Adversarial Data Augmentation for Single Domain Generalization via Lyapunov Exponent-Guided Optimization

Supplementary Material

A. Additional Theoretical Analysis

A.1. Complete Derivation of the Proposed Method

Gradient descent updates the model parameters as follows:

$$\theta_{t+1} = \theta_t - \eta_t \nabla L(\theta_t), \tag{1}$$

where $\theta_t \in \mathbb{R}^{m \times n}$ is a real-valued matrix with m rows and n columns. For gradient calculations, we treat θ_t as a column vector with mn rows, i.e., $\theta_t \in \mathbb{R}^{mn \times 1}$. The learning rate η_t is a real-valued scalar hyperparameter.

Introducing perturbation $\widetilde{\theta_t} := \theta_t + \delta\theta_t$, where δ is a tiny real-valued scalar hyperparameter, we obtain:

$$\widetilde{\theta}_{t+1} = \widetilde{\theta_t} - \eta_t \nabla L(\widetilde{\theta_t}). \tag{2}$$

The perturbation is propagated backward as follows:

$$\begin{aligned} \delta\theta_{t+1} &= \widetilde{\theta}_{t+1} - \theta_{t+1} \\ &= \delta\theta_t - \eta_t \left(\nabla L(\widetilde{\theta_t}) - \nabla L(\theta_t)\right). \end{aligned} \tag{3}$$

To analyze this, we perform a first-order Taylor expansion of the loss function $L(\widetilde{\theta_t})$ around θ_t, yielding:

$$L(\theta_t + \delta\theta_t) = L(\theta_t) + \nabla L(\theta_t)^T \delta\theta_t + o(\|\delta\theta_t\|^2), \tag{4}$$

where the remainder $o(\|\delta\theta_t\|^2)$ represents the higher-order infinitesimal terms that are not included. To simplify the calculation, we generally don't consider this remainder. Here, L represents a scalar function and $\nabla L(\theta_t)^T \in \mathbb{R}^{1 \times mn}$ is the partial derivative of the scalar function $L(\theta_t)$ with respect to the vector θ_t.

We take the derivative of both sides of the Equation 4 with respect to θ_t. First, we take the derivative of the left-hand side and apply the chain rule:

$$\begin{aligned} \nabla L(\widetilde{\theta_t}) &= \nabla(L(\theta_t + \delta\theta_t)) \\ &= \nabla L(\theta_t + \delta\theta_t)\nabla(\theta_t + \delta\theta_t), \end{aligned} \tag{5}$$

Simplifying this formula, we obtain:

$$\nabla L(\widetilde{\theta_t}) = \nabla L(\widetilde{\theta_t})(1+\delta)I, \tag{6}$$

where I is the identity matrix.

Next, we take the derivative of the right side of Equation 4 :$\nabla(L(\theta_t) + \nabla L(\theta_t)^T \delta\theta_t)$, and the first term is: $\nabla L(\theta_t)$. The second term $\nabla\left(\nabla L(\theta_t)^T \delta\theta_t\right)$ applies the product rule:

$$\begin{aligned} \nabla\left(\nabla L(\theta_t)^T \delta\theta_t\right) &= \nabla\left(\nabla L(\theta_t)^T\right)\delta\theta_t + \nabla L(\theta_t)\nabla\delta\theta_t \\ &= \nabla\left(\nabla L(\theta_t)^T\right)\delta\theta_t + \nabla L(\theta_t)\delta I. \end{aligned} \tag{7}$$

Combining the results of the first and second terms gives:

$$\begin{aligned} \nabla(L(\theta_t) + \nabla L(\theta_t)^T \delta\theta_t) &= \nabla L(\theta_t) + H[L(\theta_t)]\delta\theta_t + \nabla L(\theta_t)\delta I \\ &= \nabla L(\theta_t)(1+\delta)I + H[L(\theta_t)]\delta\theta_t, \end{aligned} \tag{8}$$

where $H[L(\theta_t)] \in \mathbb{R}^{mn \times mn}$ is the Hessian matrix, which is the second-order partial derivative of the real-valued function $L(\theta_t)$ with respect to the real vector θ_t.

