Seeking Consistent Flat Minima for Better Domain Generalization via Refining Loss Landscapes

Supplementary Material

Let us start with a brief overview of this supplementary material. In Section 1, we perform a theoretical analysis of the loss landscape consistency under the PAC-Bayesian framework. The analysis shows that the sharpness of the test loss can be constrained by that of the training domains, which also implies the generalization of consistent flat minima sought by the self-feedback training (SFT). In Section 2, we discuss the proposed projection cross-entropy loss and the algorithm used to solve the associated KL divergence minimization problem. We provide a comprehensive derivation of the algorithm, along with a time comparison against an implementation using the common convex optimization library. Subsequently, we present the full results obtained using Resnet-50, ViT-B/16, and ViT-L/14 in the following two sections (Section 3 and 4). Additionally, we offer necessary explanations of the code (in Section 5.1) and hyperparameters (in Section 5.2) required for reproducibility. Finally, in Section 6, we assure that this research is unlikely to have any significant negative social impact.

1. Theoretical Analysis

In this section, we will perform a theoretical analysis of the loss landscape consistency under the PAC-Bayesian framework. For clarity and ease of understanding, we first provide a detailed explanation of the relevant notations and concepts that will be used throughout the analysis.

1.1. Notations

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input sample space and the category space, respectively. Consider a dataset drawn from p training distributions $\{\mathcal{D}_d\}_{d=1}^{p}$, each defined over the joint space $\mathcal{X}\times\mathcal{Y}$. Let $D_d = \{(\boldsymbol{x}_i^{(d)}, y_i^{(d)})\}_{i=1}^{n_d}$ denote the dataset sampled from the d-th distribution $\mathcal{D}_d$, which is referred to as the d-th domain. $(\boldsymbol{x}_i^{(d)}, y_i^{(d)}) \in \mathcal{X}\times\mathcal{Y}$ denotes the i-th sample from domain $\mathcal{D}_d$, with n_d indicating the number of samples in the d-th domain. For convenience, we also use z_i to denote $(\boldsymbol{x}_i, y_i)$. Let Ω and Ω' denote the dataset space and distribution space, respectively. In our analysis, domain shifts are modeled by a mapping function $\omega : \Omega \to \Omega'$, which maps one dataset to another distribution with distinct statistical properties. We assume that the domain shifts ω follow a specific distribution $\mathcal{W}$.

Our SFT framework mainly involves a model and a landscape refiner. Let $\mathcal{H}_m$ and $\mathcal{H}_r$ denote the hypothesis spaces of the model and the refiner, respectively. To analyze the SFT within the PAC-Bayesian framework, we need to provide a description using Bayesian terms. Let $\mathcal{M}_m$, $\mathcal{M}_r$ denote the sets of distributions over $\mathcal{H}_m$ and $\mathcal{H}_r$. In order to obtain a model from its prior distribution, we first sample a prior distribution P for the refiner from the hyper-prior distribution $\mathcal{P}$, which is independent of training samples. Then, we use a mapping function $\psi : \mathcal{M}_r \to \mathcal{M}_m$ to obtain the model's prior distribution $\psi(P) \in \mathcal{M}_m$ and sample a model f from $\psi(P)$. In order to obtain a model from its posterior distribution, we sample a posterior distribution P for the refiner from a hyper-posterior distribution $\mathcal{Q}$, which may depend on training samples. Then, we apply the training algorithm to the dataset $D_d \in \Omega$ to obtain the model's posterior distribution $\mathcal{A}(D_d, P)$, where $\mathcal{A} : \Omega \times \mathcal{M}_r \to \mathcal{M}_m$ represents the function that maps one dataset along with the refiner's posterior distribution to the model's posterior distribution. Finally, we can sample a model f from this posterior distribution.

1.2. Main Theorem

In this subsection, we first introduce a lemma that will be used multiple times in our analysis, and then formally present the main result, which is stated as Theorem 2.

Lemma 1 (McAllester's bound [1])**.** *Let $\mathcal{X}$ be a sample space and $\mathcal{H}$ a hypothesis space of functions over $\mathcal{X}$. Given π be some prior distribution over hypothesis space $\mathcal{H}$, for bounded loss $\ell : \mathcal{H}\times\mathcal{X} \to [0,1]$ and any $\delta \in (0,1]$, the following bound holds uniformly for all posterior distributions ρ with probability at least $1-\delta$:*

$$\mathop{\mathbb{E}}_{\theta\sim\rho} \ell(\theta, \mathcal{D}) \le \mathop{\mathbb{E}}_{\theta\sim\rho} \ell(\theta, S_n) + \sqrt{\frac{\mathrm{KL}(\rho||\pi) + \log(n/\delta)}{2(n-1)}}, \tag{1}$$

where

$$\ell(\theta, \mathcal{D}) = \mathbb{E}_{z\sim D}\ell(\theta, z)$$

and

$$\ell(\theta, S_n) = \frac{1}{n}\sum_{i=1}^{n} \ell(\theta, z_i)$$

denotes the population loss and training loss, respectively. S_n represents a dataset with n training samples drawn independently and identically from distribution $\mathcal{D}$.

Theorem 2. *Consider the domain generalization problem with p training domains. For each training domain $\mathcal{D}_d$, we are given $p-1$ training-domain pairs $(\mathcal{D}_d, \mathcal{D}_{d'})$, where $d' \neq d$, with each pair consisting of a training dataset D_d of size n_d and a hold-out dataset $D_{d'}$ of size $n_{d'}$. Let the*

dataset space be Ω and the distribution space be Ω'. Assume that the domain shifts, denoted by $\omega : \Omega \to \Omega'$, follow a distribution $\mathcal{W}$. The difference in loss sharpness between domains $\mathcal{D}_d$ and $\mathcal{D}_{d'}$ is defined as:

$$\Delta\ell(f, \mathcal{D}_d, \mathcal{D}_{d'}) := |\mathop{\mathbb{E}}_{z\sim\mathcal{D}_d} \ell(f, z) - \mathop{\mathbb{E}}_{z\sim\mathcal{D}_{d'}} \ell(f, z)|, \quad (2)$$

where $\ell(f, z)$ represent the loss sharpness of the model f evaluated on data point z. Let $\mathcal{P}$ denote a predefined hyper-prior distribution over the set of all possible prior distributions for the landscape refiner. Then, for all hyper-posterior distributions $\mathcal{Q}$ that ensure a sufficiently small sharpness difference between training domains, i.e.,

$$\forall d' \neq d : \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \Delta\ell(f, \mathcal{D}_d, \mathcal{D}_{d'}) \leq \epsilon, \quad (3)$$

and for any $\delta \in (0, 1]$, the following inequality holds with probability at least $1 - \delta$:

$$\ell(\mathcal{Q}, \omega) \leq \epsilon + \hat{\ell}(\mathcal{Q}, D_d) + \frac{1}{p-1} \times \sum_{d'\neq d} \sqrt{\frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathrm{KL}\left(\mathcal{A}\|\psi(P)\right) + \log\frac{2(p-1)n_d}{\delta}}{2(n_d-1)}} + \sqrt{\frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \log\frac{2(p-1)}{\delta}}{2(p-2)}}. \quad (4)$$

