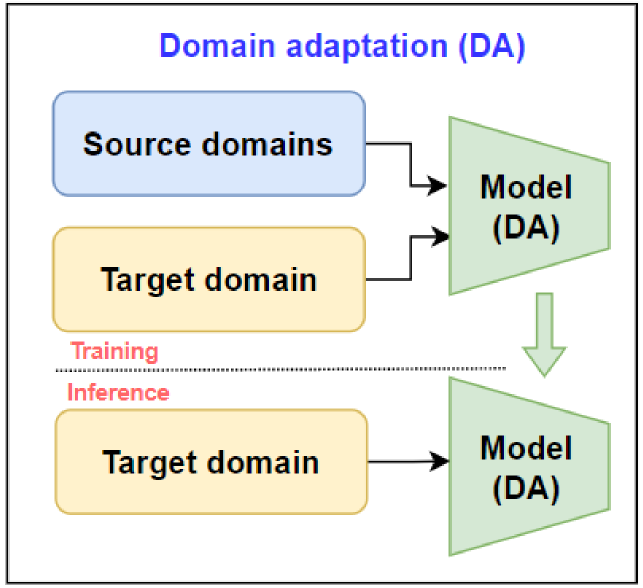
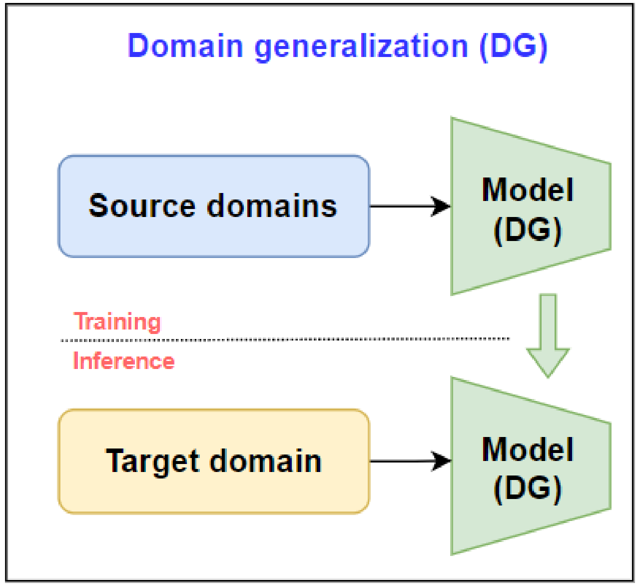
Test-Time Domain Adaptation by Learning Domain-Aware Batch Normalization

Yanan Wu* Zhixiang Chi* Yang Wang Konstantinos N. Plataniotis Songhe Feng

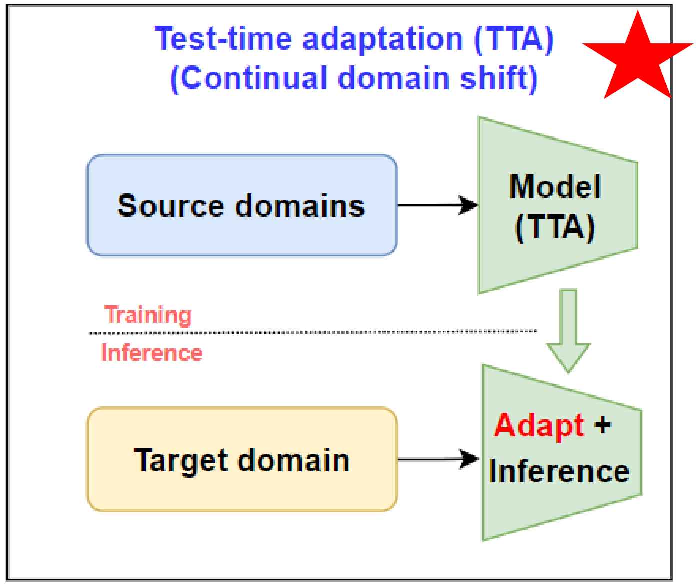
Problem setting for Test-Time Domain Adaptation (TT-DA)



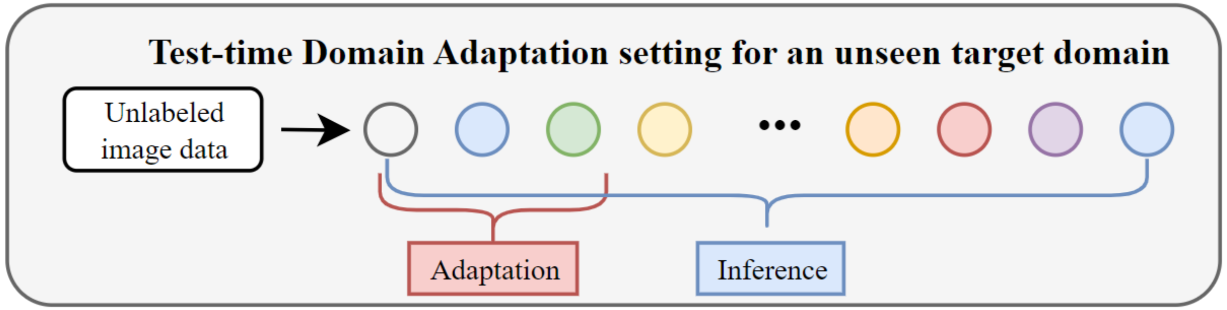
(-) Require huge amount of unlabeled target data
(-) Large-scale repetitive training



(+) One model to tackle all domains
(-) Fail to exploit domain specific information in target domains



(+) Exploit domain specific information in ***unseen*** target domains



(+) For each target domain, only adapt ***once*** using ***few unlabeled data***

Assumption: few unlabeled data convey the underlying distribution of that domain.

