A Style and Semantic Memory Mechanism for Domain Generalization — Supplementary Material

Yang Chen[†], Yu Wang[‡], Yingwei Pan[‡], Ting Yao[‡], Xinmei Tian[†], and Tao Mei[‡]

[†] University of Science and Technology of China, Hefei, China

[‡]JD AI Research, Beijing, China

cheny01@mail.ustc.edu.cn, {feather1014, panyw.ustc, tingyao.ustc}@gmail.com, xinmei@ustc.edu.cn, tmei@jd.com

Algorithm 1 Training procedure of STEAM for DG task

Input: Labeled source training domains $\mathcal{D} = \{\mathcal{D}_d\}_{d=1}^{D}$. Initialize encoders $E_f(\cdot, \boldsymbol{\theta}_{e,f})$, $E_c(\cdot, \boldsymbol{\theta}_{e,c})$ and $E_s(\cdot, \boldsymbol{\theta}_{e,s})$; memory encoders $E_{m,f}(\cdot, \boldsymbol{\theta}_{m,f})$, $E_{m,c}(\cdot, \boldsymbol{\theta}_{m,c})$ and $E_{m,s}(\cdot, \boldsymbol{\theta}_{m,s})$; classifier $C(\cdot, \boldsymbol{\phi})$; domain-specific style memory bank $\boldsymbol{V}^s$=($\boldsymbol{V}^s_1, ..., \boldsymbol{V}^s_d, ..., \boldsymbol{V}^s_D$, $d \in [1, D]$) and semantic memory bank $\boldsymbol{V}^c$. Functions enqueue and dequeue.
for $n = 1$ **to** N **do**
  Sample mini-batches of the training data $(\boldsymbol{x}_{d,i}, y_{d,i})$ and their corresponding "variant" $(\boldsymbol{x}^+_{d,i}, y_{d,i})$ *# See main texts below for description.*
  $\boldsymbol{s}_{d,i} = E_s(E_f(\boldsymbol{x}_{d,i}, \boldsymbol{\theta}_{e,f}), \boldsymbol{\theta}_{e,s})$
  $\boldsymbol{c}_{d,i} = E_c(E_f(\boldsymbol{x}_{d,i}, \boldsymbol{\theta}_{e,f}), \boldsymbol{\theta}_{e,c})$
  $\bar{\boldsymbol{s}}_{d,i} = E_{m,s}(E_{m,f}(\boldsymbol{x}^+_{d,i}, \boldsymbol{\theta}_{m,f}), \boldsymbol{\theta}_{m,s})$
  $\bar{\boldsymbol{c}}^+_{d,i} = E_{m,c}(E_{m,f}(\boldsymbol{x}^+_{d,i}, \boldsymbol{\theta}_{m,f}), \boldsymbol{\theta}_{m,c})$
  Compute $\mathcal{L}_{\text{cls}}$ on $(C(\boldsymbol{c}_{d,i}, \boldsymbol{\phi}), y_{d,i})$ using Eq. (5)
  Compute $\mathcal{L}_{\text{s}}$ on $(\boldsymbol{s}_{d,i}, \boldsymbol{V}^s)$ using Eq. (1)
  Compute $\mathcal{L}_{\text{c}}$ on $(\boldsymbol{c}_{d,i}, \bar{\boldsymbol{c}}^+_{d,i}, \boldsymbol{V}^c)$ using Eq. (4)
  Compute $\mathcal{L}_{\text{o}}$ on $(\boldsymbol{c}_{d,i}, \boldsymbol{s}_{d,i})$ using Eq. (6)
  $\text{enqueue}(\boldsymbol{V}^s, \bar{\boldsymbol{s}}_{d,i})$, $\text{enqueue}(\boldsymbol{V}^c, \bar{\boldsymbol{c}}^+_{d,i})$
  $\text{dequeue}(\boldsymbol{V}^s)$, $\text{dequeue}(\boldsymbol{V}^c)$
  Update the encoder parameters $\boldsymbol{\theta}_{e,f}$, $\boldsymbol{\theta}_{e,c}$, $\boldsymbol{\theta}_{e,s}$ and classifier parameters $\boldsymbol{\phi}$ with SGD by minimizing the total loss in Eq. (7)
  Momentum update the memory encoder parameters, $\boldsymbol{\theta}_{m,f}$, $\boldsymbol{\theta}_{m,c}$, $\boldsymbol{\theta}_{m,s}$ using Eq. (8)
end for

