MARRS: Modern Backbones Assisted Co-training for Rapid and Robust Semi-Supervised Domain Adaptation (Supplementary Material)

The structure of the supplementary material is as follows:

1. Notations

Table 1 provides a summary of the notations used in the paper.

2. Algorithmic Details

2.1. Description of CORAL

CORAL [19] is an efficient domain alignment technique, which aligns source and target domain feature distributions by matching their covariance. In semi-supervised domain adaptation, CORAL will align feature distributions of labeled data (source data, target labeled data) and unlabeled

Table 1. **Table of notations**

	Symbol	Description
Network	$\mathcal{G}_c$	ConvNeXt-XL backbone
	$\mathcal{G}_s$	Swin-L backbone
	$\mathcal{F}_c$	Linear classifier, trained using features extracted from $\mathcal{G}_c$
	$\mathcal{F}_s$	Linear classifier, trained using features extracted from $\mathcal{G}_s$
	$\boldsymbol{w}_c$	Weights of classifier $\mathcal{F}_c$
	$\boldsymbol{w}_s$	Weights of classifier $\mathcal{F}_s$
Datasets / Feature sets	$\mathcal{D}_s$	Source dataset
	$\mathcal{D}_{tl}$	Target labeled dataset
	$\mathcal{D}_{tu}$	Target unlabeled dataset
	$\mathcal{D}_s^c$	Source feature set obtained from $\mathcal{G}_c$
	$\mathcal{D}_{tl}^c$	Target labeled feature set obtained from $\mathcal{G}_c$
	$\mathcal{D}_{tu}^c$	Target unlabeled feature set obtained from $\mathcal{G}_c$
	$\mathcal{D}_{tu'}^c$	Strongly augmented target unlabeled feature set from $\mathcal{G}_c$
	$\mathcal{D}_s^s$	Source feature set obtained from $\mathcal{G}_s$
	$\mathcal{D}_{tl}^s$	Target labeled feature set obtained from $\mathcal{G}_s$
	$\mathcal{D}_{tu}^s$	Target unlabeled feature set obtained from $\mathcal{G}_s$
	$\mathcal{D}_{tu'}^s$	Strongly augmented target unlabeled feature set from $\mathcal{G}_s$
	$\mathcal{D}_{co}^c$	Co-training feature set for training $\mathcal{F}_c$
	$\mathcal{D}_{co}^s$	Co-training feature set for training $\mathcal{F}_s$
	$\mathcal{D}_{cons}^c$	Consistency regularization feature set for training $\mathcal{F}_c$
	$\mathcal{D}_{cons}^s$	Consistency regularization feature set for training $\mathcal{F}_s$
	U^c	Pseudo labeled set obtained from $\mathcal{F}_c$
	U^s	Pseudo labeled set obtained from $\mathcal{F}_s$
Losses/factors	$\mathcal{L}_{\text{ce}}$	Cross-entropy loss
	$\mathcal{L}_{\text{lsr}}$	Cross-entropy loss with label-smoothing regularization
	λ_s	Scaling factor for source data loss
	λ_{tl}	Scaling factor for target labeled data loss
	λ_{co}	Scaling factor for co-training loss
	λ_{cons}	Scaling factor for consistency regularization loss
Samples / Features / Miscellaneous	(x_s, y_s)	Paired source samples
	(x_{tl}, y_{tl})	Paired target labeled samples
	x_{tu}	Unlabeled target samples
	$y_{ps,c}$	Pseudo label from set U^c
	$y_{ps,s}$	Pseudo label from set U^s
	$\mathbf{f}_s$	Source feature obtained from $\mathcal{G}_c$
	$\mathbf{f}_{tl}$	Target labeled feature obtained from $\mathcal{G}_c$
	$\mathbf{f}_{tu}$	Target unlabeled feature obtained from $\mathcal{G}_c$
	$\mathbf{f}_{tu'}$	Strongly augmented target unlabeled feature from $\mathcal{G}_c$
	$\mathbf{g}_s$	Source feature obtained from $\mathcal{G}_s$
	$\mathbf{g}_{tl}$	Target labeled feature obtained from $\mathcal{G}_s$
	$\mathbf{g}_{tu}$	Target unlabeled feature obtained from $\mathcal{G}_s$
	$\mathbf{g}_{tu'}$	Strongly augmented target unlabeled feature from $\mathcal{G}_s$
	N_{outer}	Outer iterations
	N_{inner}	Inner iterations
	N_{max}	Maximum number of iterations
	K	Number of classes
	τ	Confidence threshold
	η	Learning rate
	ψ	Weak augmentation
	ϕ	Strong augmentation