Challenges in TT-DA

We seek a simple yet effective solution for CNN-based networks

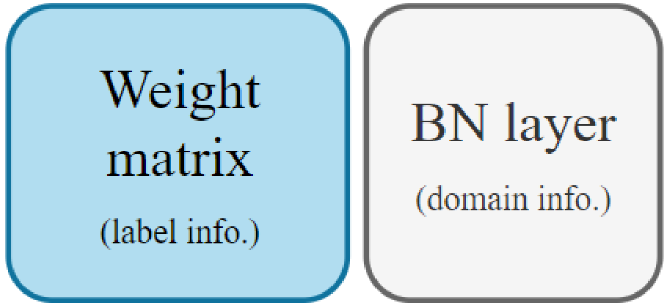
To update from few-shot unlabeled data:

1. Which parameters to update?
2. How to determine the supervision?
3. Effective training strategy?

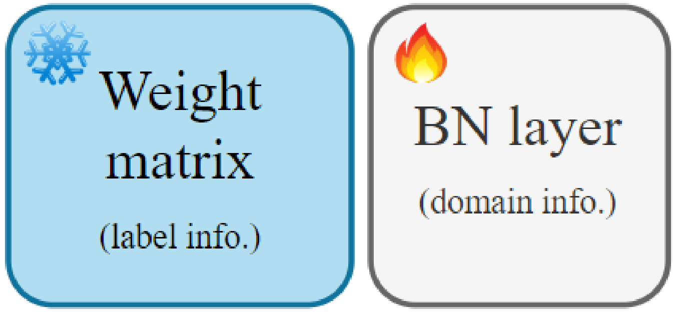
Knowledge Disentanglement

Prior research on knowledge disentanglement [1]:

- ➢ Label information -> weight matrix
- ➢ Domain information -> BN layer



- ❖ Intuition for TT-DA:
  - ❖ All domains share the label space.
    - ❖ Share semantic information.
  - ❖ Every data in a domain is drawn from the same distribution. (e.g., same style of drawing)
    - ❖ Require domain-specific knowledge.



- ➢ Keep the well-acquired label knowledge undisturbed
- ➢ Maximize domain-specific knowledge extraction

[1] Li et al. Revisiting batch normalization for practical domain adaptation, ICLR 2016.

Learning Domain-specific Knowledge

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

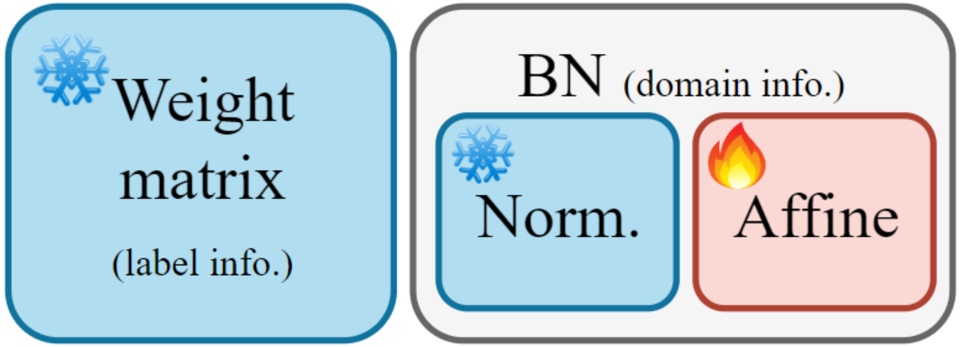
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

❖ Batch normalization:

1. Normalization
   Unstable on few-shot data.
2. Affine transformation (learnable)

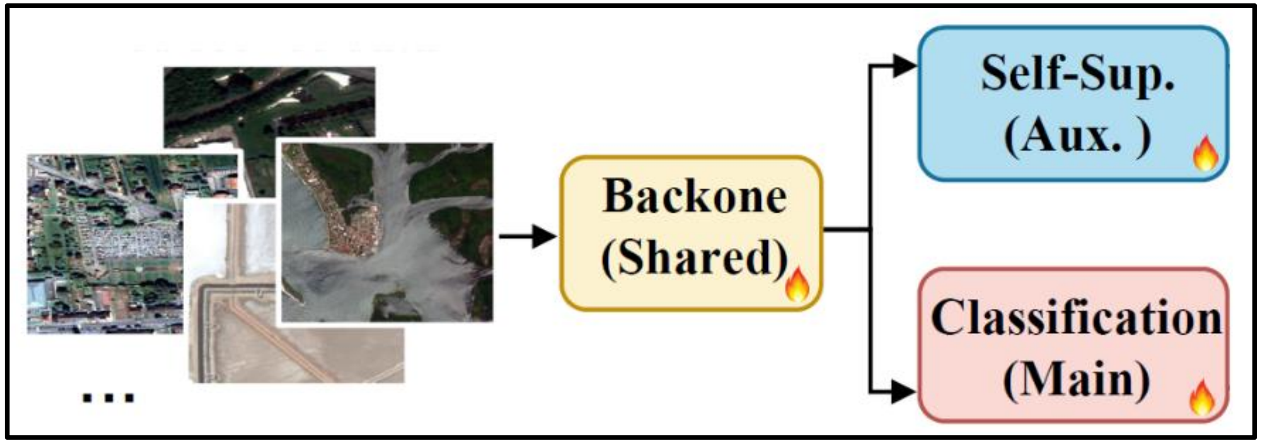


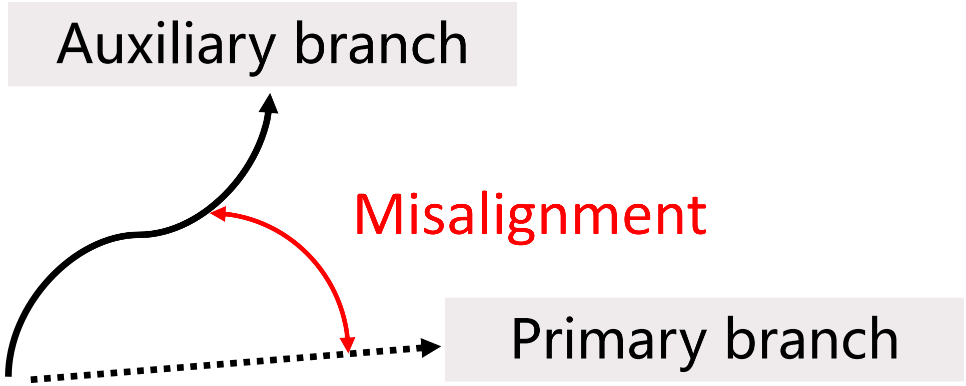
❖ Directly adopt normalization statistics from source data.
❖ Use affine parameters for correction.

Supervision?

Recap:
- ➢ Unlabeled data
- ➢ Extract domain information only

⟹ Class-independent self-supervised loss





- ❖ Formulate the learning pipeline as multi-task learning
- ❖ Network updated via auxiliary branch benefits the main branch

Domain-centric Learning to Adapt

Goal:

- Focus on the domain-level rather than the dataset/instance level.
- Enforce to learn the domain-specific knowledge
- Learning objective alignment (bi-level optimization)

Algorithm 1 Domain-centric learning (framework)

Require: $\{\mathcal{D_S}^i\}_{i=1}^N$: data of source domains; θ: learnable parameters

Initialize: θ

while *not converged* **do**

    Sample a meta batch of B source domains $\{\mathcal{D_S}^b\}^B$

    // **Inner loop: independently adapt to each domain**

    for each $\mathcal{D_S}^b$ **do**

        Adapt θ to domain $\mathcal{D_S}^b$ and evaluate

        Accumulate adaptation loss

    end for

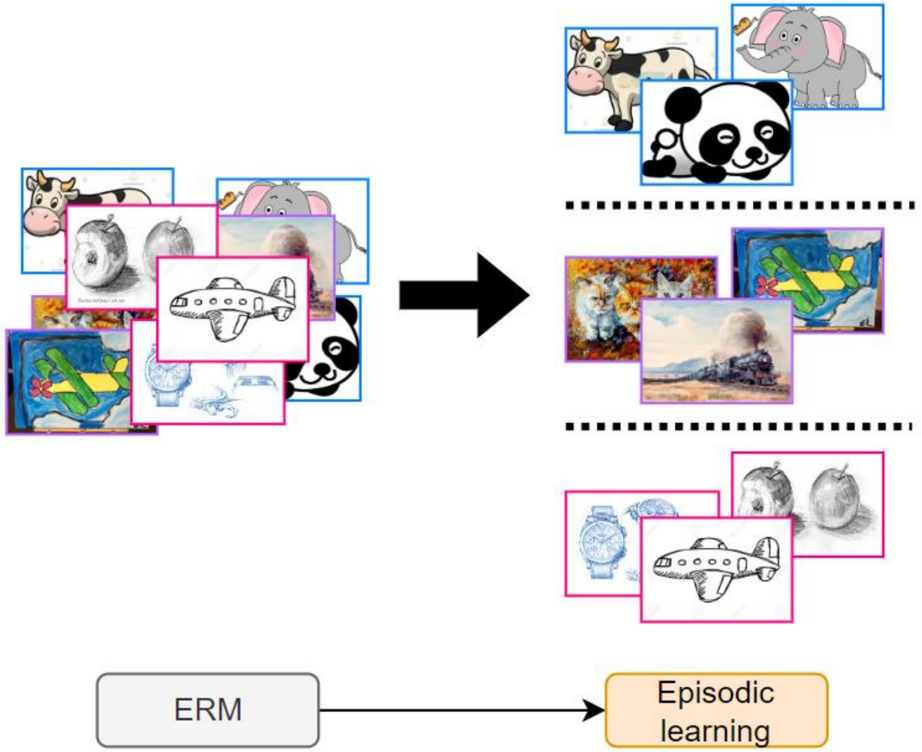
    // **Outer loop: meta update regarding adaptation results**