Here, $\psi(P)$ is the prior model distribution, and $\mathcal{A}$ is a shorthand of the posterior model distribution $\mathcal{A}(D_d, P)$. $\mathrm{KL}(\cdot\|\cdot)$ denotes the Kullback-Leibler divergence. The term $\ell(\mathcal{Q}, \omega)$ is defined as:

$$\ell(\mathcal{Q}, \omega) := \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{\omega\sim\mathcal{W}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \mathop{\mathbb{E}}_{z\sim\omega(D_d)} \ell(f, z), \quad (5)$$

which represents the expected sharpness of the model evaluated on the test distribution $\omega(D_d)$. The term $\hat{\ell}(\mathcal{Q}, D_d)$ is defined as:

$$\hat{\ell}(\mathcal{Q}, D_d) := \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \frac{1}{n_d} \sum_{i=1}^{n_d} \ell\left(f, z_i\right), \quad (6)$$

which represents the empirical sharpness of the model over the training domain D_d.

Proof. Firstly, we bound the loss sharpness in each of the domain pairs. Based on the above definitions in the subsection 1.1, we can decompose the KL divergence term of Lemma 1 in the following way:

$$\begin{aligned}
&\mathrm{KL}(\rho\|\pi) \\
&= \mathop{\mathbb{E}}_{f\sim\rho} \log\frac{\rho(f)}{\pi(f)} = \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \log\frac{\mathcal{Q}(P)\mathcal{A}\left(D_d, P\right)(f)}{\mathcal{P}(P)\psi(P)(f)} \\
&= \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \log\frac{\mathcal{Q}(P)}{\mathcal{P}(P)} + \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \log\frac{\mathcal{A}\left(D_d, P\right)(f)}{\psi(P)(f)} \\
&= \mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathrm{KL}\left(\mathcal{A}\left(D_d, P\right)\|\psi(P)\right). \quad (7)
\end{aligned}$$

This decomposition separates the KL divergence into two components: $\mathrm{KL}(\mathcal{Q}\|\mathcal{P})$ and $\mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathrm{KL}\left(\mathcal{A}\left(D_d, P\right)\|\psi(P)\right)$. Now, based on this, we can establish a probabilistic upper bound on the loss sharpness. Thus, with probability at least $1 - \delta_{d'}$, we have:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \mathop{\mathbb{E}}_{z\sim\mathcal{D}_{d'}} \ell(f, z) \\
&\leq \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \frac{1}{n_d} \sum_{i=1}^{n_d} \ell\left(h, z_i\right) \\
&\quad + \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \Delta\ell(h, \mathcal{D}_d, \mathcal{D}_{d'}) \\
&\quad + \sqrt{\frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathrm{KL}\left(\mathcal{A}\|\psi(P)\right) + \log\frac{n_d}{\delta_{d'}}}{2\left(n_d-1\right)}}. \quad (8)
\end{aligned}$$

This bound captures the generalization ability between training domains, i.e., from domain $\mathcal{D}_d$ to domain $\mathcal{D}_{d'}$.

Next, assuming that the domain shifts $\omega : \Omega \to \Omega'$ are governed by a distribution $\mathcal{W}$, we can apply Lemma 1 again to bound the loss sharpness on the test domain $\omega(D_d)$. This step is critical because it extends the generalization to the test domains. Specifically, with probability at least $1 - \delta'$, we obtain the following bound:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{\omega\sim\mathcal{W}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \mathop{\mathbb{E}}_{z\sim\omega(D_d)} \ell(f, z) \\
&\leq \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \frac{1}{p-1} \sum_{d'\neq d} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \mathop{\mathbb{E}}_{z\sim\mathcal{D}_{d'}} \ell(f, z) \\
&\quad + \sqrt{\frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \log\frac{p-1}{\delta'}}{2(p-2)}}. \quad (9)
\end{aligned}$$

Here, the first term represents an average over all domain pairs, while the second term corresponds to the KL divergence between hyper-prior and hyper-posterior distributions. The confidence parameter δ' controls the probability of the bound holding.

Finally, we set $\delta' = \delta/2$ and $\delta_{d'} = \delta/[2(p-1)]$ and apply the union bound to combine the above results. This leads to the final bound, which holds with probability at least $1 - \delta$,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{\omega\sim\mathcal{W}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \mathop{\mathbb{E}}_{z\sim\omega(D_d)} \ell(h, z) \\
&\leq \frac{1}{p-1} \sum_{d'\neq d} \left[\mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \frac{1}{n_d} \sum_{i=1}^{n_d} \ell\left(f, z_i\right) \right. \\
&\quad + \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathop{\mathbb{E}}_{f\sim\mathcal{A}(D_d,P)} \Delta\ell(f, \mathcal{D}_d, \mathcal{D}_{d'}) + \\
&\quad \left. \sqrt{\frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathrm{KL}\left(\mathcal{A}\|\psi(P)\right) + \log\frac{2(p-1)n_d}{\delta}}{2(n_d-1)}} \right] \\
&\quad + \sqrt{\frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \log\frac{2(p-1)}{\delta}}{2(p-2)}}. \quad (10)
\end{aligned}$$

The inequality above provides a comprehensive bound of the loss sharpness on the test domains, completing the proof of Theorem 2. □

In conclusion, this theorem shows that if domain shifts are governed by a specific distribution and the domain shifts in the training domains are independently and identically sampled from this distribution, then the sharpness of the test loss is bounded by the sharpness observed in the training domains with high probability. In other words, the consistency of the flat minima achieved in the training domains can be transferred to unseen test domains. As a result, the model can exhibit strong generalization performance when applied to test domains.

2. Projection Cross Entropy

As mentioned in the main text, the projection cross entropy (PCE) can be used as a loss term to maintain the label correctness during the refinement phase. An efficient algorithm (Alogrithm 1) has been presented to address the associated KL divergence minimization problem there.