Finally, we combine the Equation 6 and Equation 8 to get:

$$\nabla L(\widetilde{\theta_t})(1+\delta)I = \nabla L(\theta_t)(1+\delta)I + H[L(\theta_t)]\delta\theta_t. \tag{9}$$

We multiply both sides of the formula by $\frac{1}{1+\delta}I$:

$$\nabla L(\widetilde{\theta_t}) = \nabla L(\theta_t) + \frac{1}{1+\delta}H[L(\theta_t)]\delta\theta_t, \tag{10}$$

δ is a very small scalar, it can often be ignored in actual calculations. In the end, we can get:

$$\nabla L(\widetilde{\theta_t}) \approx \nabla L(\theta_t) + H[L(\theta_t)]\delta\theta_t. \tag{11}$$

We can get:

$$\nabla L(\widetilde{\theta_t}) - \nabla L(\theta_t) \approx H[L(\theta_t)]\delta\theta_t. \tag{12}$$

Then substituting Formula 12 into Equation 3, we can get:

$$\delta\theta_{t+1} = \left(I - \eta_t H[L(\theta_t)]\right)\delta\theta_t. \tag{13}$$

Then by recursion we can get:

$$\begin{aligned} \delta\theta_t = &\left(I - \eta_{t-1} H[L(\theta_{t-1})]\right)\left(I - \eta_{t-2} H[L(\theta_{t-2})]\right) \\ &\cdots \left(I - \eta_0 H[L(\theta_0)]\right)\delta\theta_0. \end{aligned} \tag{14}$$

In this way, we have established the equation for the propagation of perturbations in the model and its relationship with the learning rate and the Hessian matrix.

The definition of the Lyapunov exponent is:

$$\text{LE} = \lim_{t\to\infty} \frac{1}{t} \ln\left(\frac{\|\delta\theta_t\|}{\|\delta\theta_0\|}\right). \tag{15}$$

Equation 14 considers the 2-norm and establishes the following relationship:

$$\|\delta\theta_t\| \leq \|\delta\theta_0\| \prod_{i=0}^{t-1} \|(I - \eta_i H[L(\theta_i)])\| \,. \tag{16}$$

We also derive the following inequality:

$$\|\delta\theta_t\| \geq \|\delta\theta_0\| \prod_{i=0}^{t-1} (1 - \eta_i \|H[L(\theta_i)])\|). \tag{17}$$

Next, we use mathematical induction to prove Inequality 17. When $t = 1$, obviously we have:

To verify Inequality 17, we employ mathematical induction.

$$\begin{aligned} \|\delta\theta_1\| &= \|(I - \eta_0 H[L(\theta_0)]\delta\theta_0\| \\ &\geq \|\delta\theta_0\| - \eta_0\|H[L(\theta_0)]\delta\theta_0\| \\ &\geq \|\delta\theta_0\| - \eta_0\|H[L(\theta_0)]\|\|\delta\theta_0\|. \end{aligned} \tag{18}$$

Assume that the inequality holds for $t = k$, such that:

$$\|\delta\theta_k\| \geq \|\delta\theta_0\| \prod_{i=0}^{k-1} (1 - \eta_i \|H[L(\theta_i)])\|). \tag{19}$$

So we can get the following representation relationship:

$$\begin{aligned} \|\delta\theta_{k+1}\| &= \|(I - \eta_k H[L(\theta_k)])\delta\theta_k\| \\ &= \|\delta\theta_k - \eta_k H[L(\theta_k)]\delta\theta_k\|. \end{aligned} \tag{20}$$

Using the vector 2-norm triangle inequality, we get:

$$\|\delta\theta_k - \eta_k H[L(\theta_k)]\delta\theta_k\| \geq \|\delta\theta_k\| - \eta_k\|H[L(\theta_k)]\delta\theta_k\|. \tag{21}$$

According to the properties of the 2-norm, we can obtain:

$$\eta_k\|H[L(\theta_k)]\delta\theta_k\| \leq \eta_k\|H[L(\theta_k)]\|\|\delta\theta_k\|. \tag{22}$$

Then combining Inequality 21 and Inequality 22, we can get:

$$\begin{aligned} \|\delta\theta_k - \eta_k H[L(\theta_k)]\delta\theta_k\| &\geq \|\delta\theta_k\| - \eta_k\|H[L(\theta_k)]\delta\theta_k\| \\ &\geq \|\delta\theta_k\| - \eta_k\|H[L(\theta_k)]\|\|\delta\theta_k\|. \end{aligned} \tag{23}$$