data (target unlabeled data). More precisely, CORAL first *whitens* labeled data feature distribution and then *re-color* it with the covariance of unlabeled data feature distribution. However, in our work, we use CORAL for introducing diversity at the feature distribution level by applying it at the features of only one of the backbones.

Algorithm 1: CORAL (correlation alignment)

Input : Source data features $\boldsymbol{F}_s$, target labeled data features $\boldsymbol{F}_{tl}$ and target unlabeled data features $\boldsymbol{F}_{tu}$

Output: Updated features $\boldsymbol{F}_s$, $\boldsymbol{F}_{tl}$

$\boldsymbol{F}_{lab} = \text{concatenate}\ (\boldsymbol{F}_s, \boldsymbol{F}_{tl})$

$\boldsymbol{F}_{lab} = \boldsymbol{F}_{lab} - \mathbb{E}[\boldsymbol{F}_{lab}]$

$\boldsymbol{F}_{tu} = \boldsymbol{F}_{tu} - \mathbb{E}[\boldsymbol{F}_{tu}]$

$\boldsymbol{C}_{lab} = \text{cov}(\boldsymbol{F}_{lab}) + \lambda I$

$\boldsymbol{C}_{tu} = \text{cov}(\boldsymbol{F}_{tu}) + \lambda I$

$\boldsymbol{F}_{lab} = \boldsymbol{F}_{lab} * \boldsymbol{C}_{lab}^{-1/2}$ `// whitening labeled data`

$\boldsymbol{F}_{lab} = \boldsymbol{F}_{lab} * \boldsymbol{C}_{tu}^{1/2}$ `// re-coloring labeled data`

Where, $\mathbb{E}[\cdot]$ and cov$(\cdot)$ represents the mean and covariance of the data features respectively. λ is the small constant added to the diagonal elements of the covariance matrix to make it an invertible matrix. Investigation of using other techniques [4, 5, 18] similar to CORAL in our framework (MARRS) are left for future exploration.

2.2. Weak augmentation details

In MARRS, we use perspective preserving padding as a weak data augmentation for image level diversity module. It preserves perspective while padding the image, which equalizes the dimensions of the resultant image. PyTorch-like pseudocode is given in Pseudo-code 1.

2.3. Initial training algorithm

In our framework, initially classifiers $\mathcal{F}_c$ and $\mathcal{F}_s$ are trained on their individual labeled feature sets. An outline of the initial training algorithm for $\mathcal{F}_c$ is given in Algorithm 2. The same can be obtained for $\mathcal{F}_s$ by using $\mathcal{D}_s^s$ as source feature set and $\mathcal{D}_{tl}^s$ as target labeled feature set.

2.4. Student model training algorithm

We use knowledge distillation [6] to transfer the knowledge from teacher model (MARRS-trained classifers) to student model ($i.e.$ MobileNetV2). Detailed steps of training the student model are given in Algorithm 3.

Pseudo-code 1: PyTorch-like pseudocode for perspective preserving padding

Input : Image

Output: Image with perspective preserving padding

```
// Calculating maximum dimension of image
maxd = max(image.size)
// Left and top padding size
pleft, ptop = [(maxd - s) // 2 for s in image.size]
// Right and bottom padding size
pright, pbottom = [maxd - (s+pad) for s, pad in
  zip(image.size, [pleft, ptop])]
// Combining all padding sizes
padding = (pleft, ptop, pright, pbottom)
// returning image after padding
return transforms.functional.pad(image, padding, 0,
  'constant')
```

Algorithm 2: Initial training

Input : Linear Classifier $\mathcal{F}$, parameters $\boldsymbol{w}$, learning rate η, source feature set $\mathcal{D}_s^c$, target labeled feature set $\mathcal{D}_{tl}^c$, weight balancing parameters λ_s, λ_{tl} and iterations N_{max}.

