DLCV Project Presentation

Paper : FixBi Bridging Domain Spaces for Unsupervised Domain Adaptation *(CVPR 2021)* [1]

Group Members :
1. Parth Bansal (22ucs146)
2. Priyanshu Gupta (22ucs155)
3. Sanidhya Gupta (22ucs182)
4. Shreyansh Agarwal (22ucs203)

Submitted to : Dr. Preety Singh

Introduction

What is Domain Adaptation?

Domain Adaptation is a technique used to help machine learning models perform well on a target domain that differs from the source domain they were originally trained on.

Why it's needed?

In real-world scenarios, data often comes from a variety of sources or environments.These differences can lead to poor model performance when a model trained on one domain is applied directly to another.

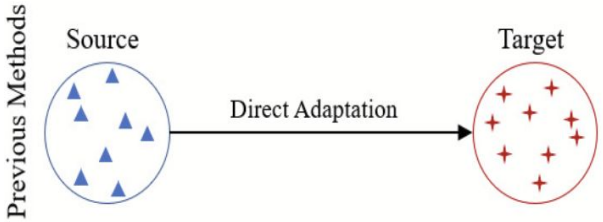
How it helps

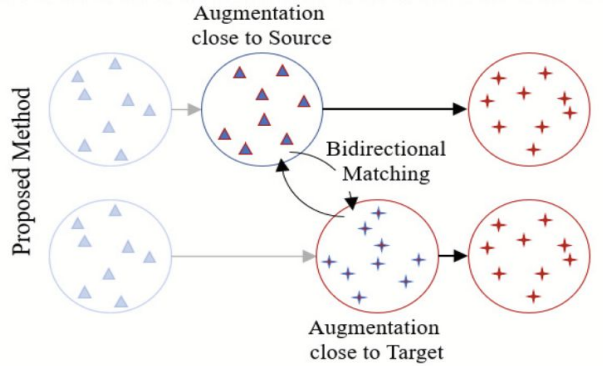
Domain Adaptation addresses the mismatch between source and target domains, allowing models to adapt and generalize better across different data distributions.

Motivation

1) However, most of the studies were based on direct adaptation from the source domain to the target domain and have suffered from large domain discrepancies.
2) In this project, we propose a UDA method that effectively handles such large domain discrepancies.

Goal: Learn a more generalized and robust model for target domain.



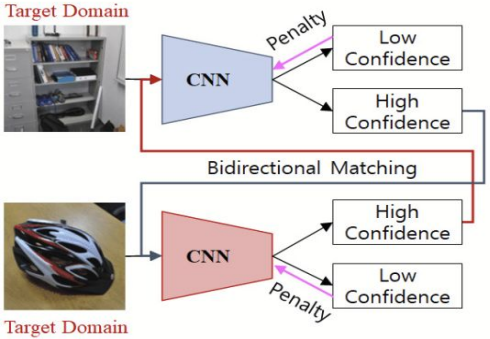


Literature Survey

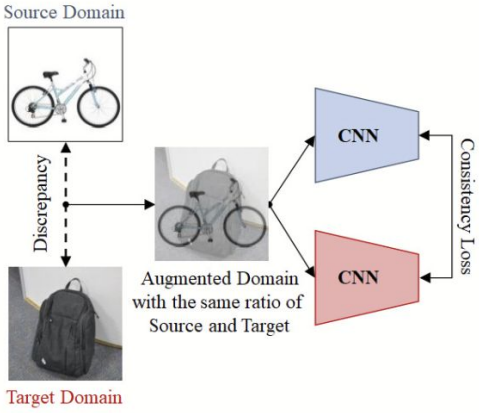
Method	Key Idea	Limitation	Results (Average Accuracy on Office-31)
Domain-Adversarial Training of Neural Networks (DANN) [2]	Uses a **Gradient Reversal Layer (GRL)** to align feature distributions of source and target via adversarial training.	Domain alignment may ignore class boundaries, leading to misclassification.	82.2%
Learning Semantic Representations for Unsupervised Domain Adaptation (MSTN) [6]	Matches moments between domains and introduces residual learning for transfer.	Struggles with complex domain gaps; less robust for high variability.	86.5%
FixBi Bridging Domain Spaces for Unsupervised Domain Adaptation [1]	Introduces a mixup-based intermediate domain, bidirectional matching, and self-penalization for better generalization.	Needs careful tuning of mixup ratios and may add computation overhead.	91.4%