    Update θ for the current meta batch:

    $\theta \leftarrow \theta - \beta \nabla_\theta \text{loss}$

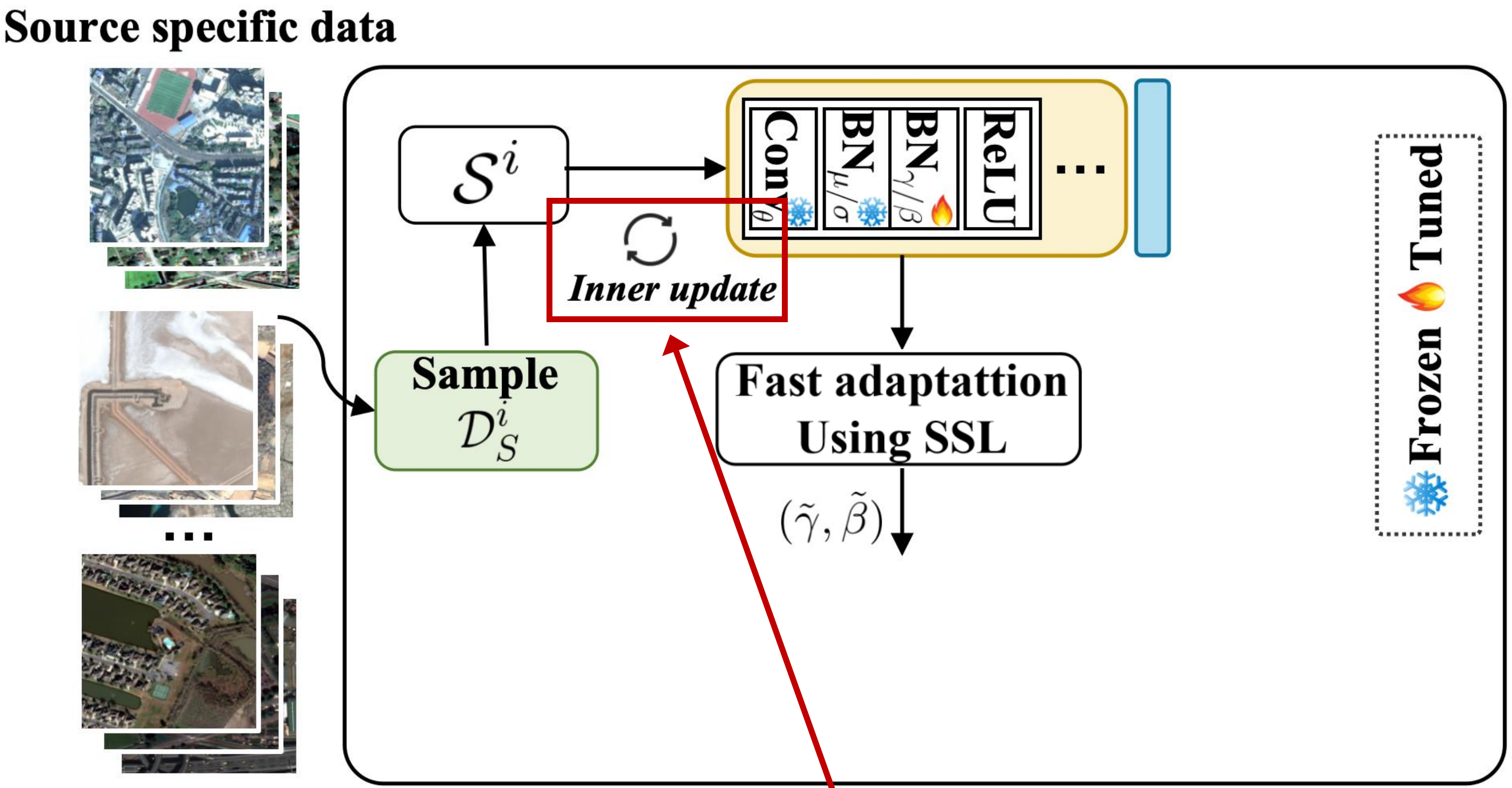
end while

*Methods and setting-dependent



Learning to Adapt (second stage)