Output: updated parameters $\boldsymbol{w}$.

for $n \leftarrow 1$ **to** N_{max} **do**

  $\mathcal{L}_s = \mathcal{L}_{\text{ce}}\left(\mathcal{D}_s; \boldsymbol{w}\right)$

  $\mathcal{L}_{tl} = \mathcal{L}_{\text{ce}}\left(\mathcal{D}_{tl}; \boldsymbol{w}\right)$

  $\mathcal{L}_{lab} = \lambda_s * \mathcal{L}_s + \lambda_{tl} * \mathcal{L}_{tl}$

  $w = w - \eta \cdot \nabla \mathcal{L}_{lab}$

end

3. Training Details

3.1. Feature extractors details

In our framework MAARS, we use ConvNeXt-XL [11] and Swin-L [10] as two fixed feature extractors. Implementation of feature extractors in our work MARRS is based on the timm[1] python library. Details about feature extractors are given in Table 2.

3.2. Hyperparameter details

We set N_{inner} = N_{max} = 400 and N_{outer} = 30. We set the balancing hyperparameters as $\lambda_{co} = 0.9$, $\lambda_{cons} = 0.5$. Smoothing parameter $\epsilon = 0.1$ for DomainNet dataset and $\epsilon = 0.001$ for other datasets. The values of other hyperparameters $\lambda_s, \lambda_{tl}, \tau_s$ and τ_{tu} are taken from recent work [9] and set as $\lambda_s = 0.4$ for Algorithm 2 and $\lambda_s = 0.1$ otherwise, $\lambda_{tl} = 0.2$ for Algorithm 2 and $\lambda_{tl} = 0.05$ otherwise,

[1] https://github.com/rwightman/pytorch-image-models

Algorithm 3: Knowledge Distillation

Input : MARRS-trained classifiers $\mathcal{F}_c, \mathcal{F}_s$ with parameters $\boldsymbol{w}_c$ and $\boldsymbol{w}_s$ respectively, student model $\boldsymbol{w}$, learning rate η, iterations N, weight balancing parameters λ_s, λ_{tl} and λ_{tu}, Batch Size B, confidence threshold τ and datasets $\mathcal{D}_s, \mathcal{D}_{tl}$ and $\mathcal{D}_{tu}$.

Output: Student model $\boldsymbol{w}$.

for $n \leftarrow 1$ **to** N **do**
  Sample $S = \left\{\left(x_s^i, y_s^i\right)\right\}_{i=1}^{B}$ from $\mathcal{D}_s$.
  Sample $T = \left\{\left(x_{tl}^i, y_{tl}^i\right)\right\}_{i=1}^{B}$ from $\mathcal{D}_{tl}$.
  Sample $U = \left\{x_{tu}^i\right\}_{i=1}^{2B}$ from $\mathcal{D}_{tu}$.
  Set $U_{pl} = \emptyset$. `// empty pseudo-label set`
  $\mathcal{L}_s = \mathcal{L}_{\text{ce}}\left(S; \boldsymbol{w}\right)$
  $\mathcal{L}_{tl} = \mathcal{L}_{\text{ce}}\left(T; \boldsymbol{w}\right)$
  `// Calculating confident and agreed pseudo labels`
  for $i \leftarrow 1$ **to** $2B$ **do**
    $f_i \leftarrow$ extracted feature of x_{tu}^i for $\mathcal{F}_C$.
    $g_i \leftarrow$ extracted feature of x_{tu}^i for $\mathcal{F}_s$.
    $p_c = \max_k p\left(k \mid f_i; \boldsymbol{w}_c\right)$.
    $p_s = \max_k p\left(k \mid g_i; \boldsymbol{w}_s\right)$.
    $\hat{y}_c = \arg\max_k p\left(k \mid f_i; \boldsymbol{w}_c\right)$
    $\hat{y}_s = \arg\max_k p\left(k \mid g_i; \boldsymbol{w}_s\right)$
    if $\left(\hat{y}_c = \hat{y}_s$ ***and*** $p_c > \tau$ ***and*** $p_s > \tau\right)$
      Update $U_{pl} \leftarrow U_{pl} + \left\{\left(x_{tu}^i, \hat{y}_c\right)\right\}$.
    end
  end
  $\mathcal{L}_{tu} = \mathcal{L}_{\text{ce}}\left(U_{pl}; \boldsymbol{w}\right)$
  $\mathcal{L}_{total} = \lambda_s * \mathcal{L}_s + \lambda_{tl} * \mathcal{L}_{tl} + \lambda_{tu} * \mathcal{L}_{tu}$
  $\boldsymbol{w} = \boldsymbol{w} - \eta \cdot \nabla \mathcal{L}_{total}$
end