Combining Equation 20 and Inequality 23, and substituting our results into the inductive hypothesis:

$$\begin{aligned} \|\delta\theta_{k+1}\| &\geq (1 - \eta_k\|H[L(\theta_k)]\|)\|\delta\theta_k\| \\ &\geq (1 - \eta_k\|H[L(\theta_k)]\|)\|\delta\theta_0\| \prod_{i=0}^{k-1}(1 - \eta_i\|H[L(\theta_i)]\|). \end{aligned} \tag{24}$$

We can get:

$$\|\delta\theta_t\| \geq \|\delta\theta_0\| \prod_{i=0}^{t-1}(1 - \eta_i\|H[L(\theta_i)]\|), \tag{25}$$

where we have proved that Inequality 17.

Substituting Inequality 16 into the definition of the Lyapunov Exponent (LE), the relationship between the model learning rate η nd LE can be expressed as follows:

$$\begin{aligned} \text{LE} &= \lim_{t\to\infty} \frac{1}{t}(\ln\|\delta\theta_t\| - \ln\|\delta\theta_0\|) \\ &\leq \lim_{t\to\infty} \frac{1}{t}(\sum_{i=0}^{t-1} \ln(\|I - \eta_i H[L(\theta_i)]\|) + \ln\|\delta\theta_0\| \\ &- \ln\|\delta\theta_0\|). \end{aligned} \tag{26}$$

Similarly, by substituting Inequality 17 into the definition of the Lyapunov Exponent, we derive the following bounds for LE based on the model learning rate η:

$$\begin{aligned} \text{LE} &\geq \lim_{t\to\infty} \frac{1}{t} \sum_{i=0}^{t-1} \ln(1 - \eta_i\|H[L(\theta_i)]\|) \\ \text{LE} &\leq \lim_{t\to\infty} \frac{1}{t} \sum_{i=0}^{t-1} \ln(\|I - \eta_i H[L(\theta_i)]\|)\,. \end{aligned} \tag{27}$$

A.2. Further Analysis for the Results Presented in Figure 3 of the Manuscript

To further explain the results shown in Figure 3 of the manuscript, we made certain simplifications in the computation of LE. Specifically, the exact value of the natural logarithm term, $\ln\left(\frac{\|\delta\theta_t\|}{\|\delta\theta_0\|}\right)$, used in the LE definition in Equation 15, represents a normalized value. For better visualization of the model's LE curve trend, we omit normalization, resulting in the following formulation:

$$\text{LE} = \lim_{t\to\infty} \frac{1}{t} \ln(\|\delta\theta_t\|)\,. \tag{28}$$

By substituting Inequality 16 into Equation 28, we can use the properties of logarithms to decompose the product and convert the terms into summations:

$$\text{LE} \leq \lim_{t\to\infty} \frac{1}{t} \left(\sum_{i=0}^{t-1} \ln(\|I - \eta_i H[L(\theta_i)]\|) + \ln(||\delta\theta_0||) \right). \tag{29}$$

Since $\ln(||\delta\theta_0||)$ is a constant, it can be extracted and expressed separately as:

$$\text{LE} \leq \lim_{t\to\infty} \left(\frac{1}{t} \sum_{i=0}^{t-1} \ln(\|I - \eta_i H[L(\theta_i)]\|) + \frac{\ln||\delta\theta_0||}{t} \right). \tag{30}$$

As t approaches infinity, the term $\frac{\ln||\delta\theta_0||}{t}$ converges to zero. Consequently, the final expression simplifies to:

$$\text{LE} \leq \lim_{t\to\infty} \frac{1}{t} \sum_{i=0}^{t-1} \ln\left(\|I - \eta_i H[L(\theta_i)]\|\right). \tag{31}$$