In this section, we first provide the derivation of this algorithm, followed by a comparison of its time efficiency with that of widely-used convex optimization libraries. Finally, we further discuss the connections between the PCE loss and other related loss functions.

2.1. KL Divergence Minimization

As stated in the main text, the optimization is formulated as:

$$\min_{\mathbf{y}} \mathrm{KL}(\mathbf{y} \,||\, \tilde{\mathbf{y}}) \quad \text{subject to} \quad \mathbf{y} \in C_1, \tag{11}$$

where $\tilde{\mathbf{y}}$ represents the soft label output by the landscape refiner, and the label space C_1 can be expressed as:

$$C_1 = \{(q_1, \ldots, q_N) \mid \forall k \neq 1 : q_1 \geq \alpha q_k, \sum_{k=1}^{N} q_k = 1\}. \tag{12}$$

Here, $\alpha \geq 1$ is a hyperparameter that controls the minimum ratio between q_1 and q_k, while N denotes the number of categories for classification. For ease of understanding, we restate the problem more explicitly as follows:

$$\begin{aligned} &\min_{q_i} \sum_{i=1}^{N} q_i \log \frac{q_i}{p_i} \\ \text{s. t.} \quad &\sum_{i=1}^{N} q_i = 1, \quad q_1 \geq \alpha q_k \ (k \neq 1), \end{aligned} \tag{13}$$

where we use $(p_1, \ldots, p_N)$ to represent $\tilde{\mathbf{y}}$ for clarity.

In the following, we will offer a detailed derivation of Algorithm 1, which is capable of finding the exact optimal solution to this optimization problem. The general idea of the derivation is as follows: first, by examining the Lagrangian function and the Karush-Kuhn-Tucker (KKT) conditions of the problem, we obtain the general form of the optimal solution. Then, we determine which constraints among the inequality constraints hold as a strict equality (i.e., active), and finally, we find the optimal solution based on these active constraints.

Algorithm 1: KL Divergence Minimization

Input: The hyperparameter α, $(p_1, \ldots, p_N)$.
Output: The optimal solution $(q_1, \ldots, q_N)$.

Initialization: $A \leftarrow \emptyset$, $B \leftarrow \{j | \alpha p_j > p_1\}$, $t \leftarrow 1$.
Sort the elements of B in descending order: $p_{j_1} \geq \cdots \geq p_{j_{|B|}}$.
Update: $A \leftarrow A \cup \{j_1\}$, $t \leftarrow t + 1$.
while $t \leq |B|$ **do**
  if $\left(p_1^{\alpha} \left(\prod_{j \in A} \alpha p_j\right)\right)^{\frac{1}{|A|+\alpha}} < \alpha p_{j_t}$ **then**
    Update: $A \leftarrow A \cup \{j_t\}$, $t \leftarrow t + 1$.
  else
    Break
  end
end
Calculate q_1: $q_1 = \left(p_1^{\alpha} \left(\prod_{j \in A} \alpha p_j\right)\right)^{\frac{1}{|A|+\alpha}}$.
for $i \in \{2, \ldots, N\}$ **do**
  if $i \in A$ **then**
    $q_i = q_1 / \alpha$.
  else
    $q_i = p_i$.
  end
end
Normalize $q_1, \ldots, q_N$ such that $\sum_i q_i = 1$.

2.1.1. Lagrangian Function and KKT Conditions

To solve the given optimization problem, we first construct the Lagrangian function by incorporating the objective function and the constraints using Lagrange multipliers:

$$\begin{aligned} L(q_i, \mu_k, \lambda) = &\sum_{i=1}^{N} q_i \log \frac{q_i}{p_i} + \sum_{k=2}^{N} \mu_k (\alpha q_k - q_1) \\ &+ \lambda (\sum_{i=1}^{N} q_i - 1), \end{aligned} \tag{14}$$

where $\mu_k \geq 0$ represents the Lagrange multiplier for the inequality constraint $q_1 \geq \alpha q_k$, and λ denotes the Lagrange multiplier for the equality constraint $\sum_{i=1}^{N} q_i = 1$. Then, the Karush-Kuhn-Tucker (KKT) conditions are the necessary conditions for optimality in constrained optimization problems. We apply these conditions to the Lagrangian function and derive the following set of equations:

1. Stationarity. The stationarity conditions require that the partial derivatives of the Lagrangian with respect to each of the variables be zero, which corresponds to the opti-

mality condition. For q_1, we have the following equation:

$$\frac{\partial L}{\partial q_1} = 1 + \log q_1 - \log p_1 - \sum_{k=2}^{N} \mu_k + \lambda = 0. \quad (15)$$

Similarly, for q_j, where $j = 2, \ldots, n$, we get:

$$\frac{\partial L}{\partial q_j} = 1 + \log q_j - \log p_j + \mu_j \alpha + \lambda = 0. \quad (16)$$

These two equations can be solved to express q_1 and q_j in terms of the Lagrange multipliers μ_k and λ. Thus, we have the following solutions:

$$q_1 = p_1 \exp\left(\sum_{k=2}^{N} \mu_k - 1 - \lambda\right) \quad (17)$$

and

$$q_j = p_j \exp\left(-\mu_j \alpha - 1 - \lambda\right). \quad (18)$$

2. Primal feasibility. The primal feasibility condition ensures that the original constraints are satisfied. Therefore, we have:

$$\alpha q_j - q_1 \leq 0. \quad (19)$$

3. Dual feasibility. The dual feasibility condition imposes non-negativity on the Lagrange multipliers associated with the inequality constraints:

$$\mu_j \geq 0. \quad (20)$$

4. Complementary slackness. Finally, the complementary slackness condition relates the primal and dual variables. In this case, the complementary slackness condition for the inequality constraint is:

$$\mu_j(\alpha q_j - q_1) = 0. \quad (21)$$

This condition implies that either $\mu_j = 0$ or $\alpha q_j = q_1$ (or both). It ensures that if the constraint is inactive (i.e., it holds as a strict inequality), the corresponding multiplier is zero, and if the multiplier is positive, the corresponding constraint is active (i.e., it holds as a strict equality).

In the following, we can obtain the general form of the optimal solution by examining the two cases of $\mu > 0$ and $\mu = 0$ as per the KKT conditions.

- **Case 1:** $\mu_j > 0$.