$\tau_s = 0.8$ and $\tau_{tu} = 0.9$ (initially, and decrease by 0.1 after each 10 N_{outer} iterations). In knowledge distillation, we set the value of τ to 0.7 and the values of balancing hyperparameters λ_s, λ_{tl} and λ_{tu} are set as 0.1, 0.05 and 0.9 respectively, which are same as their previous values.

4. Additional Results

4.1. Feasibility analysis of co-training

To ensure the effectiveness of co-training in our framework, we determine whether the linear classifiers in our framework satisfy the relaxed version of the ϵ- expandability [1, 2] condition or not. For this, we removed co-training from our framework MARRS and replaced it with self-training [8, 12]. Now, each classifier will use its own pseudo-label sets to construct two labeled co-training feature sets, namely $\mathcal{D}_{co}^c$ and $\mathcal{D}_{co}^s$ (cf. Subsection 3.2 of the main paper). After training both classifiers $\mathcal{F}_c$ and $\mathcal{F}_s$, we apply each classifier on the entire unlabeled dataset, as $\mathcal{F}_c$ on $\mathcal{D}_{tu}^c$ and $\mathcal{F}_s$ on $\mathcal{D}_{tu}^s$. So after applying $\mathcal{F}_c$ on $\mathcal{D}_{tu}^c$, we calculate the binary confidence indicator for each $u \in \mathcal{D}_{tu}^c$, as:

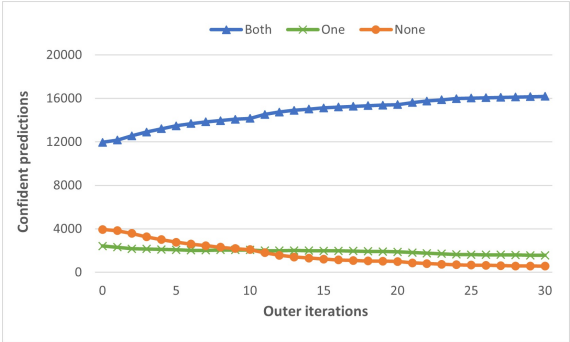


Figure 1. **Feasibility analysis of effective co-training** on the Real to Clipart, 3-shot adaptation task on the DomainNet [14] dataset. The numbers Both, One, and None indicate the proportion of unlabeled examples in which both, exactly one and none of the classifiers have confidence respectively.