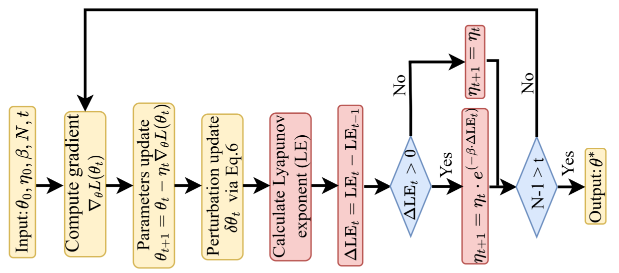
Similarly, substituting Inequality 17 into Equation 28, we can finally obtain the expression:

$$\begin{aligned} \text{LE} &\geq \lim_{t\to\infty} \frac{1}{t} \sum_{i=0}^{t-1} \ln(1 - \eta_i \|H[L(\theta_i)]\|) \\ \text{LE} &\leq \lim_{t\to\infty} \frac{1}{t} \sum_{i=0}^{t-1} \ln\left(\|I - \eta_i H[L(\theta_i)]\|\right). \end{aligned} \tag{32}$$

Here, θ_t is processed as a column vector, and $\|\theta_t\|$ denotes its L2 norm, also referred to as the Euclidean norm. θ_t represents the difference between the current parameter state and the parameter state from the previous iteration. The introduction of perturbations increases the diversity of the model's parameter distribution, causing parameters to shift in various directions and increasing the entropy of the parameter distribution. At the initial time $t + 1$, the 2-norm calculated by the unperturbed method yields a larger Euclidean norm $\|\theta_t\|$. Consequently, as indicated by Inequality 32, the Lyapunov Exponent (LE) for our method is initially higher than for other methods. As training progresses, with continual parameter updates, the overall LE curve for our approach remains higher compared to other methods.

B. LEAwareSGD Pipeline Overview

A pipeline diagram has been added to illustrate Algorithm **??** and provide an overview of the entire framework, enhancing clarity and aiding reader comprehension.



Input: $\theta_0, \eta_0, \beta, N, t$
Compute gradient $\nabla_\theta L(\theta_t)$
Parameters update $\theta_{t+1} = \theta_t - \eta_t \nabla_\theta L(\theta_t)$
Perturbation update $\delta\theta_t$ via Eq.6
Calculate Lyapunov exponent (LE)
$\Delta\text{LE}_t = \text{LE}_t - \text{LE}_{t-1}$
$\Delta\text{LE}_t > 0$
No
$\eta_{t+1} = \eta_t$
Yes
$\eta_{t+1} = \eta_t \cdot e^{(-\beta \cdot \Delta\text{LE}_t)}$
N-1 > t
No
Yes
Output: θ^*

C. Additional Experimental Results

C.1. Comparison with AdvST on DomainNet

Table 1 shows the classification accuracy (%) on the DomainNet dataset, comparing our method with AdvST [1]. Our method consistently outperforms AdvST across all domains, with improvements of 0.72% in P (Painting), 1.78% in I (Infograph), and 1.13% in C (Clipart). The average accuracy of our method is 22.44%, which is a 0.55% increase over AdvST (21.89%). These results highlight the effectiveness of our approach in enhancing domain generalization.

Methods	P	I	C	S	Q	R	Avg.
AdvST [1]	24.56	18.85	26.32	27.00	6.67	27.95	21.89
Ours	**25.28**	**20.63**	**27.45**	**27.39**	**7.17**	**28.15**	**22.44** $\pm$0.24

Table 1. Classification accuracy (%) comparison on the DomainNet dataset. The best results are in bold.

C.2. Comparison with Optimizer Approaches on OfficeHome

The results shown in Table 2 underscore the effectiveness of our LEAwareSGD optimizer for SDG. LEAwareSGD achieves the highest average accuracy of 54.38%, surpassing the best comparison optimizer, SGD, which achieves an average accuracy of 53.20%. LEAwareSGD consistently improves performance across all domains (A, C, P, and R). In contrast, adaptive optimizers such as Adam, AdamW, and RMSprop achieve significantly lower average accuracies of 48.10%, 47.81%, and 46.50%, respectively. These results suggest that adaptive learning rates may lead these optimizers to overfit on domain-specific features, reducing their generalization capacity. The consistent gains with LEAwareSGD highlight the benefits of LE guidance, which supports broader exploration of the parameter space and helps avoid overfitting. Overall, these results demonstrate that the SGD's integration of LE-guided learning provides substantial advantages over traditional optimizers, enhancing model generalization across diverse domains.

Methods	A	C	P	R	Avg.
Adam	47.48	45.68	44.82	54.41	48.10
AdamW	45.88	45.91	44.68	54.75	47.81
RMSprop	44.74	44.52	42.85	53.89	46.50
SGD	51.56	52.15	49.57	59.53	53.20
Ours	52.89	55.18	51.69	58.95	**54.38**$\pm$0.30

Table 2. Comparison results (%) using different optimizers on the OfficeHome dataset. The best results are in bold.