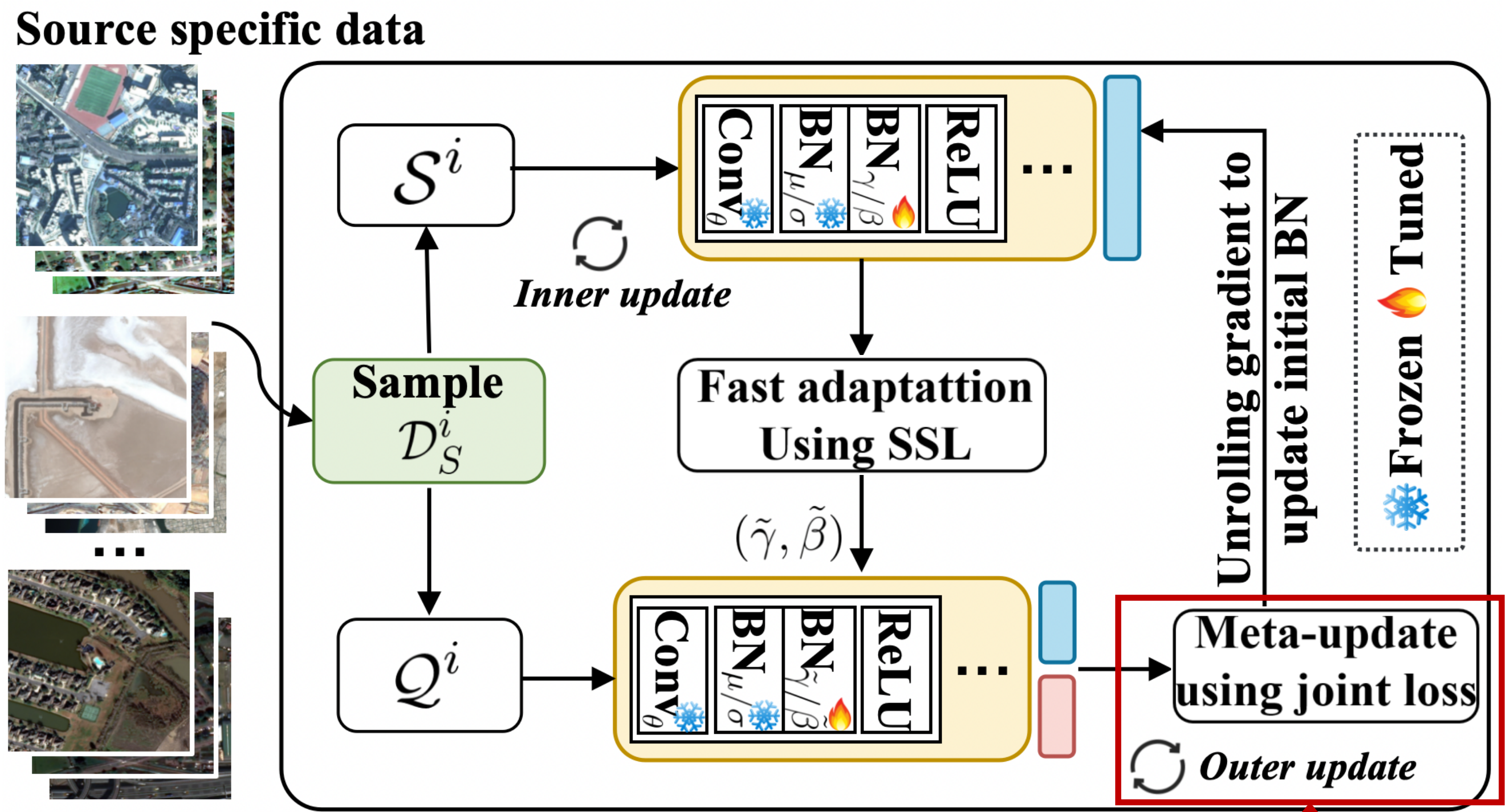
❖ Task formulation: adapting to every domain using a few unlabeled data



$$(\tilde{\gamma}, \tilde{\beta}) = (\gamma, \beta) - \alpha \nabla_{(\gamma,\beta)} \mathcal{L}_{SSL}(\mathcal{S}^i; \theta, (\gamma, \beta))$$

Learning to Adapt (second stage)

❖ Meta-objective: evaluate the adapted affine parameters on a disjoint set in the task.



$$\boxed{(\gamma, \beta) \leftarrow (\gamma, \beta) - \delta \nabla_{(\gamma,\beta)} \mathcal{L}_{Joint}(\mathcal{Q}^i; \theta, (\tilde{\gamma}, \tilde{\beta}))}$$

Learning to Adapt (second stage)