Algorithm 2 Training procedure of STEAM for MSDA task

Input: Labeled source training domains $\mathcal{D} = \{\mathcal{D}_d\}_{d=1}^{D}$ and unlabeled target domain $\mathcal{D}_t$. Initialize encoders $E_f(\cdot, \boldsymbol{\theta}_{e,f})$, $E_c(\cdot, \boldsymbol{\theta}_{e,c})$ and $E_s(\cdot, \boldsymbol{\theta}_{e,s})$; memory encoders $E_{m,f}(\cdot, \boldsymbol{\theta}_{m,f})$, $E_{m,c}(\cdot, \boldsymbol{\theta}_{m,c})$ and $E_{m,s}(\cdot, \boldsymbol{\theta}_{m,s})$; classifier $C(\cdot, \boldsymbol{\phi})$; domain-specific style memory bank $\boldsymbol{V}^s$=($\boldsymbol{V}^s_1, ..., \boldsymbol{V}^s_d, ..., \boldsymbol{V}^s_D, \boldsymbol{V}^s_{\mathcal{D}_t}$, $d \in [1, D]$) and semantic memory bank $\boldsymbol{V}^c$. Functions enqueue and dequeue.
for $n = 1$ **to** N **do**
  Sample mini-batches of the training data $(\boldsymbol{x}_{d,i}, y_{d,i})$ and their corresponding "variant" $(\boldsymbol{x}^+_{d,i}, y_{d,i})$ *# See main texts below for description.*
  $\boldsymbol{s}_{d,i} = E_s(E_f(\boldsymbol{x}_{d,i}, \boldsymbol{\theta}_{e,f}), \boldsymbol{\theta}_{e,s})$
  $\boldsymbol{c}_{d,i} = E_c(E_f(\boldsymbol{x}_{d,i}, \boldsymbol{\theta}_{e,f}), \boldsymbol{\theta}_{e,c})$
  $\bar{\boldsymbol{s}}_{d,i} = E_{m,s}(E_{m,f}(\boldsymbol{x}^+_{d,i}, \boldsymbol{\theta}_{m,f}), \boldsymbol{\theta}_{m,s})$
  $\bar{\boldsymbol{c}}^+_{d,i} = E_{m,c}(E_{m,f}(\boldsymbol{x}^+_{d,i}, \boldsymbol{\theta}_{m,f}), \boldsymbol{\theta}_{m,c})$
  Compute $\mathcal{L}_{\text{cls}}$ on $(C(\boldsymbol{c}_{d,i}, \boldsymbol{\phi}), y_{d,i})$ using Eq. (5) *# $\mathcal{L}_{\text{cls}}$ are only applied on labeled source samples.*
  Compute $\mathcal{L}_{\text{s}}$ on $(\boldsymbol{s}_{d,i}, \boldsymbol{V}^s)$ using Eq. (1)
  Compute $\mathcal{L}_{\text{c}}$ on $(\boldsymbol{c}_{d,i}, \bar{\boldsymbol{c}}^+_{d,i}, \boldsymbol{V}^c)$ using Eq. (4)
  Compute $\mathcal{L}_{\text{o}}$ on $(\boldsymbol{c}_{d,i}, \boldsymbol{s}_{d,i})$ using Eq. (6)
  $\text{enqueue}(\boldsymbol{V}^s, \bar{\boldsymbol{s}}_{d,i})$, $\text{enqueue}(\boldsymbol{V}^c, \bar{\boldsymbol{c}}^+_{d,i})$
  $\text{dequeue}(\boldsymbol{V}^s)$, $\text{dequeue}(\boldsymbol{V}^c)$
  Update the encoder parameters $\boldsymbol{\theta}_{e,f}$, $\boldsymbol{\theta}_{e,c}$, $\boldsymbol{\theta}_{e,s}$ and classifier parameters $\boldsymbol{\phi}$ with SGD by minimizing the total loss in Eq. (7)
  Momentum update the memory encoder parameters, $\boldsymbol{\theta}_{m,f}$, $\boldsymbol{\theta}_{m,c}$, $\boldsymbol{\theta}_{m,s}$ using Eq. (8)
end for