If $\mu_j > 0$, from the stationarity condition and complementary slackness, we can derive the following equation:

$$\begin{aligned} \alpha q_j &= \alpha p_j \exp\left(-\mu_j \alpha - 1 - \lambda\right) \\ =& q_1 = p_1 \exp\left(\sum_{k=2}^{N} \mu_k - 1 - \lambda\right). \end{aligned} \quad (22)$$

Through a simple manipulation of the equation, we can obtain:

$$\frac{p_1}{\alpha p_j} = \frac{\exp(-\mu_j \alpha)}{\exp(\sum_{k=2}^{N} \mu_k)}. \quad (23)$$

Since $\mu_j > 0$ in this case, the right side of the above equation is less than 1. In other words, if $p_1 \geq \alpha p_j$, then we can determine that $\mu_j = 0$. On the other hand, by setting the value of j in equation (22) and multiplying these expressions together, we can obtain the following relationship:

$$\alpha^{|A|} \exp(-\alpha \sum_{j \in A} \mu_j) \prod_{j \in A} p_j = p_1^{|A|} \exp(|A| \sum_{k \in A} \mu_k). \quad (24)$$

Here, A is defined as the set of indices k such that $\mu_k > 0$, i.e., $A := \{k | \mu_k > 0\}$. $|A|$ denotes the cardinality of set A. Then, we can derive:

$$\exp(\sum_{k \in A} \mu_k) = (p_1^{-|A|} (\prod_{j \in A} \alpha p_j))^{\frac{1}{|A|+\alpha}}. \quad (25)$$

By using the above equation, we can express q_1 and q_j without using μ_j:

$$q_1 = \alpha q_j = \exp\left(-1 - \lambda\right) (p_1^{\alpha} (\prod_{j \in A} \alpha p_j))^{\frac{1}{|A|+\alpha}}. \quad (26)$$

- **Case 2:** $\mu_j = 0$.
If $\mu_j = 0$, q_j can be easily expressed without using μ_j:

$$q_j = p_j \exp\left(-1 - \lambda\right). \quad (27)$$

Then, by using the equations (22), (26) and (27), we can also express the objective function without using μ_j:

$$\sum_{i=1}^{N} q_i \log \frac{q_i}{p_i} = -(1 + \lambda). \quad (28)$$

Finally, by applying the normalization condition $\sum_{i=1}^{N} q_i = 1$, we can solve for the value of λ using the following equation:

$$\exp(1+\lambda) = (1 + \frac{|A|}{\alpha})(p_1^{\alpha} (\prod_{j \in A} \alpha p_j))^{\frac{1}{|A|+\alpha}} + \sum_{j \notin A \cup \{1\}} p_j. \quad (29)$$

2.1.2. Determine Active Constraints

To determine the active constraints, we have the following proposition, which directly leads to the formulation of Algorithm 1.

Proposition 3. *Consider the optimization problem in (13). Let A be the set of indices k such that $\mu_k > 0$, i.e., $A := \{k | \mu_k > 0\}$. Define B as the set $\{p_j | \alpha p_j > p_1\}$. Sort the elements of B in descending order as:*

$$p_{j_1} \geq p_{j_2} \geq \ldots \geq p_{j_{|B|}}. \quad (30)$$

Note that duplicate elements in B are not removed. Then, the following conclusions holds:

1. *The index j_1, which corresponds to the largest element in B, must belong to A, i.e., $j_1 \in A$.*
2. *For any $C = \{j_1, \ldots, j_{t-1}\} \subseteq A$, if the inequality*

$$(p_1^\alpha(\prod_{j\in C} \alpha p_j))^{\frac{1}{|C|+\alpha}} < \alpha p_{j_t} \tag{31}$$

holds, then $j_t \in A$. Otherwise, for all $s \in \{t, t+1, \ldots, |B|\}$, we have $j_s \notin A$. That is, $A = C$.

Proof. Firstly, we can prove that $j_1 \in A$. To do so, assume for the sake of contradiction that $j_1 \notin A$. By using equations (26) and (27), we obtain the following expression for q_1:

$$\begin{aligned} q_1 &= \exp(-1-\lambda)(p_1^\alpha(\prod_{j\in A} \alpha p_j))^{\frac{1}{|A|+\alpha}} \\ &< \exp(-1-\lambda)((\alpha p_{j_1})^\alpha(\prod_{j\in A} \alpha p_{j_1}))^{\frac{1}{|A|+\alpha}} \\ &= \alpha \exp(-1-\lambda) p_{j_1} = \alpha q_{j_1}. \end{aligned} \tag{32}$$

Here, the first equality follows from the equation (26), and the inequality in the second-to-last line is based on the definition of B, in which $p_1 < \alpha p_{j_1}$. The first equality in the last line is simply an identity transformation, and the last equation uses the equation (27) under the assumption that $j_1 \notin A$. The above analysis shows that $q_1 < \alpha q_{j_1}$, which clearly violates the inequality constraints of the original problem. Thus, our assumption that $j_1 \notin A$ must be false.

Secondly, we consider the case where $C = \{j_1, \ldots, j_{t-1}\} \subseteq A$, and we assume that the inequality

$$(p_1^\alpha(\prod_{j\in C} \alpha p_j))^{\frac{1}{|C|+\alpha}} < \alpha p_{j_t} \tag{33}$$

holds. We now want to prove that $j_t \in A$. Suppose, for contradiction, that $j_t \notin A$. In this case, we proceed as follows. Using the equations (26) and (27), we can express q_1 as:

$$\begin{aligned} q_1 &= \exp(-1-\lambda)(p_1^\alpha(\prod_{j\in A} \alpha p_j))^{\frac{1}{|A|+\alpha}} \\ &= \exp(-1-\lambda)(p_1^\alpha(\prod_{j\in C} \alpha p_j)(\prod_{j\in A-C} \alpha p_j))^{\frac{1}{|A|+\alpha}} \\ &< \exp(-1-\lambda)(p_1^\alpha(\prod_{j\in C} \alpha p_j)(\prod_{j\in A-C} \alpha p_{j_t}))^{\frac{1}{|A|+\alpha}} \\ &< \exp(-1-\lambda)((\alpha p_{j_t})^{|C|+\alpha}(\prod_{j\in A-C} \alpha p_{j_t}))^{\frac{1}{|A|+\alpha}} \\ &= \alpha \exp(-1-\lambda) p_{j_t} = \alpha q_{j_t}. \end{aligned} \tag{34}$$

The first equality uses equation (26). The first inequality follows from the fact that $p_{j_t} > p_j$ for any $j \in A-C$, which is because the elements of B have been sorted in descending order. The second inequality uses the condition stated in (33). The last equality follows from equation (27) under the assumption that $j_t \notin A$. The above analysis shows that $q_1 < \alpha q_{j_t}$, which, again, violates the inequality constraints of the original problem. Hence, our assumption that $j_t \notin A$ is false, and it holds that $j_t \in A$.