C.3. Comparison with Methods Employing Only Optimizers without Data Augmentation

To further validate the effectiveness of our LEAwareSGD optimizer, we removed the data augmentation module to highlight the optimizer's contribution. As shown in Table 3, the results demonstrate our method consistently outperforms SGD and Adam across different learning rates on both the PACS and OfficeHome datasets. These results further demonstrate the effectiveness of our method in the context of domain generalization.

Dataset	Optimizer	lr=0.001	lr=0.0001	lr=0.0005
PACS	Adam	23.70	59.02	26.70
	SGD	62.88	57.04	62.23
	Ours	**63.03**	**59.28**	**63.25**
OfficeHome	Adam	26.79	49.40	36.51
	SGD	53.18	49.46	53.40
	Ours	**53.79**	**50.03**	**54.15**

Table 3. Comparison results (%) using different optimizers without data augmentation on the PACS and OfficeHome datasets. The best results are in bold.

C.4. Comparison under Different Learning Rate Schedules

To further analyze the effectiveness of LEAwareSGD, we compare Adam and SGD using various learning rate schedules. Standard schedulers adjust learning rates per epoch, while our method, an optimizer, adapts them per batch. We evaluated Adam, SGD, and our method under three commonly used learning rate schedulers: StepLR (decaying to 0.1× at epoch 30), CosineAnnealingLR, and ReduceLROnPlateau (based on validation accuracy), as well as a baseline without any scheduling (denoted as Step, Cosine, Plateau, and w/o, respectively). As presented in Table 4, results on both datasets indicate that our method surpasses Adam and SGD across all configurations and maintains its advantage even in the absence of any learning rate schedule.

Dataset	Optimizer	Step	Cosine	Plateau	w/o
PACS	Adam	67.06	64.99	67.60	65.83
	SGD	66.43	67.19	66.77	66.71
	Ours	**69.46**	**69.26**	**68.55**	**69.33**
OfficeHome	Adam	48.08	47.81	49.27	47.67
	SGD	52.23	53.20	53.71	53.61
	Ours	**52.47**	**54.38**	**54.09**	**54.13**

Table 4. Comparison results (%) using different learning rate schedules on the PACS and OfficeHome datasets. The best results are in bold.

C.5. Comparison with Optimizer Approaches across Different Learning Rates

We initially set the learning rate to 0.001 for Adam, AdamW, RMSprop, SGD and our approach and further tested additional learning rates. Table 5 presents the results for PACS dataset, highlighting the optimizer's high sensitivity to the chosen learning rate, with their best performance (67.06% for SGD) remaining inferior to our method (69.46%), highlighting the effectiveness and robustness of LEAwareSGD.Table 6 displays the results on the OfficeHome dataset, emphasizing the optimizer's strong dependence on the learning rate. Notably, even the best performance achieved by SGD (53.20%) falls short of our method (54.38%), demonstrating the superior effectiveness and robustness of LEAwareSGD.

Optimizer	lr=0.001	lr=0.0001	lr=0.0005
Adam	18.03	66.43	36.88
AdamW	15.70	66.83	34.34
RMSprop	15.76	62.33	16.64
SGD	67.06	47.17	63.26
Ours	68.29	67.09	69.46

Table 5. Average accuracy (%) on PACS under Optimizer different learning rates.

Optimizer	lr=0.001	lr=0.0001	lr=0.0005
Adam	26.62	48.10	34.21
AdamW	27.57	47.81	34.69
RMSprop	17.43	46.50	26.55
SGD	52.60	53.20	53.08
Ours	53.31	54.38	53.83

Table 6. Average accuracy (%) on OfficeHome under Optimizer different learning rates.

C.6. Comparison with Backbone Approaches on OfficeHome

Table 7 shows that our approach achieves 58.56% on ResNet-34, improves to 60.91% on ResNet-50, further reaches 62.99% on ResNet-101, and attains its highest performance of 64.95% on ResNet-152. These results highlight the robustness, versatility, and superior accuracy of our method across different backbone architectures.