The supplementary material contains: 1) pseudocode of our STEAM [1], 2) analysis of the learned style feature and 3) visualization of semantic feature distributions.

1. Pseudocode of STEAM

The pseudocode of STEAM for domain generalization (DG) and multi-source domain adaptation (MSDA) are respectively presented in Algorithm 1 and Algorithm 2. To facilitate mini-batch training, $\boldsymbol{x}^+_{d,i}$ in DG (Algorithm 1) is randomly chosen from all possible domains that either is from the same class of $\boldsymbol{x}_{d,i}$, or simply is selected from the augmentation pool (RandAug [2]) of $\boldsymbol{x}_{d,i}$. In regard of MSDA (Algorithm 2), for labeled source domains, $\boldsymbol{x}^+_{d,i}$ is randomly chosen from all possible source domains that either is from the SAME class of $\boldsymbol{x}_{d,i}$, or simply is selected from the augmentation pool of $\boldsymbol{x}_{d,i}$. For unlabeled target domain, $\boldsymbol{x}^+_{d,i}$ is only selected from the augmentation pool of $\boldsymbol{x}_{d,i}$.

2. More Ablation Studies

The ablation study in the main paper (Sec. 4.3) does not separate the $\mathcal{L}_s$ and $\mathcal{L}_o$. Without the term $\mathcal{L}_o$, we would lose the only coupling constraint between style and semantic features. In this case, semantic features will independently encode each input sample without being constrained by style assumptions $\mathcal{L}_s$. Correspondingly, any invariance on domain-specific style features forced through loss $\mathcal{L}_s$ will be blocked without even reaching the semantic fea-

Table 1. Ablation on PACS for domain generaliztion. A, C, P, and S indicates *Art*, *Cartoon*, *Photo*, and *Sketch*, respectively.

$\mathcal{L}_{cls}$	$\mathcal{L}_s$	$\mathcal{L}_o$	$\mathcal{L}_c$	A	C	P	S	Avg.
√				77.0	75.9	96.1	69.2	79.5
√	√			77.1	75.8	95.7	68.8	79.3
√		√		76.5	75.3	95.9	68.9	79.1

tures, unless orthogonality constraint $\mathcal{L}_o$ is present. Certainly, $\mathcal{L}_o$ alone without $\mathcal{L}_s$ would not impose any constraint on semantic features, too. As shown in Table 1, $\mathcal{L}_{cls} + \mathcal{L}_s$ and $\mathcal{L}_{cls} + \mathcal{L}_o$ respectively returns average scores of 79.3% and 79.1% on PACS, similar to vanilla model $\mathcal{L}_{cls}$ (79.5%).

3. Style Feature Analysis

In this section, we justify STEAM's effectiveness on achieving the intra-domain style feature invariance as described in Sec 3.2. We train the network for DG task by using STEAM on PACS dataset, and we examine the similarity distributions of intra-domain and inter-domain instance style features (extracted from feature extractor E_f and style encoder E_s) using all training data. Specifically, we calculate the similarity score of any two samples belonging to the same domain or from different domains. We display the distributional histograms of style features by respectively using the blue color (inter-domain style feature similarity) and red color (intra-domain style feature similarity).

We observe that, under the STEAM training scheme, the style similarities of samples belong to the same domain (the "red" part in Fig.1) are indeed much higher than those of samples belonging to different domains (the "blue" part in Fig.1). This validates that our proposed losses are able to achieve the assumed intra-domain style invariance via domain-specific style memory banks construction, and can indeed capture the domain-specific style features.