Finally, we address the case where the inequality (33) does not holds. In this case, for all $s \in \{t, t+1, \ldots, |B|\}$, we assert that $j_s \notin A$. We will now prove this by considering two cases. In the first case, we assume that there exists some $j_s \in A$ such that $p_1^\alpha(\prod_{j\in C} \alpha p_j))^{\frac{1}{|C|+\alpha}} > \alpha p_{j_s}$. If $j_s \in A$, we then have:

$$\begin{aligned} &\sum_{i=1}^N q_i \log \frac{q_i}{p_i} = -(1+\lambda) \\ &= -\log\left[(1+\frac{|C|+1}{\alpha})((p_1^\alpha(\prod_{j\in C\cup\{j_s\}} \alpha p_j))^{\frac{1}{|C|+1+\alpha}} \right. \\ &\quad \left. + \sum_{j\notin C\cup\{1,j_s\}} p_j\right]. \end{aligned} \tag{35}$$

The equations (28) and (29) have been applied above. However, if $j_s \notin A$, we have

$$\begin{aligned} &\sum_{i=1}^N q_i \log \frac{q_i}{p_i} = -(1+\lambda) \\ &= -\log\left[(1+\frac{|C|}{\alpha})(p_1^\alpha(\prod_{j\in C} \alpha p_j))^{\frac{1}{|C|+\alpha}} + \sum_{j\notin C\cup\{1\}} p_j\right]. \end{aligned} \tag{36}$$

Now we consider the function of p_{j_s}:

$$\begin{aligned} f(p_{j_s}) &= (1+\frac{|C|}{\alpha})(p_1^\alpha(\prod_{j\in C} \alpha p_j))^{\frac{1}{|C|+\alpha}} + p_{j_s} \\ &\quad - (1+\frac{|C|+1}{\alpha})(p_1^\alpha(\alpha p_{j_s})(\prod_{j\in C} \alpha p_j))^{\frac{1}{|C|+1+\alpha}}. \end{aligned} \tag{37}$$

Then, the first derivative of $f(p_{j_s})$ with respect to p_{j_s} is given by:

$$f'(p_{j_s}) = 1 - \frac{1}{\alpha}(\alpha p_1^\alpha(\prod_{j\in C} \alpha p_j))^{\frac{1}{|C|+1+\alpha}} p_{j_s}^{\frac{1}{|C|+1+\alpha}-1}. \tag{38}$$

Obviously, $f'(p_{j_s})$ increases gradually as p_{j_s} increases, and $f'(\frac{1}{\alpha}(p_1^\alpha(\prod_{j\in C} \alpha p_j))^{\frac{1}{|C|+\alpha}}) = 0$. Since

$(p_1^\alpha(\prod_{j\in C}\alpha p_j))^{\frac{1}{|C|+\alpha}} > \alpha p_{j_s}$, we can get

$$f(p_{j_s}) > f(\frac{1}{\alpha}(p_1^\alpha(\prod_{j\in C}\alpha p_j))^{\frac{1}{|C|+\alpha}}) = 0. \quad (39)$$

From equation (39), we can observe that the value of the objective function is smaller if $j_s \notin A$. In the second case, if there exists j_s such that $(p_1^\alpha(\prod_{j\in C}\alpha p_j))^{\frac{1}{|C|+\alpha}} = \alpha p_{j_s}$ and $j_s \in A$, we have

$$\begin{aligned} q_{j_s} &= \frac{q_1}{\alpha} \\ &= \frac{1}{\alpha}\exp(-1-\lambda)(p_1^\alpha(\alpha p_{j_s})(\prod_{j\in C}\alpha p_j))^{\frac{1}{|C|+1+\alpha}} \\ &= p_{j_s}\exp(-1-\lambda), \end{aligned} \quad (40)$$

which leads to the conclusion that $\mu_{j_s} = 0$ according to the equation (18), contradicting the definition of the set A. Hence, no such j_s can exist. In conclusion, for all $s \in \{t, t+1, \ldots, |B|\}$, it holds that $j_s \notin A$.

□

At this time, we have successfully identified the active set A. Subsequently, we can obtain the optimal solution by applying equations (26), (27), and (29), which leads to the formulation of Algorithm 1.

2.2. Time Efficiency

Although the proof may be somewhat intricate, it is worth emphasizing that Algorithm 1 exhibits significantly higher efficiency in comparison to conventional convex programming tools such as MOSEK. The table below presents the average time taken by the classical convex programming tool (MOSEK) and the proposed Algorithm 1 over 1000 runs. It is evident that for $N = 100$, Algorithm 1 consumes only around one-tenth of the time required by traditional convex programming tools.

Table 1. Comparisons about average training time required by Algorithm 1 and the MOSEK tool.

	MOSEK	Algo. 1 (Ours)	Speedup ratio
Time (Avg.)	21.93 ms	2.16 ms	10.15

2.3. Further Discussion

By examining Algorithm 1, we can find that if α is sufficiently large such that $|A| = N - 1$, the optimal soft labels degenerate into smoothed labels with a label smoothing factor of $s = N/(\alpha - 1 + N)$. Furthermore, as α increases towards infinity, the optimal soft labels converge towards one-hot labels. Then, the PCE loss degenerate into the traditional cross entropy.

3. Full Results with ResNet-50

In this section, we demonstrate the full results of our experiments that were carried out on five well-known benchmark datasets, including VLCS, PACS, OfficeHome, TerraIncognita and DomainNet. These experiments were conducted using ResNet-50, which was pre-trained on ImageNet.

3.1. VLCS

Table 2. Out-of-domain accuracies (%) on each domain of VLCS and their average.