$$\mathcal{M}_c(u) = \begin{cases} 1 & \text{if } \max_k p\left(k \mid u; \boldsymbol{w}_c\right) > \tau, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The value of binary confidence indicator will be 1 for confident samples whose maximal probability score lies above a threshold (τ = 0.5) and 0 otherwise. Similarly, we can apply $\mathcal{F}_s$ on $\mathcal{D}_{tu}^s$ and calculate the binary confidence indicator $\mathcal{M}_s(u)$ for each $u \in \mathcal{D}_{tu}^s$. Now, we calculate three values N_{none}, N_{one} and N_{both} representing the number of unlabeled examples on which *none*, *exactly one* and *both* of the classifiers have confidence respectively. These three values are formulated as:

$$\begin{aligned} N_{\text{none}} &: \sum_{u \in \mathcal{D}_{tu}^c, u' \in \mathcal{D}_{tu}^s} \bar{\mathcal{M}}_c(u)\bar{\mathcal{M}}_s(u'), \\ N_{\text{one}} &: \sum_{u \in \mathcal{D}_{tu}^c, u' \in \mathcal{D}_{tu}^s} \mathcal{M}_c(u)\bar{\mathcal{M}}_s(u') + \bar{\mathcal{M}}_c(u)\mathcal{M}_s(u'), \\ N_{\text{both}} &: \sum_{u \in \mathcal{D}_{tu}^c, u' \in \mathcal{D}_{tu}^s} \mathcal{M}_c(u)\mathcal{M}_s(u'), \end{aligned} \tag{2}$$

Where, $\bar{\mathcal{M}}_c(u) = 1 - \mathcal{M}_c(u)$ and $\bar{\mathcal{M}}_s(u') = 1 - \mathcal{M}_s(u')$. We ran the experiment on Real to Clipart, 3-shot setting on the DomainNet [14] dataset in which $|\mathcal{D}_s|$ = 70,358, $|\mathcal{D}_{tu}|$ = 18,325 and $|\mathcal{D}_{tl}|$ = 378, and then we verified the relaxed version of the ϵ-expandability [1, 2] condition which can be written as:

$$N_{\text{one}} \geq \epsilon \min\left(N_{\text{both}}, N_{\text{none}}\right), \tag{3}$$

for some $\epsilon > 0$. Fig. 1 demonstrates the results, where after the end of 30 outer iterations, the values of N_{none}, N_{one}

Table 2. **Feature extractors details**

Feature extractor	Input resolution	Output resolution	Pretrain	Finetune
ConvNeXt-XL	384 × 384	2048	ImageNet 22K	ImageNet 1K
Swin-L	384 × 384	1536	ImageNet 22K	ImageNet 1K

Table 3. **Accuracy (%) on *DomainNet* dataset under the 5-shot and 10-shot settings**. MARRS* (student model) uses MobileNetV2 (3.4 M parameters), and all other baselines use ResNet34 (22 M parameters). 5-shot (10-shot) represents that 5 target labeled samples (10 target labeled samples) per class are available during training. The best results are in **bold**, and the second-best results are underlined.

Method	R→C		R→P		P→C		C→S		S→P		R→S		P→R		**Mean**	
	5-sh	10-sh	5-sh	10-sh	5-sh	10-sh	5-sh	10-sh	5-sh	10-sh	5-sh	10-sh	5-sh	10-sh	5-sh	10-sh
S+T	64.5	68.5	63.1	66.4	64.2	69.2	59.2	64.8	60.4	64.2	56.2	0.7	75.7	77.3	63.3	67.3
ENT	77.1	79.0	71.0	72.9	75.7	78.0	61.9	68.9	66.2	68.4	64.6	68.1	81.1	82.6	71.1	74.0
MME	75.5	77.1	70.4	71.9	74.0	76.3	65.0	67.0	68.2	69.7	65.5	67.8	79.9	81.2	71.2	73.0
CLDA	80.3	81.2	76.0	77.7	77.8	80.3	71.6	74.1	74.5	77.1	72.9	74.1	84.0	85.1	76.7	78.5
CDAC	80.8	83.1	75.3	77.2	79.9	81.7	72.1	74.3	74.7	76.3	72.9	74.6	83.2	84.7	76.9	78.9
MARRS*	86.2	87.6	84.2	85.5	86.5	87.7	77.9	**79.0**	84.2	85.3	76.5	78.1	91.0	91.8	83.8	85.0
MARRS	**86.6**	**87.9**	**85.8**	**86.9**	**86.8**	**87.8**	**78.2**	**79.0**	**86.3**	**87.2**	**77.3**	**79.0**	**92.9**	**93.5**	**84.8**	**85.9**

Table 4. **Accuracy (%) on *Office-31***. MARRS* (student model) uses MobileNetV2 (3.4 M parameters), and all other baselines use AlexNet (61 M parameters). The best results are in **bold** and the second-best results are underlined.