Backbone	Methods	A	C	P	R	Avg.
ResNet34	PSDG	56.63	56.04	54.13	60.58	56.85
	AdvST	50.22	52.73	51.43	59.68	53.52
	Ours	56.59	58.41	56.91	62.34	**58.56**
ResNet50	PSDG	59.94	58.71	56.71	61.76	59.28
	AdvST	57.69	59.16	56.09	63.36	59.08
	Ours	60.31	61.05	58.54	63.73	**60.91**
ResNet101	PSDG	59.32	59.65	58.98	63.68	60.41
	AdvST	59.47	62.09	58.70	64.41	61.17
	Ours	62.24	62.76	61.01	65.93	**62.99**
ResNet152	PSDG	64.08	62.89	61.27	65.67	63.48
	AdvST	63.00	62.30	60.40	67.29	63.25
	Ours	64.41	63.62	63.97	67.78	**64.95**

Table 7. Comparison results (%) using different ResNet series backbones on the OfficeHome dataset. The best results are in bold.

C.7. Comparison with limited training data on OfficeHome

Our method yields significant gains, as shown in Table 8. At the 10% data ratio, our approach achieves an average accuracy of 38.77%, a 4.00% increase over AdvST's 34.77%. With 20% data, the average accuracy rises to 45.55%, outperforming AdvST by 4.04%. At the 50% data ratio, our method achieves an average accuracy of 51.79%, a 3.55% improvement over AdvST's 48.24%.

Ratio	Methods	A	C	P	R	Avg.	ΔAvg.
10%	AdvST	27.93	32.72	36.88	41.56	34.77	
	Ours	32.00	37.13	42.03	43.92	38.77	+4.00
20%	AdvST	34.10	41.84	42.73	47.38	41.51	
	Ours	37.46	45.53	47.81	51.41	45.55	**+4.04**
50%	AdvST	43.33	47.63	47.92	54.06	48.24	
	Ours	47.88	51.64	51.37	56.26	51.79	+3.55

Table 8. Performance comparison (%) using different data ratios on OfficeHome dataset.

References

[1] Guangtao Zheng, Mengdi Huai, and Aidong Zhang. Advst: Revisiting data augmentations for single domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 21832–21840, 2024. 3