Algorithms	C	L	S	V	Avg.
ERM	97.7 ± 0.4	64.3 ± 0.9	73.4 ± 0.5	74.6 ± 1.3	77.5
IRM	98.6 ± 0.1	64.9 ± 0.9	73.4 ± 0.6	77.3 ± 0.9	78.5
GroupDRO	97.3 ± 0.3	63.4 ± 0.9	69.5 ± 0.8	76.7 ± 0.7	76.7
Mixup	98.3 ± 0.6	64.8 ± 1.0	72.1 ± 0.5	74.3 ± 0.8	77.4
MLDG	97.4 ± 0.2	65.2 ± 0.7	71.0 ± 1.4	75.3 ± 1.0	77.2
CORAL	98.3 ± 0.1	66.1 ± 1.2	73.4 ± 0.3	77.5 ± 1.2	78.8
MMD	97.7 ± 0.1	64.0 ± 1.1	72.8 ± 0.2	75.3 ± 3.3	77.5
DANN	99.0 ± 0.3	65.1 ± 1.4	73.1 ± 0.3	77.2 ± 0.6	78.6
CDANN	97.1 ± 0.3	65.1 ± 1.2	70.7 ± 0.8	77.1 ± 1.5	77.5
MTL	97.8 ± 0.4	64.3 ± 0.3	71.5 ± 0.7	75.3 ± 1.7	77.2
SagNet	97.9 ± 0.4	64.5 ± 0.5	71.4 ± 1.3	77.5 ± 0.5	77.8
ARM	98.7 ± 0.2	63.6 ± 0.7	71.3 ± 1.2	76.7 ± 0.6	77.6
VREx	98.4 ± 0.3	64.4 ± 1.4	74.1 ± 0.4	76.2 ± 1.3	78.3
RSC	97.9 ± 0.1	62.5 ± 0.7	72.3 ± 1.2	75.6 ± 0.8	77.1
SFT (Ours)	99.5 ± 0.1	66.2 ± 0.2	74.8 ± 0.5	78.7 ± 0.4	79.8

3.2. PACS

Table 3. Out-of-domain accuracies (%) on each domain of PACS and their average.

Algorithms	A	C	P	S	Avg.
ERM	84.7 ± 0.4	80.8 ± 0.6	97.2 ± 0.3	79.3 ± 1.0	85.5
IRM	84.8 ± 1.3	76.4 ± 1.1	96.7 ± 0.6	76.1 ± 1.0	83.5
GroupDRO	83.5 ± 0.9	79.1 ± 0.6	96.7 ± 0.3	78.3 ± 2.0	84.4
Mixup	86.1 ± 0.5	78.9 ± 0.8	97.6 ± 0.1	75.8 ± 1.8	84.6
MLDG	85.5 ± 1.4	80.1 ± 1.7	97.4 ± 0.3	76.6 ± 1.1	84.9
CORAL	88.3 ± 0.2	80.0 ± 0.5	97.5 ± 0.3	78.8 ± 1.3	86.2
MMD	86.1 ± 1.4	79.4 ± 0.9	96.6 ± 0.2	76.5 ± 0.5	84.6
DANN	86.4 ± 0.8	77.4 ± 0.8	97.3 ± 0.4	73.5 ± 2.3	83.6
CDANN	84.6 ± 1.8	75.5 ± 0.9	96.8 ± 0.3	73.5 ± 0.6	82.6
MTL	87.5 ± 0.8	77.1 ± 0.5	96.4 ± 0.8	77.3 ± 1.8	84.6
SagNet	87.4 ± 1.0	80.7 ± 0.6	97.1 ± 0.1	80.0 ± 0.4	86.3
ARM	86.8 ± 0.6	76.8 ± 0.5	97.4 ± 0.3	79.3 ± 1.2	85.1
VREx	86.0 ± 1.6	79.1 ± 0.6	96.9 ± 0.5	77.7 ± 1.7	84.9
RSC	85.4 ± 0.8	79.7 ± 1.8	97.6 ± 0.3	78.2 ± 1.2	85.2
SFT (Ours)	90.1 ± 0.3	80.3 ± 1.0	98.6 ± 0.2	84.3 ± 1.4	88.3

3.3. OfficeHome

Table 4. Out-of-domain accuracies (%) on each domain of OfficeHome and their average.

Algorithms	A	C	P	R	Avg.
ERM	61.3 ± 0.7	52.4 ± 0.3	75.8 ± 0.1	76.6 ± 0.3	66.5
IRM	58.9 ± 2.3	52.2 ± 1.6	72.1 ± 2.9	74.0 ± 2.5	64.3
GroupDRO	60.4 ± 0.7	52.7 ± 1.0	75.0 ± 0.7	76.0 ± 0.7	66.0
Mixup	62.4 ± 0.8	54.8 ± 0.6	76.9 ± 0.3	78.3 ± 0.2	68.1
MLDG	61.5 ± 0.9	53.2 ± 0.6	75.0 ± 1.2	77.5 ± 0.4	66.8
CORAL	65.3 ± 0.4	54.4 ± 0.5	76.5 ± 0.1	78.4 ± 0.5	68.7
MMD	60.4 ± 0.2	53.3 ± 0.3	74.3 ± 0.1	77.4 ± 0.6	66.3
DANN	59.9 ± 1.3	53.0 ± 0.3	73.6 ± 0.7	76.9 ± 0.5	65.9
CDANN	61.5 ± 1.4	50.4 ± 2.4	74.4 ± 0.9	76.6 ± 0.8	65.8
MTL	61.5 ± 0.7	52.4 ± 0.6	74.9 ± 0.4	76.8 ± 0.4	66.4
SagNet	63.4 ± 0.2	54.8 ± 0.4	75.8 ± 0.4	78.3 ± 0.3	68.1
ARM	58.9 ± 0.8	51.0 ± 0.5	74.1 ± 0.1	75.2 ± 0.3	64.8
VREx	60.7 ± 0.9	53.0 ± 0.9	75.3 ± 0.1	76.6 ± 0.5	66.4
RSC	60.7 ± 1.4	51.4 ± 0.3	74.8 ± 1.1	75.1 ± 1.3	65.5
SFT (Ours)	65.8 ± 0.3	58.8 ± 0.3	78.3 ± 0.4	80.6 ± 0.2	70.9

3.4. TerraIncognita

Table 5. Out-of-domain accuracies (%) on each domain of TerraIncognita and their average.