Method	W → A		D → A		**Mean**	
	1-shot	3-shot	1-shot	3-shot	**1-shot**	**3-shot**
S+T	50.4	61.2	50.0	62.4	50.2	61.8
ENT	50.7	64.0	50.0	66.2	50.4	65.1
MME	57.2	67.3	55.8	67.8	56.5	67.6
CDAC	63.4	70.1	62.8	70.0	63.1	70.0
CLDA	64.6	70.5	62.7	72.5	63.6	71.5
MARRS*	88.4	89.5	**87.5**	**89.9**	**88.0**	89.7
MARRS	**88.5**	**89.7**	**87.5**	**89.9**	**88.0**	**89.8**

and N_{both} are 570, 1563 and 16,192 respectively. Our two linear classifiers satisfy the Eq. (3) with $\epsilon = 2.74$, which again demonstrates the usefulness of the proposed diversity module in inducing diversity among classifiers, which makes classifiers diverse enough to become applicable for applying co-training efficiently.

4.2. Results on larger shots

We conducted additional experiments by using higher number of labeled target samples per class during training *i.e.* 5-shot and 10-shot adaptation tasks on the DomainNet dataset using the data splits provided by APE [7]. As can be seen in Table 3, our framework MARRS and MARRS* (student model) consistently outperforms previous SSDA approaches with large margin across all adaptation scenarios under the both 5-shot and 10-shot settings showing the dominance of our framework (MARRS) and student model (MARRS*) in even larger shot settings.

Table 5. **Performance of MARRS with different modern backbones** on mean test accuracy (%) of DomainNet, 3-shot task. Here, $\mathcal{G}_c$ and $\mathcal{G}_s$ represent the first backbone (where weak augmentation is applied) and second backbone (where CORAL is applied) respectively.

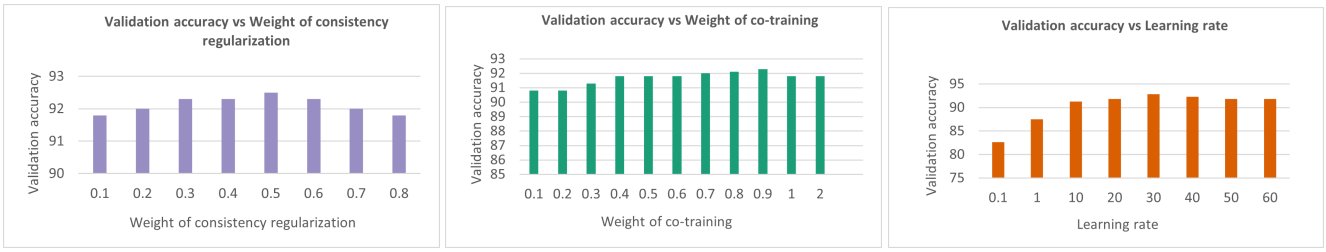
$\mathcal{G}_c$	$\mathcal{G}_s$	Accuracy
ConvNeXt-B	Swin-B	83.4
ConvNeXt-B	VIT-L	83.7
ConvNeXt-L	Swin-B	83.5
ConvNeXt-L	VIT-L	83.8
ConvNeXt-B	ConvNeXt-L	83.1
VIT-L	Swin-B	82.8

4.3. Results on Office-31

Office-31 [16] is a small size adaptation dataset consists of 31 categories belonging to 3 domains, namely DSLR (D), Webcam (W) and Amazon (A). Following [17] we report results on two adaptation scenarios namely W → A and D → A. As shown in Table 4, MARRS and MARRS* achieve nearly the same performance under both 1-shot and 3-shot settings. The MARRS achieves mean accuracies of 88.0% and 89.8%, which outperforms the previously best performing method CLDA by a large margin of 24.4% and 18.3% in 1-shot and 3-shot settings.