Algorithm 1: Meta-auxiliary training of MABN

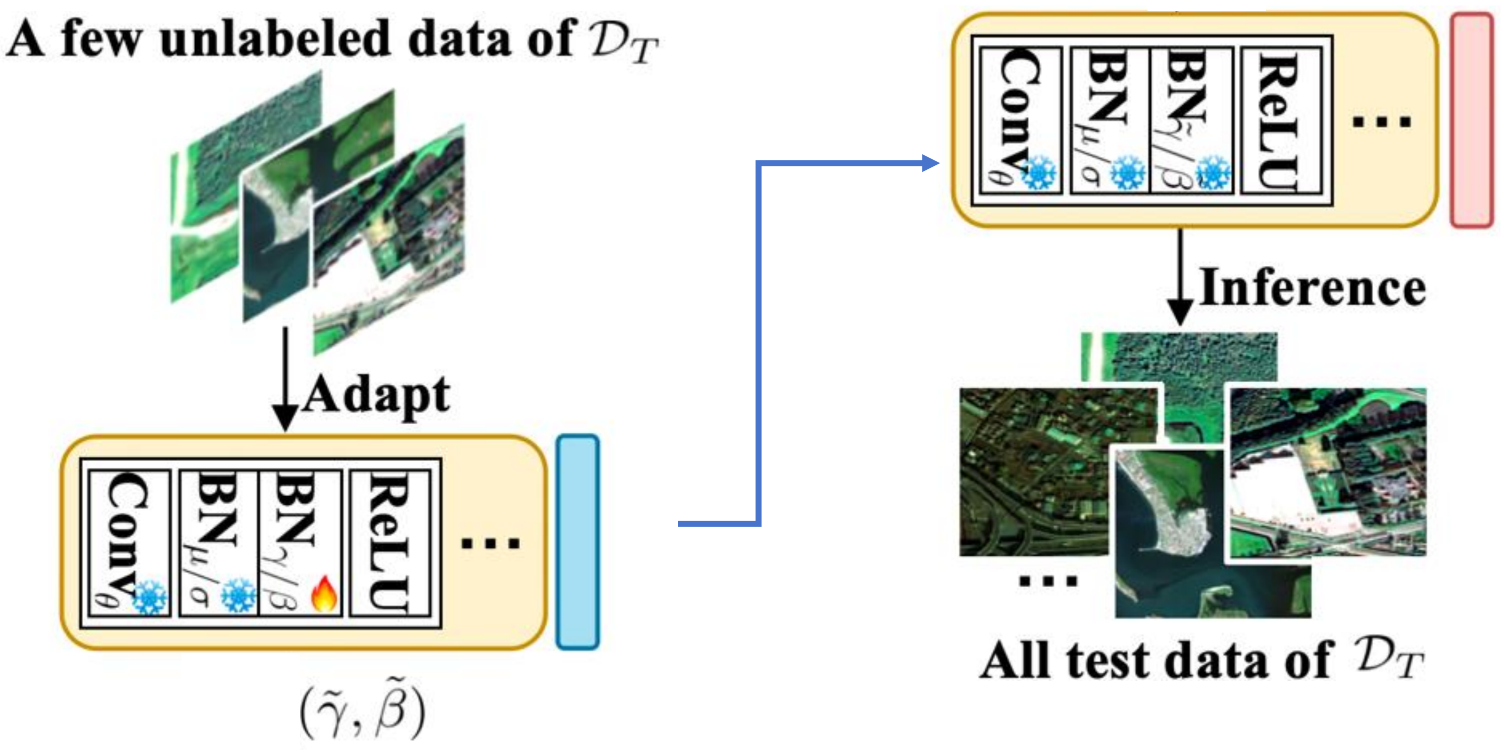
Require: α, δ, η: learning rates; B: meta batch size
Require: $\{\mathcal{D}_S^i\}_{i=1}^M$: data of source domains
Initialize weight matrix θ and affine params (γ, β)
// Learning label representation on mixed source data
$(\theta, \gamma, \beta) \leftarrow (\theta, \gamma, \beta) - \eta \nabla_{(\theta,\gamma,\beta)} \mathcal{L}_{Joint}(\mathcal{D}_S; \theta; (\gamma, \beta))$
while not converged **do**
  // Learning to adapt to domain-specific knowledge
  Sample a meta batch of B source domains: $\{\mathcal{D}_S^i\}_{i=1}^B$
  Reset the loss of current meta batch: $\mathcal{L}_B = 0$
  for each $\mathcal{D}_S^i$ in $\{\mathcal{D}_S^i\}_{i=1}^B$ **do**
    Sample support and query set: $(\mathcal{S}^i, \mathcal{Q}^i) \sim \mathcal{D}_S^i$
    // Perform adaptation via self-supervised loss
    $(\tilde{\gamma}, \tilde{\beta}) = (\gamma, \beta) - \alpha \nabla_{(\gamma,\beta)} \mathcal{L}_{SSL}(\mathcal{S}; \theta; (\gamma, \beta))$
    // Evaluate the adapted $(\tilde{\gamma}, \tilde{\beta})$ using $\mathcal{Q}$ and
    // accumulate the loss
    $\boxed{\mathcal{L}_B = \mathcal{L}_B + \mathcal{L}_{Joint}(\mathcal{Q}; \theta; (\tilde{\gamma}, \tilde{\beta})^{S,A})}$
  end for
  // Update (γ, β) for current meta batch
  $(\gamma, \beta) \leftarrow (\gamma, \beta) - \delta \nabla_{(\gamma,\beta)} \mathcal{L}_B$
end while

Training and evaluation protocol alignment via simulation

Learning objective alignment

Test-time Domain Adaptation on Unseen Domains

❖ Acquire domain-specific knowledge via auxiliary branch followed by inference



$$\boxed{(\tilde{\gamma}, \tilde{\beta}) = (\gamma, \beta) - \alpha \nabla_{(\gamma,\beta)} \mathcal{L}_{SSL}(\mathcal{S}^i; \theta, (\gamma, \beta))}$$