Algorithms	L100	L38	L43	L46	Avg.
ERM	49.8 ± 4.4	42.1 ± 1.4	56.9 ± 1.8	35.7 ± 3.9	46.1
IRM	54.6 ± 1.3	39.8 ± 1.9	56.2 ± 1.8	39.6 ± 0.8	47.6
GroupDRO	41.2 ± 0.7	38.6 ± 2.1	56.7 ± 0.9	36.4 ± 2.1	43.2
Mixup	59.6 ± 2.0	42.2 ± 1.4	55.9 ± 0.8	33.9 ± 1.4	47.9
MLDG	54.2 ± 3.0	44.3 ± 1.1	55.6 ± 0.3	36.9 ± 2.2	47.7
CORAL	51.6 ± 2.4	42.2 ± 1.0	57.0 ± 1.0	39.8 ± 2.9	47.6
MMD	41.9 ± 3.0	34.8 ± 1.0	57.0 ± 1.9	35.2 ± 1.8	42.2
DANN	51.1 ± 3.5	40.6 ± 0.6	57.4 ± 0.5	37.7 ± 1.8	46.7
CDANN	47.0 ± 1.9	41.3 ± 4.8	54.9 ± 1.7	39.8 ± 2.3	45.8
MTL	49.3 ± 1.2	39.6 ± 6.3	55.6 ± 1.1	37.8 ± 0.8	45.6
SagNet	53.0 ± 2.9	43.0 ± 2.5	57.9 ± 0.6	40.4 ± 1.3	48.6
ARM	49.3 ± 0.7	38.3 ± 2.4	55.8 ± 0.8	38.7 ± 1.3	45.5
VREx	48.2 ± 4.3	41.7 ± 1.3	56.8 ± 0.8	38.7 ± 3.1	46.4
RSC	50.2 ± 2.2	39.2 ± 1.4	56.3 ± 1.4	40.8 ± 0.6	46.6
SFT (Ours)	57.5 ± 0.4	44.6 ± 1.4	59.6 ± 0.5	41.0 ± 1.0	50.7

3.5. DomainNet

Table 6. Out-of-domain accuracies (%) on each domain of TerraIncognita and their average.

Algorithm	clip	info	paint	quick	real	sketch	Avg.
ERM	58.1 ± 0.3	18.8 ± 0.3	46.7 ± 0.3	12.2 ± 0.4	59.6 ± 0.1	49.8 ± 0.4	40.9
IRM	48.5 ± 2.8	15.0 ± 1.5	38.3 ± 4.3	10.9 ± 0.5	48.2 ± 5.2	42.3 ± 3.1	33.9
GroupDRO	47.2 ± 0.5	17.5 ± 0.4	33.8 ± 0.5	9.3 ± 0.3	51.6 ± 0.4	40.1 ± 0.6	33.3
Mixup	55.7 ± 0.3	18.5 ± 0.5	44.3 ± 0.5	12.5 ± 0.4	55.8 ± 0.3	48.2 ± 0.5	39.2
MLDG	59.1 ± 0.2	19.1 ± 0.3	45.8 ± 0.7	13.4 ± 0.3	59.6 ± 0.2	50.2 ± 0.4	41.2
CORAL	59.2 ± 0.1	19.7 ± 0.2	46.6 ± 0.3	13.4 ± 0.4	59.8 ± 0.2	50.1 ± 0.6	41.5
MMD	32.1 ± 13.3	11.0 ± 4.6	26.8 ± 11.3	8.7 ± 2.1	32.7 ± 13.8	28.9 ± 11.9	23.4
DANN	53.1 ± 0.2	18.3 ± 0.1	44.2 ± 0.7	11.8 ± 0.1	55.5 ± 0.4	46.8 ± 0.6	38.3
CDANN	54.6 ± 0.4	17.3 ± 0.1	43.7 ± 0.9	12.1 ± 0.7	56.2 ± 0.4	45.9 ± 0.5	38.3
MTL	57.9 ± 0.5	18.5 ± 0.4	46.0 ± 0.1	12.5 ± 0.1	59.5 ± 0.3	49.2 ± 0.1	40.6
SagNet	57.7 ± 0.3	19.0 ± 0.2	45.3 ± 0.3	12.7 ± 0.5	58.1 ± 0.5	48.8 ± 0.2	40.3
ARM	49.7 ± 0.3	16.3 ± 0.5	40.9 ± 1.1	9.4 ± 0.1	53.4 ± 0.4	43.5 ± 0.4	35.5
VREx	47.3 ± 3.5	16.0 ± 1.5	35.8 ± 4.6	10.9 ± 0.3	49.6 ± 4.9	42.0 ± 3.0	33.6
RSC	55.0 ± 1.2	18.3 ± 0.5	44.4 ± 0.6	12.2 ± 0.2	55.7 ± 0.7	47.8 ± 0.9	38.9
SFT (Ours)	64.9 ± 0.0	22.0 ± 0.3	52.5 ± 0.1	16.3 ± 0.3	64.4 ± 0.1	55.6 ± 0.4	46.0

4. Full Results with ViT-B/16 and ViT-L/14

In this section, we show the full results of visual prompt tuning with the pre-trained large-scale vision transformers (including ViT-B/16 and ViT-L/14).

4.1. VLCS

Table 7. Out-of-domain accuracies (%) on each domain of VLCS and their average.

Backbone	Algorithms	C	L	S	V	Avg.
ViT-B/16	ERM	95.9	66.2	81.8	79.7	80.9
	IRM	96.5	67.7	85.0	78.7	81.9
	DANN	96.6	68.3	82.1	79.8	81.7
	CDANN	95.2	66.7	85.2	80.3	81.9
	CORAL	96.6	67.4	84.3	81.5	82.5
	MMD	95.4	67.3	83.2	81.6	81.9
	IIB	97.6	65.7	84.0	82.7	82.5
	SAM	96.6	68.1	84.9	84.5	83.5
	SFT (Ours)	96.8	68.8	84.8	85.8	84.1
ViT-L/14	ERM	95.8	66.8	86.7	82.4	82.9
	SAM	96.8	68.6	85.5	86.0	84.2
	SFT (Ours)	96.6	68.7	85.0	87.2	84.4

4.2. PACS

Table 8. Out-of-domain accuracies (%) on each domain of PACS and their average.

Backbone	Algorithms	A	C	P	S	Avg.
ViT-B/16	ERM	97.7	97.5	99.6	91.4	96.6
	IRM	98.2	96.7	99.8	90.7	96.4
	DANN	96.6	98.1	99.6	87.5	95.5
	CDANN	97.4	97.8	99.9	88.7	96.0
	CORAL	96.2	97.7	99.6	88.2	95.4
	MMD	96.9	97.7	99.6	86.0	95.1
	IIB	97.5	97.4	99.8	89.2	96.0
	SAM	97.8	98.1	99.7	88.9	96.1
	SFT (Ours)	98.2	98.8	99.9	90.2	96.8
ViT-L/14	ERM	99.2	99.5	99.9	96.6	98.8
	SAM	99.3	99.6	99.9	96.0	98.7
	SFT (Ours)	99.3	99.3	99.6	96.2	98.6

4.3. OfficeHome

Table 9. Out-of-domain accuracies (%) on each domain of OfficeHome and their average.