4.4. MARRS with other modern backbones

In MARRS, we use ConvNeXt-XL and Swin-L as two modern backbones. Here, XL and L denote extra large and large respectively, which represents the size of backbone. We choose ConvNeXt-XL and Swin-L in our work based on the recent analysis reported by [15], which ex-



(a) (b) (c)

Figure 2. **Hyperparameter analysis** on Clipart to Art (C $\rightarrow$ A), 1-shot adaptation task of the OfficeHome dataset. (a) Effect of changing the weight of consistency regularization (λ_{cons}) on the validation accuracy. (b) Effect of changing the weight of co-training (λ_{co}) on the validation accuracy. (c) Effect of changing the learning rate (η) on the validation accuracy.

Table 6. **Analysis of co-training** on test accuracy (%) of *DomainNet*, 3-shot task. $\mathcal{F}_c$ ($\mathcal{F}_s$) are ConvNeXt-XL (Swin-L) based classifiers.

Method	Model	R→C	R→P	P→C	C→S	S→P	R→S	P→R	Mean
MARRS (w/o co-training)	$\mathcal{F}_c$	81.6	83.5	82.8	76.0	84.6	75.0	92.3	82.3
	$\mathcal{F}_s$	83.1	83.8	83.4	73.5	83.4	73.8	91.6	81.8
	Ensemble	84.4	85.4	85.4	77.2	85.7	76.5	92.6	83.9
MARRS	$\mathcal{F}_c$	85.3	85.1	85.8	77.1	85.5	76.7	92.7	84.0
	$\mathcal{F}_s$	85.2	85.3	85.7	76.9	85.1	76.7	92.8	84.0
	Ensemble	85.5	85.9	86.1	77.6	86.1	77.3	92.9	**84.5**

hibits their outstanding performance across various tasks like: Out-of-Distribution Detection [20], model calibration [13] etc. In Table 5, we also show the analysis of MARRS with relatively smaller or different modern backbones. From the family of Vision Transformers, we choose VIT-L [3] and Swin-B, while from the fully CNN based family, we pick relatively smaller variants of ConvNeXt [11] namely, ConvNeXt-L and ConvNeXt-B. The first four rows in Table 5 show that even with other modern backbones, MARRS is consistently outperforming previous ResNet34 based works. Note that all the first four settings also outperform PACE [9], which uses 28 modern backbones, including ConvNeXt-XL and Swin-L, to obtain a mean accuracy of 83.3% for DomainNet 3-shot task. This suggests that our framework, MARRS is highly stable. However, the gap in performance with respect to PACE becomes thinner due to the use of relatively less strong and small modern backbones. The performance decline in the last two rows highlights the significance of the backbone level diversity module, which emphasizes that backbones from different family (i.e. one backbone from CNN family and other from Vision Transformers family) should be employed for larger performance gains.

4.5. Significance of co-training

We analyze the significance of co-training in our framework by comparing MARRS to MARRS without co-training. The difference between the two approaches is in how co-training datasets ($\mathcal{D}^c_{co}$ and $\mathcal{D}^s_{co}$) will be generated. In MARRS without co-training, each classifier will use its own pseudo-label sets to generate co-training datasets. As can be seen in Table 6, MARRS performs better than MARRS w/o co-training on both ensemble and each model alone. Notably, each model of MARRS also outperforms ensemble of MARRS w/o co-training, which demonstrates the power of two models exchanging their strengths for mutual gain.

4.6. Multiple runs results

We report multi-run results for DomainNet 3-shot, Office-Home 3-shot domain adaptation tasks in Table 7 and Table 8 respectively. In particular, we conducted three runs of our framework MARRS using three different seeds $\{123, 1234, 12345\}$ and reported mean performance and standard deviation. Low standard deviation in both DomainNet 3-shot and Office-Home 3-shot domain adaptation tasks shows the reliability of our framework MARRS.

