Eq.2
    #Compute similar average representations $\mathbf{R}$
    Sim = cosine similarity matrix between each $\mathbf{P_c}$
    $\mathbf{R_c} = \frac{\sum_{j=1}^{C} \mathrm{Sim}_{cj} \cdot \mathbf{P_j}}{\sum_{j=1}^{C} \mathrm{Sim}_{cj}}$
    $\mathbf{R} = [\mathbf{R_1}, \mathbf{R_2}, \ldots, \mathbf{R_C}]$
    #Generating Pseudo-Labels
    $S_i = [\ ]$
    for k = 1 to 5 **do**
        $S_{i\mathrm{k}} = \mathrm{softmax}(\mathcal{C}(\mathcal{M}(f(u_i), \mathbf{R})))$
        $S_i = S_i \cup S_{i\mathrm{k}}$
    end for
    $D_\mu, D_\sigma = \mathrm{mean}(\mathrm{diag}(S_i)), \mathrm{std}(\mathrm{diag}(S_i))$
    $p_{\max}, \tilde{y}_i = \max(D_\mu), \mathrm{argmax}(D_\mu)$
    σ = std of $p_{\max}$
    $l_{\mathrm{scale}} = [\mathbb{1}\{p_{\max} - \sigma > \tau\} * Q(p_{\max})\ ]$
    #Compute Supervised Loss
    $S_i = \log\_\mathrm{softmax}(\mathcal{C}(\mathcal{M}(f(\alpha(x_i)), \mathbf{R})))$
    $y_{\mathrm{pred}} = \mathrm{argmax}(\mathrm{diag}(S_i))$
    $\mathrm{col}_{\max}$ = Maximum of each class in S_i
    $L_s = \mathrm{NLL}(y_{\mathrm{pred}}, y_i)$
    $L_d = \mathrm{MSE}(\mathrm{diag}(S_i), \mathrm{col}_{\max})$
    #Compute Unsupervised Loss
    $S_{ui} = \log\_\mathrm{softmax}(\mathcal{C}(\mathcal{M}(f(\mathcal{A}(u_i)), \mathbf{R})))$
    $y_{\mathrm{upred}} = \mathrm{argmax}(\mathrm{diag}(S_{ui}))$
    $\mathrm{col}_{\mathrm{umax}}$ = Maximum of each class in S_{ui}
    $L_u = \mathrm{NLL}(y_{\mathrm{upred}}, \tilde{y}_i) * l_{\mathrm{scale}}$
    $L_{ui} = \mathrm{MSE}(\mathrm{diag}(S_{ui}), \mathrm{col}_{\mathrm{umax}}) * l_{\mathrm{scale}}$
    return $L = L_s + L_u + \beta L_d + \gamma L_{ud}$
end for

Method	5 labels per class				10 labels per class			
	PACS	OfficeHome	VLCS	DigitsDG	PACS	OfficeHome	VLCS	DigitsDG
ERM	51.2±3.0	51.7±0.6	67.2±1.8	22.7±1.0	59.8±2.3	56.7±0.8	68.0±0.3	29.1±2.9
MixUp	45.3±3.8	52.7±0.6	69.9±1.3	21.7±1.9	58.5±2.2	57.2±0.6	69.6±1.0	29.7±3.1
GroupDRO	48.2±3.6	53.8±0.6	69.8±1.2	23.1±1.9	57.3±1.2	57.8±0.4	69.4±0.9	31.5±2.5
ERM + PL	62.8±3.0	54.2±0.6	65.4±2.9	43.4±2.9	63.0±1.5	55.5±0.3	60.5±1.1	55.0±2.4
MixUp + PL	60.6±2.9	51.9±0.4	60.8±2.8	35.4±1.3	62.3±1.9	55.1±0.2	64.4±1.1	43.5±1.0
GroupDRO + PL	62.3±1.9	54.5±0.5	69.3±0.3	39.4±1.3	62.1±2.0	58.5±0.3	66.5±0.2	49.9±1.9

Table 9. Comparison with the DG methods, DG+PL [27] methods under the first setting, i.e., only a few instances (5,10) are labeled from each source domain.

	Expression	OfficeHome ($c = 65$)	PACS ($c = 7$)
$\mathcal{M}$	$c \times 512$	33,280	3,584
F	11,176,512	11,176,512	11,176,512
$\mathcal{C}$	$512 \times c + c$	33,345	3,591

Table 10. Number of learnable parameters for different components of proposed method in OfficeHome and PACS datasets. Here, M, f, and C are feature modulator, feature extractor and classifier respectively

Method	Average time/epoch	Overhead
FixMatch	22.5	-
FBCSA	36.5	58.22%
StyleMatch	68.25	203.33%
DGWM	25.5	13.33%
FM (Ours)	24.75	10.00%

Table 11. Training overhead comparison over the FixMatch baseline.

F. Performance when SARs are generated only from classes with high similarity

We make the model more generalized by shifting the feature spaces towards SAR, which explained in the Sec.3.1.1. The goal is to show the worst case scenarios with closely related classes that the classifier can encounter during the testing. We compute the SAR for each class by considering the similarity of that class with other classes.