WILDS

Results: comparison with the state-of-the-art

Methods	iWildCam Acc	iWildCam Macro F1	Camelyon17 Acc	RxRx1 Acc	FMoW WC Acc	FMoW Avg Acc	PovertyMap WC Pearson r	PovertyMap Pearson r
ERM	71.6±2.5	31.0±1.3	70.3±6.4	29.9±0.4	32.3±1.25	53.0±0.55	0.45±0.06	0.78±0.04
CORAL	73.3±4.3	32.8±0.1	59.5±7.7	28.4±0.3	31.7±1.24	50.5±0.36	0.44±0.06	0.78±0.05
Group DRO	72.7±2.1	23.9±2.0	68.4±7.3	23.0±0.3	30.8±0.81	52.1±0.5	0.39±0.06	0.75±0.07
IRM	59.8±3.7	15.1±4.9	64.2±8.1	8.2±1.1	30.0±1.37	50.8±0.13	0.43±0.07	0.77±0.05
ARM-CML	70.5±0.6	28.6±0.1	84.2±1.4	17.3±1.8	27.2±0.38	45.7±0.28	0.37±0.08	0.75±0.04
ARM-BN	70.3±2.4	23.7±2.7	87.2±0.9	31.2±0.1	24.6±0.04	42.0±0.21	0.49±0.21	**0.84±0.05**
ARM-LL	71.4±0.6	27.4±0.8	84.2±2.6	24.3±0.3	22.1±0.46	42.7±0.71	0.41±0.04	0.76±0.04
Meta-DMoE	77.2±0.3	34.0±0.6	91.4±1.5	29.8±0.4	35.4±0.58	52.5±0.18	0.51±0.04	0.80±0.03
PAIR	74.9±1.1	27.9±0.9	74.0±7.2	28.8±0.0	35.4±1.30	-	0.47±0.09	-
MABN (ours)	**78.4±0.6**	**38.3±1.2**	**92.4±1.9**	**32.7±0.2**	**36.6±0.41**	**53.2±0.52**	**0.56±0.05**	**0.84±0.04**

WILDS benchmark:

- **Large number of unseen target domains.**
- **Data imbalance at both domain- and class-level.**

Method	clip	info	paint	quick	real	sketch	avg
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
Meta-DMoE	63.5(0.2)	21.4(0.3)	51.3(0.4)	14.3(0.3)	62.3(1.0)	52.4(0.2)	44.2
Ours	**64.2(0.3)**	**23.6(0.4)**	**51.5(0.2)**	**15.2(0.3)**	**64.6(0.5)**	**54.1(0.4)**	**45.5**

Table 6: **Comparison on the DomainNet with std across 3 random seeds.**

Evaluation on Domain-specific Knowledge

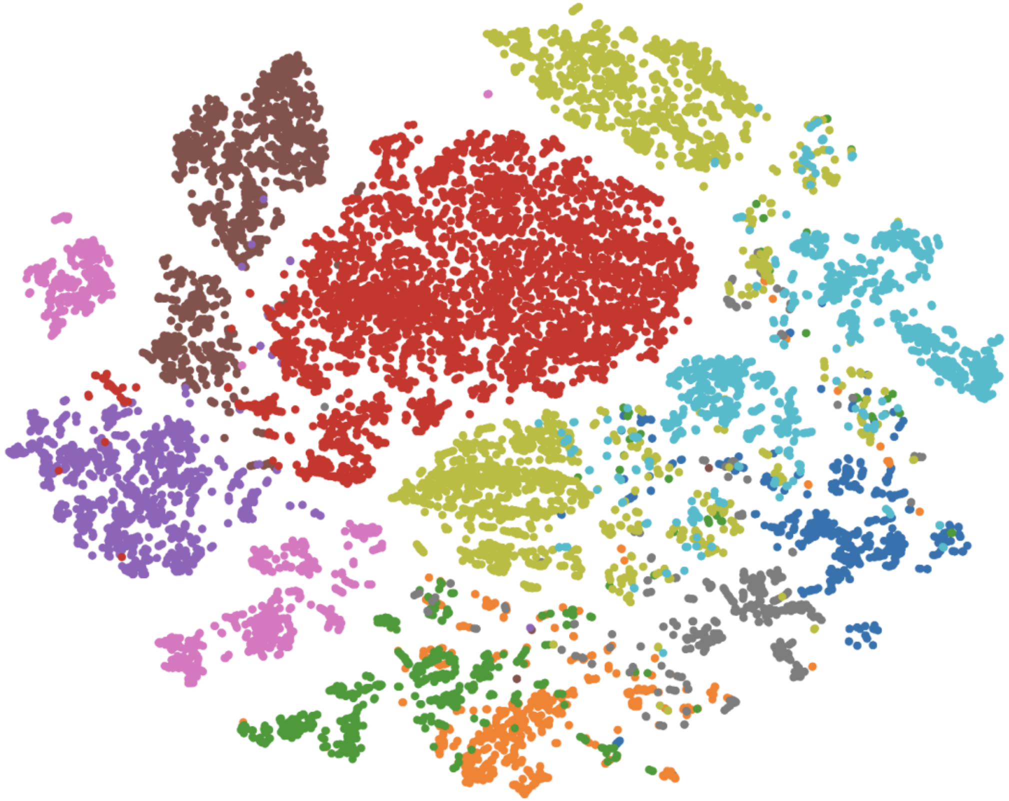
iWildCam benchmark:

- **48 OOD unseen target domains.**

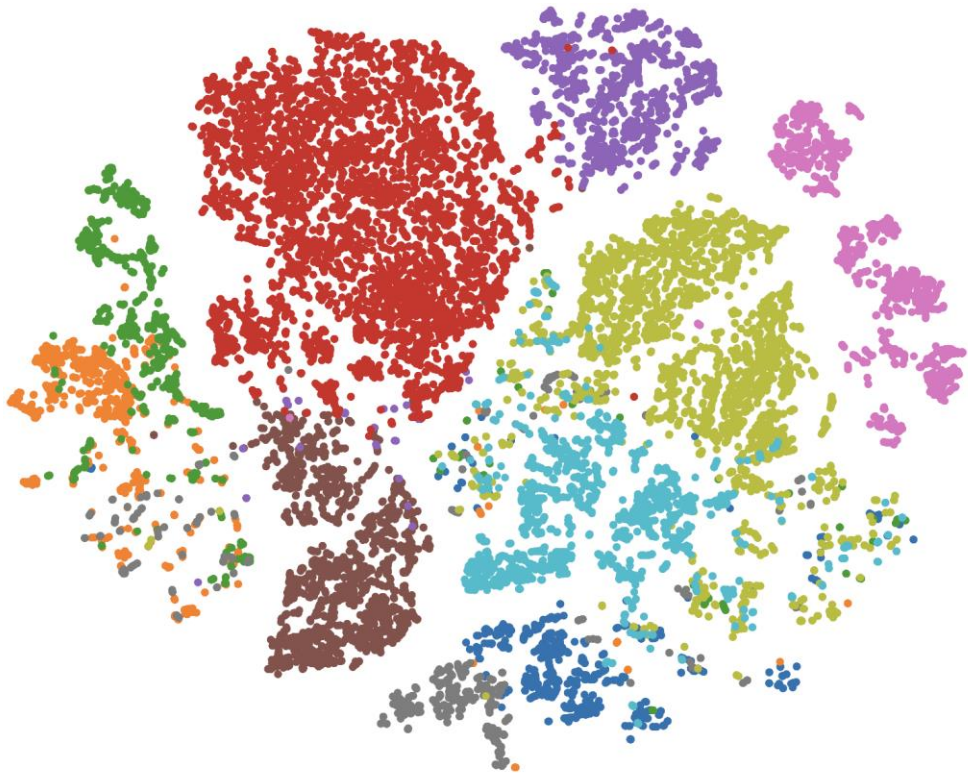
Adapted $(\tilde{\gamma}, \tilde{\beta})$	**No adapt**	**Not-matched**	**Matched**
Accuracy	74.69	72.39	**78.40**
Macro-F1	36.77	33.32	**38.27**

- ❖ Compute adapted Affine Parameters for each domain.
- ❖ Random shuffle them

Visualizations: representation of target domain partial-classes from iWildCam



Left: feature distribution before adaptation



Right: feature distribution after adaptation

Integration with other TTA Method

Method	Update BN		Update Affine	
	Acc	Macro F1	Acc	Macro F1
TENT (min. entropy)	33.27	0.77	75.92	36.40
Our (min. auxiliary)	75.86	36.76	78.40	38.27
Our+TENT	75.84	31.93	79.68	38.85

- ❖ Adapt to domain first.
- ❖ Further improvement with instance-based TTA

Ablation

Index	*SSL*	*Param.*	*TS*	*Adapt*	iWildCam Acc	iWildCam F1
1	✗	All	CE	✗	68.7	31.3
2	✓	All	Joint	✗	70.5	33.2
3	✓	BN	Joint	✓	68.2	30.5
4	✓	Aff	Joint	✓	71.1	33.9
5	✓	All	Meta	✓	72.0	29.4
6	✓	Aff	Meta	✗	74.7	36.8
7	✓	Aff	Meta	✓	78.4	38.3

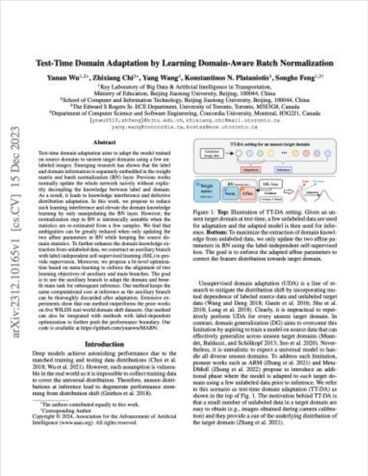
❖ Evaluation on each component.

Self-supervised	Backbone	Training	iWildCam Acc	iWildCam F1
None (baseline)	ResNet50	CE	68.7	31.3
Rotation (Sun et al. 2020)	ResNet50	Joint	69.2	31.5
Rotation (Sun et al. 2020)	ResNet50	Meta	72.8	33.0
MAE (He et al. 2022)	ViT-Base	Joint	71.7	33.8
MAE (He et al. 2022)	ViT-Base	Meta	74.9	35.1
Ours (BYOL)	ResNet50	Joint	70.5	33.2
Ours (BYOL)	ResNet50	Meta	78.4	38.3

❖ Evaluation on SSL methods.

Thanks!

Poster #386


Test-Time Domain Adaptation by Learning Domain-Aware Batch Normalization

arXiv:2312.10165v1 [cs.CV] 15 Dec 2023

Abstract

Introduction

Test-Time Domain Adaptation by Learning Domain-Aware Batch Normalization

Webpage: https://chi-chi-zx.github.io/MABN_project

Code: https://github.com/ynanwu/MABN