4. Semantic Feature Visualization

In Fig.2, we visualize semantic features respectively learned by Vanilla model (Fig.2 (a)(c)) and our STEAM (feature extracted by feature extractor E_f and semantic encoder E_c, illustrated in Fig.2 (b)(d)) using t-SNE [3]. Here the source data is from PACS dataset, and the unseen target domain is *sketch*. Both of the source domain and target domain data are present in all subfigures of Fig.2. Specifically, in Fig.2 (a) and (b), different colors indicate different classes for Vanilla and STEAM; in (c) and (d), different colors represent different domains for Vanilla and STEAM.

When comparing Fig.2 (a) and (b), STEAM model obviously leads to a better class separation than Vanilla model, by well preserving the sharpened class boundaries. This implies that STEAM has captured better discriminativeness

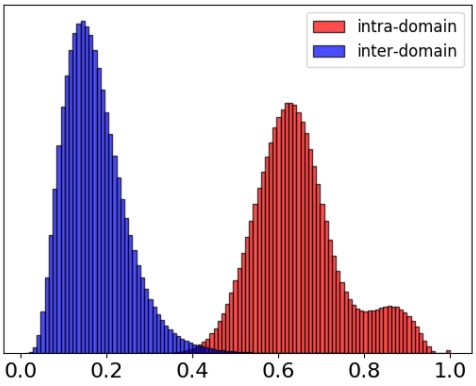

Figure 1. Qualitatively analysis of the style feature histograms learned by our STEAM. See main texts for description.

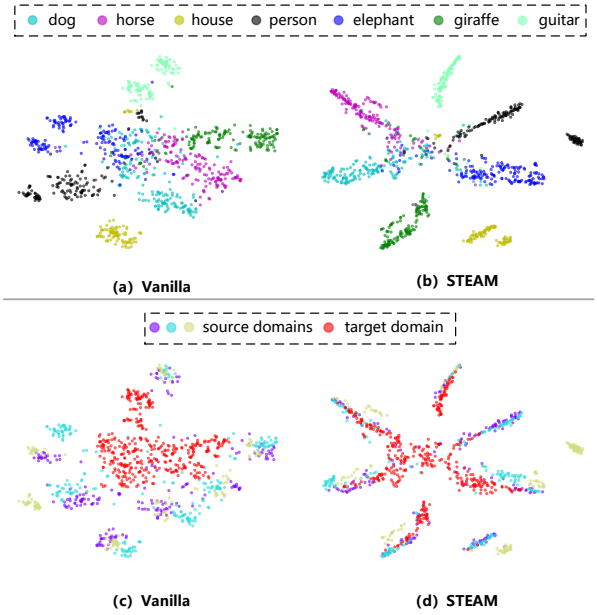

Figure 2. The t-SNE visualization of extracted semantic features out of Vanilla model (a) and (c), and features out of our proposed STEAM (b) and (d). Training source data is PACS dataset. The target domain is *sketch*. In (a) and (b), different colors indicate different classes; in (c) and (d), different colors represent different domains. See main texts for description.

on semantic features regarding the final classification task. Also, Fig.2 (c)(d) show that STEAM achieves a much better semantic feature alignment among all source and target domains than Vanilla model, even if the target data is unseen under the domain generalization (DG) setting. In comparison, the Vanilla model could hardly achieve any feature alignment between source and target domain, where test data semantic features are shown to be locating remotely from training domains (Fig.2 (c)). These t-SNE illustrations qualitatively demonstrate that our assumption of "instance level intra-domain style invariance" and the proposed "jury"

mechanism on learning "domain-agnostic" semantic features can indeed effectively disentangle semantic features from style features, thus leading to a better generalization performance than the state-of-the-art methods.

References

[1] Yang Chen, Yu Wang, Yingwei Pan, Ting Yao, Xinmei Tian, and Tao Mei. A style and semantic memory mechanism for domain generalization. In *ICCV*, 2021.

[2] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, 2020.

[3] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008.