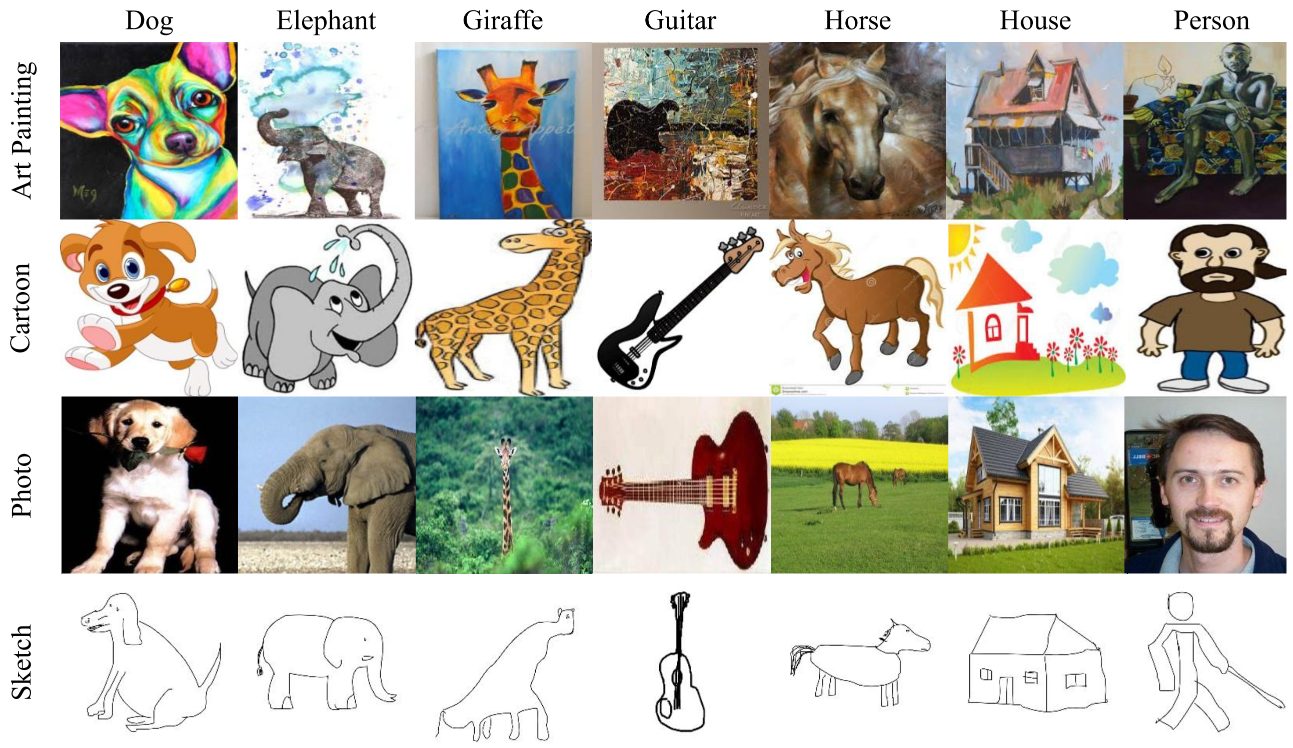


Figure 1. PACS overview.

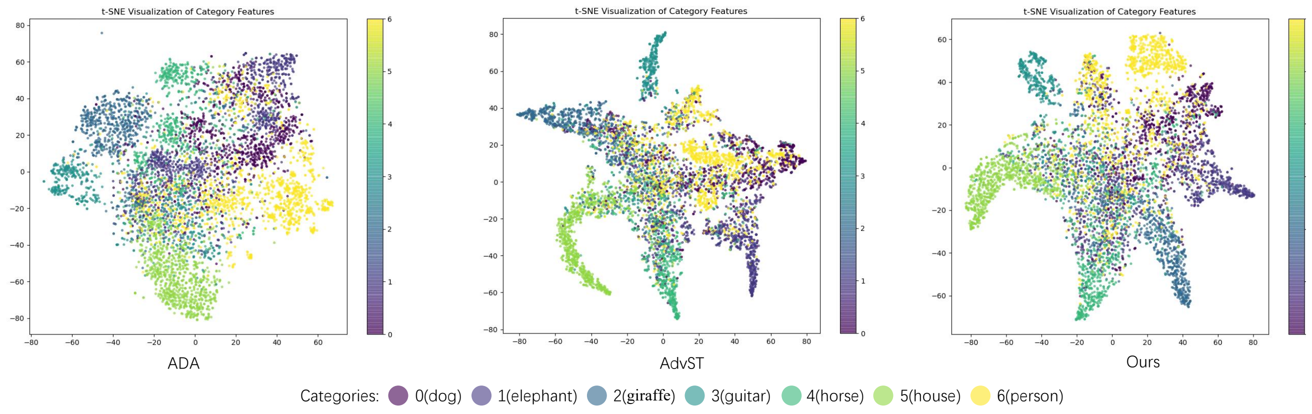


Figure 2. The t-SNE visualization of features from different categories on the PACS dataset. The model was trained on the Sketch domain and tested on the remaining domains. Different colors indicate different categories. It shows that our method achieves enhanced classification performance, as indicated by well-separated clusters corresponding to different categories.