4.7. Hyperparameter analysis

We utilize validation performance on the complex, 1-shot C $\rightarrow$ A (Clipart to Art) task of the OfficeHome dataset for deciding hyperparameters values. From Fig. 2a, we notice that validation accuracy increases on increasing weight of consistency regularization (λ_{cons}) and achieves the peak value of 92.5% at λ_{cons} = 0.5. Nonetheless, a difference of just 0.7% between the greatest and lowest validation accuracy demonstrates that our method also performs well with other values of λ_{cons} and is less sensitive to it. Simi-

Table 7. **Multiple run results on *DomainNet*, 3-shot task**. Mean performance and standard deviation are reported based on the three runs of our framework MARRS using three different seeds $\{123, 1234, 12345\}$.

Method	R→C	R→P	P→C	C→S	S→P	R→S	P→R	**Mean**
MARRS	85.5±0.1	85.93±0.06	86.3±0.17	77.47±0.15	86.07±0.06	77.3±0.00	92.9±0.00	84.5±0.00

Table 8. **Multiple run results on *OfficeHome*, 3-shot task**. Mean performance and standard deviation are reported based on the three runs of our framework MARRS using three different seeds $\{123, 1234, 12345\}$.

Method	R→C	R→P	R→A	P→R	P→C	P→A	A→P	A→C	A→R	C→R	C→A	C→P	**Mean**
MARRS	87.03±0.06	96.03±0.06	90.67±0.11	95.3±0	86.43±0.11	90.8±0.1	95.6±0	86.57±0.06	94.77±0.06	95.23±0.06	92±0.17	95.03±0.06	92.13±0.06

larly, from Fig. 2b, we can see that the validation accuracy achieves its maximum values of 92.3% at λ_{co} = 0.9, and it starts to decrease on further rising value of λ_{co}. Nevertheless, except from early values of λ_{co} such as 0.1 and 0.2, the performance of MARRS is steady and does not fluctuate greatly when the value of λ_{co} is changed, indicating that our method is likewise not sensitive to the value of λ_{co}. From Fig. 2c, we find that intermediate values of learning rate (η) like 30 and 40 give the best validation accuracy relative to lower and larger values of η.

References

[1] Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. In *Advances in Neural Information Processing Systems*, 2011. 3

[2] Minmin Chen, Kilian Q. Weinberger, and Yixin Chen. Automatic feature decomposition for single view co-training. In *International Conference on Machine Learning*, 2011. 3

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 5

[4] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *IEEE International Conference on Computer Vision*, pages 2960–2967, 2013. 2

[5] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073, 2012. 2

[6] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015. 2

[7] Taekyung Kim and Changick Kim. Attract, perturb, and explore: Learning a feature alignment network for semi-supervised domain adaptation. In *European Conference on Computer Vision*, 2020. 4

[8] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013. 3

[9] Christopher Liao, Theodoros Tsiligkaridis, and Brian Kulis. Pick up the pace: Fast and simple domain adaptation via ensemble pseudo-labeling. *ArXiv*, abs/2205.13508, 2022. 2, 5

[10] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE International Conference on Computer Vision*, pages 9992–10002, 2021. 2

[11] Zhuang Liu, Hanzi Mao, Chaozheng Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11966–11976, 2022. 2, 5

[12] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, 2006. 3

[13] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Ann Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. In *Advances in Neural Information Processing Systems*, 2021. 5

[14] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *IEEE International Conference on Computer Vision*, pages 1406–1415, 2019. 3

[15] Francesco Pinto, Philip H. S. Torr, and Puneet Kumar Dokania. An impartial take to the cnn vs transformer robustness contest. In *European Conference on Computer Vision*, 2022. 4

[16] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European Conference on Computer Vision*, 2010. 4

[17] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *IEEE International Conference on Computer Vision*, pages 8049–8057, 2019. 4

[18] Suranjana Samanta and Sukhendu Das. Unsupervised domain adaptation using eigenanalysis in kernel space for categorisation tasks. *IET Image Process.*, 9:925–930, 2015. 2

[19] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016. 1

[20] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *ArXiv*, abs/2110.11334, 2021. 5