Backbone	Algorithms	A	C	P	R	Avg.
ViT-B/16	ERM	83.7	73.8	89.9	89.2	84.1
	IRM	81.4	73.2	88.9	88.9	83.1
	DANN	80.7	73.1	88.9	88.2	82.7
	CDANN	82.6	71.1	87.9	87.5	82.3
	CORAL	82.7	72.9	88.4	89.3	83.3
	MMD	83.5	73.0	89.5	88.6	83.7
	IIB	81.9	73.5	90.7	89.5	83.9
	SWAD	84.9	74.9	90.8	89.9	85.1
	SAM	84.2	76.6	91.0	90.8	85.7
	SFT (Ours)	86.3	77.7	91.1	90.9	86.5
ViT-L/14	ERM	89.8	83.4	93.9	93.8	90.2
	SAM	89.8	84.7	94.6	94.0	90.8
	SFT (Ours)	90.8	85.2	94.5	94.5	91.3

4.4. TerraIncognita

Table 10. Out-of-domain accuracies (%) on each domain of TerraIncognita and their average.

Backbone	Algorithms	L100	L38	L43	L46	Avg.
ViT-B/16	ERM	58.5	58.2	64.1	41.1	55.5
	IRM	45.7	53.5	55.4	48.8	50.9
	DANN	52.9	52.4	56.7	45.9	52.0
	CDANN	58.6	51.9	61.5	47.6	54.9
	CORAL	51.4	45.2	60.9	50.6	52.0
	MMD	57.5	57.1	62.0	50.9	56.9
	IIB	65.3	53.6	65.6	47.5	58.0
	SAM	64.6	52.0	61.5	48.3	56.6
	SFT (Ours)	70.7	58.3	65.3	50.5	61.2
ViT-L/14	ERM	65.1	55.1	69.6	55.4	61.3
	SAM	67.3	56.4	72.8	54.8	62.8
	SFT (Ours)	65.4	61.1	71.2	62.9	65.2

4.5. DomainNet

Table 11. Out-of-domain accuracies (%) on each domain of TerraIncognita and their average.

Backbone	Algorithms	clip	info	paint	quick	real	sketch	Avg.
ViT-B/16	ERM	77.6	44.4	66.4	18.8	81.2	66.7	59.2
	IRM	73.1	45.6	67.1	19.3	81.2	68.6	59.1
	DANN	74.9	42.9	67.9	19.1	79.2	67.3	58.6
	CDANN	74.7	44.5	66.1	19.2	79.2	67.0	58.4
	CORAL	77.6	44.7	66.6	19.1	81.0	68.1	59.5
	MMD	76.8	45.9	67.4	20.1	80.9	68.3	59.9
	IIB	76.5	42.4	66.5	18.5	79.9	67.6	58.6
	SAM	76.8	45.4	68.4	18.8	81.1	68.3	59.8
	SFT (Ours)	77.9	46.3	68.6	19.4	81.6	68.9	60.5
ViT-L/14	ERM	82.7	54.5	73.1	23.7	84.3	74.4	65.4
	SAM_VPT	83.1	51.9	73.3	23.1	85.3	74.7	65.2
	SFT (Ours)	83.2	55.0	75.2	24.6	86.1	75.3	66.5

5. Reproducibility

To guarantee reproducibility, we will provide an explanation of the code and hyperparameters in this section.

5.1. Code

Our work is built upon DomainBed, which is released under the MIT license. All experiments are conducted on a single NVIDIA Tesla V100 or A40.

5.2. Hyperparameters

5.2.1. Experiments on Toy Dataset

In our toy experiments, we generate a dataset with three classes (C_1, C_2 and C_3) and four domains (D_1, D_2, D_3 and D_4). There are 3×100 samples in each domain. We use data from the first three domains (D_1, D_2 and D_3) as the training domains, with the remaining domain D_4 serves as the test domain. For simplicity, we consider the covariance matrices to be diagonal with the same elements: $\mathbf{\Sigma}_i = \sigma_i^2\mathbf{I}$ and $\mathbf{\Sigma}_{ij} = \sigma_{ij}^2\mathbf{I}$. The specific parameters for each domain and class are provided in Table 12. During training, we use a linear classifier with the Adam optimizer. The batch size is set to 16 and the learning rate is 5e-4.

Table 12. Parameters for the generation of the toy dataset.

Classes	μ_i	σ_i	Domains	μ_{ij}	σ_{ij}
C_1	$(0, \sqrt{3}/2)$	0.4	D_1	(0.71,1.03)	0.2
			D_2	(-0.04,0.20)	0.2
			D_3	(0.08,1.22)	0.2
			D_4	(-0.52,0.54)	0.2
C_2	$(-1/2, 0)$	0.4	D_1	(-0.11,0.90)	0.2
			D_2	(-0.45,0.15)	0.2
			D_3	(-0.68,0.03)	0.2
			D_4	(-0.81,-0.11)	0.2
C_3	$(1/2, 0)$	0.4	D_1	(1.25,-0.39)	0.2
			D_2	(-0.20,0.52)	0.2
			D_3	(0.80,0.23)	0.2
			D_4	(0.83,-0.12)	0.2

5.2.2. Experiments on Real Dataset

The hyperparameter search spaces for ResNet-50, ViT-B/16 and ViT-L/14 are shown below. In the table, U and list indicate Uniform distribution and random choice, respectively.

Table 13. Hyperparameter search space for ResNet-50, ViT-B/16 and ViT-L/14.

Parameter	ResNet50	ViT-B/16	ViT-L/14
batch size	$U[24, 32]$	$U[16, 28]$	$U[8, 16]$
learning rate	$U[5.0e-6, 5.5e-5]$	$10^{U[-4.5,-3.0]}$	$10^{U[-4.5,-3.0]}$
ResNet dropout	[0.0,0.1,0.5]	–	–
weight decay	[1e-4, 1e-6]	0.0	0.0
ρ	[0.01, 0.02, 0.03, 0.05, 0.1]	[0.1, 0.2, 0.3, 0.5]	[0.05,0.1,0.2,0.3,0.5]
λ_1	$U[0, 1]$	$U[0, 0.5]$	$U[0, 0.5]$
λ_2	$U[0, 1]$	$U[0, 0.5]$	$U[0, 0.5]$
α	$10^{U[0.5,3]}$	$10^{U[0.5,3]}$	$10^{U[0.5,3]}$

6. Broader Impacts

This paper primarily focuses on developing an effective domain generalization method to address the problem of domain shifts. Given that domain shifts are ubiquitous in real-world applications, this work has the potential to make a positive impact by learning models that are less biased towards ethical aspects. We do not foresee any significant negative social impact of this work.

References

[1] David A McAllester. Pac-bayesian model averaging. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 164–170, 1999. 1