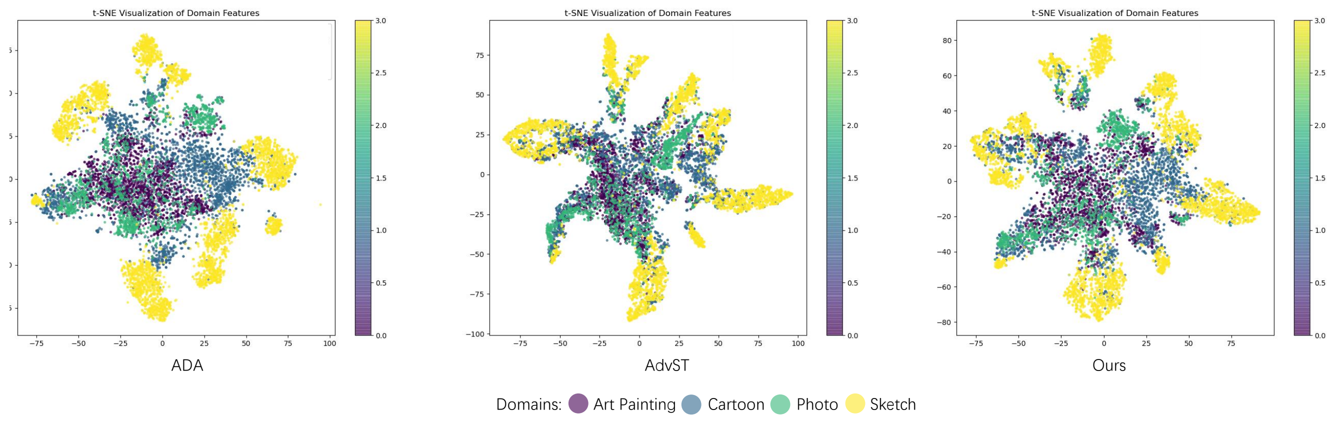


Figure 3. The t-SNE visualization of features from different domains on the PACS dataset. Different colors indicate different domains. It highlights the ability of our method to bridge the gap between domains.

Figure 4. OfficeHome overview.

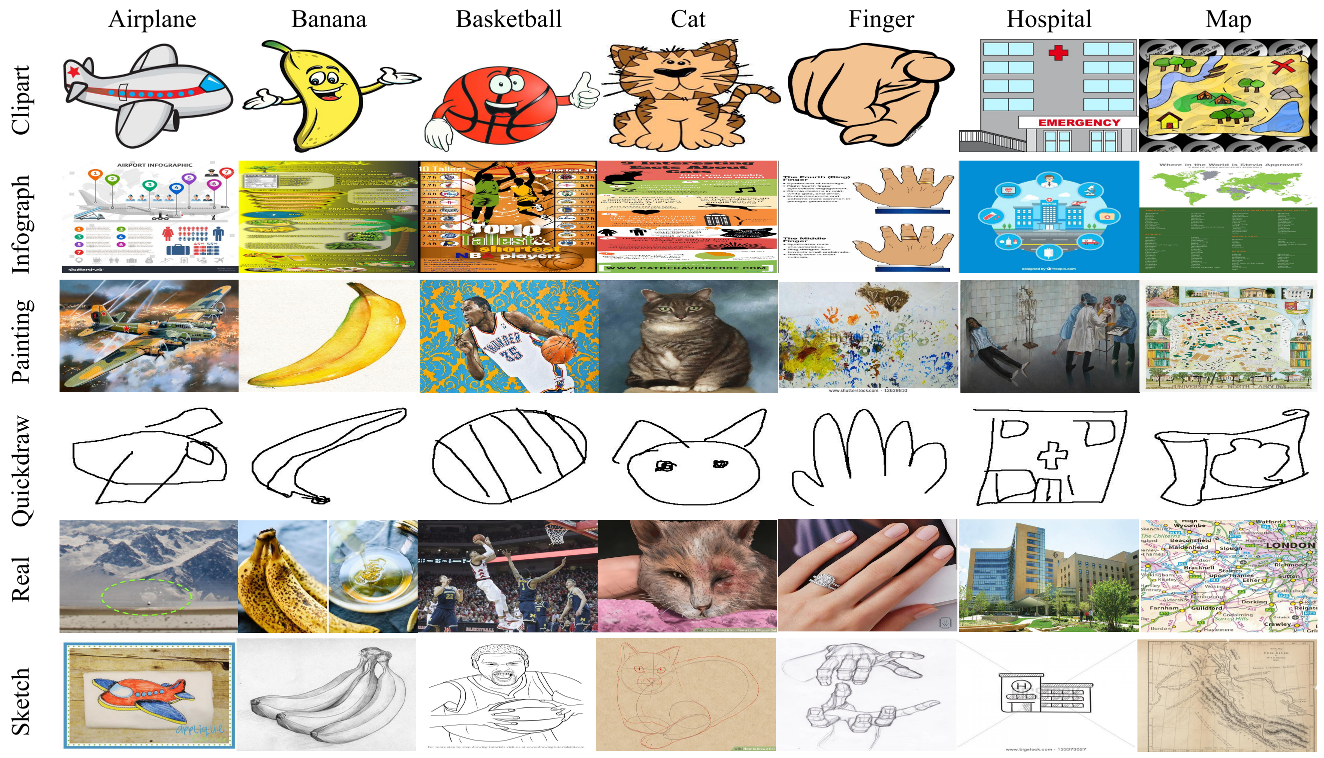


Figure 5. DomainNet overview.