As an alternative method, we experiment our method by creating the SAR in a different way. Here we get the average of the prototypes most similar classes to a respective class. In this way we decided on a similarity threshold and get the prototypes that have a higher similarity than that threshold (set to 0.87 after testing). We validate that the SAR generation is more effective when SARs are generated using a similarity matrix using the results shown in table 3

G. Detailed description of datasets

We conducted experiments on four widely used DG datasets: PACS [20], OfficeHome [30], VLCS [29], and DigitsDG [44]. The OfficeHome [30] dataset consists of images from four distinct domains: Art, Clipart, Product, and Real-World, covering a total of 65 classes with 15,500 images. The PACS [20] dataset features images from four domains—art painting, cartoon, photo and sketch—, and it includes 9,991 images and 7 common classes in total. The VLCS [29] dataset aggregates images from four major public datasets—Pascal VOC 2007 (V), LabelMe (L), Caltech-101 (C), and Sun09 (S), with five shared classes. VLCS is useful for domain generalization as each dataset in VLCS serves as a distinct domain with different data collection sources and environments. In addition, the VLCS dataset can be used to evaluate the robustness to class imbalance. DigitsDG [44] is a synthetic domain generalization dataset with digit classification tasks across four domains: MNIST, MNIST-M, SVHN, and SYN. All of these benchmark datasets make domain generalization a more challenging task generalization due to the significant visual differences between domains such as varying color schemes and background textures.

H. Performance under class imbalance

The class imbalance is prominent in the VLCS dataset as shown in Table 7. The experiments in Table 1 with the VLCS dataset prove that our method performs well under class imbalance. Our method has a significant accuracy improvement of 5.4% and 5.4% for the setting of 5 labels and 10 labels, respectively in VLCS dataset.

LR	PACS			OfficeHome			VLCS			DigitsDG		
	$\beta = 0.5$ $\gamma = 0.5$	$\beta = 0.1$ $\gamma = 0.5$	$\beta = 1.0$ $\gamma = 0.5$	$\beta = 0.5$ $\gamma = 0.5$	$\beta = 0.1$ $\gamma = 0.5$	$\beta = 1.0$ $\gamma = 0.5$	$\beta = 0.5$ $\gamma = 0.5$	$\beta = 0.1$ $\gamma = 0.5$	$\beta = 1.0$ $\gamma = 0.5$	$\beta = 0.5$ $\gamma = 0.5$	$\beta = 0.1$ $\gamma = 0.5$	$\beta = 1.0$ $\gamma = 0.5$
0.02	77.5	78.0	78.4	61.0	61.0	60.7	75.2	75.3	**75.5**	69.1	71.9	70.1
0.03	77.9	78.1	**78.7**	60.3	60.2	**61.0**	75.1	75.2	75.5	68.7	71.4	**73.0**
0.04	78.4	78.3	78.6	60.7	60.2	60.5	75.2	75.5	75.3	68.5	71.7	72.0

Table 12. Hyperparameter tuning of learning rates and loss scales across datasets PACS, OfficeHome, VLCS, and DigitsDG.

I. Impact of feature modulation during pseudo labeling

The purpose of the feature modulation is to reduce the effect of domain-related features on the classification process. In our method we used the feature modulation during the pseudo label process to improve the accuracy of pseudo labels by mitigating the domain effect. We validate our claim by the following experiments shown in Table 8 which includes the pseudo label accuracies and keep rates in the presence and absence of feature modulation during the pseudo labeling process.

Table 8 presents an ablation study evaluating the effect of the feature modulator on pseudo-label generation. Feature modulation operates between the feature extractor ($\mathcal{F}$) and the classifier ($\mathcal{C}$). In this PL without FM setting, modulation is applied when obtaining predictions for strongly augmented unlabeled data, weakly augmented labeled data, and during testing, but not when generating pseudo-labels from weakly augmented unlabeled data. In our method, we apply feature modulation during pseudo-label generation. The results show that incorporating modulation at this stage improves both the accuracy and utilization of pseudo-labels.

J. Impact of uncertainty estimate

We propose a new addition to the pseudo-label creation as the uncertainty estimate using MC dropout. We introduce a dropout layer to the ResNet-18 [16] feature extractor after its last batch normalization layer with a dropout probability of 0.05. During the training process, we run five iterations of the complete model with feature extractor f, feature modulator $\mathcal{M}$ and classifier $\mathcal{C}$ for the same data samples and compute the mean and standard deviation of the diagonal values of the prediction matrix. We use the calculated mean and standard deviation for pseudo-label generation as described in Sec. 3.1.4. Although this iterative process generates additional overhead of 2%, it contributes to the overall accuracy improvement as described in Sec. 4. This modified pseudo-label mask is capable of improving the pseudo-label accuracy and improving the unlabaled data utilization. With the introduction of this uncertainty estimate, the pseudo-label threshold becomes adaptive based on the uncertainty of it. If the model is more certain about the class prediction, although the prediction probability is lower (around 0.75), we consider it as a correct label. On the other hand, if the uncertainty of the prediction is high, we disregard those labels despite their high prediction probability. As we introduce an adaptive threshold, we are able to reduce the fixed threshold down to 0.75.

K. Hyperparameter Tuning

We identify the learning rate of feature modulator, coefficients of labeled and unlabeled diagonal losses, β, γ respectively as the hyperparameters. First, we selected suitable values for them as in Table 12 considering the performance for OfficeHome dataset that has a lower variance. Then, we validated the performance of other datasets for the selected hyperparameters. We could gain consistent improvements for learning rate of 0.03, $\beta = 1.0$ and $\gamma = 0.5$.

L. Performance compared to DG baselines

In our method, we address the semi-supervised domain generalization problem. As mentioned in Sec.1 semi-supervised learning baselines perform better than domain generalization baselines. Here in Table 9 we compare the performance of SSDG problem setting of four common SSDG datasets when applied to DG baselines.

M. Limitations and Future Work

We identify that our method has limitations when there are vast style distributions in the data, specifically in PACS dataset. We plan to mitigate this by increasing the robustness of our method to style shifts.