Key Idea of FixBi

- Think of source and target domains as two islands.
- FixBi builds a **bridge** between them using controlled mixup samples.
- **Core Innovations:**
  - **Fixed-ratio mixup** for synthetic intermediate samples
  - **Bi-directional matching** for robust alignment
  - **Self-penalization** to down-weight uncertain pseudo-labels
  - **Consistency regularization** for stable predictions



(b) Confidence-based Learning



(c) Consistency Regularization

Methodology

What is Mixup?

Mixup creates a new *virtual* sample by combining one source sample (xs,ys) and one target sample (xt,yt) using a ratio λ. This helps the model generalize better by learning smoother decision boundaries.

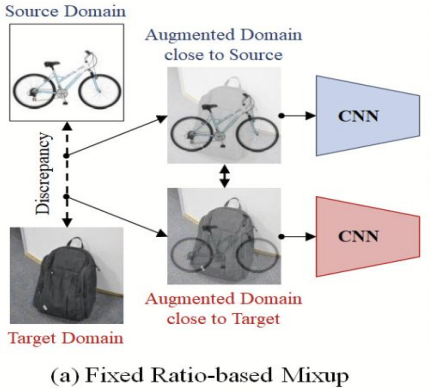
Fixed Ratio Instead of Random

Instead of choosing λ randomly from a Beta distribution (as in standard mixup), FixBi uses **two fixed values**:

- λ_{sd}: source-dominant (more weight on source sample)
- λ_{td} : target-dominant (more weight on target sample)

Formulas for Augmentation (intermediate domain construction)

- Input: $\tilde{x}_{st} = \lambda x_s + (1-\lambda)x_t$
- Label: $\tilde{y}_{st} = \lambda y_s + (1-\lambda)y_t$



(a) Fixed Ratio-based Mixup

x_s: feature from **source** domain

y_s: true label for x_s

x_t: feature from **target** domain

y_t: **pseudo-label** for x_t, predicted by a model (since target is unlabeled)

Purpose of Augmented Samples

The virtual samples lie in between the source and target domains. This helps bridge the domain gap by allowing the model to learn from samples that contain characteristics of **both domains**.

Used in Dual Models (SDM and TDM)

Both models are trained with their respective augmented samples:

- SDM uses $\lambda=\lambda_{sd}$ to focus more on source domain knowledge.
- TDM uses $\lambda=\lambda_{td}$ focus more on adapting to the target domain.

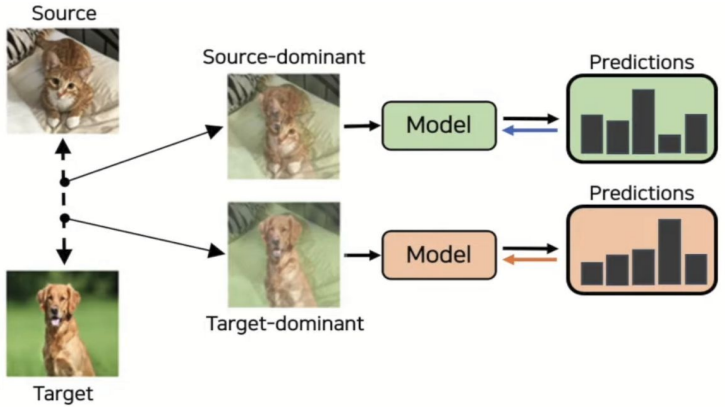
Loss Function (L_{fm})

L_{fm} is used to train SDM and TDM using the mixup-augmented data. It ensures the model's predictions on these virtual samples match the pseudo-labels $\hat{y}_t$

The Formula

$$\mathcal{L}_{fm} = \frac{1}{B}\sum_{i=1}^{B} \hat{y}_t^{(i)} \cdot \log\left(p(y|\tilde{x}_{st}^{(i)})\right)$$

- B is batch size
- $\tilde{x}_{st}$: mixed input sample
- $\hat{y}_t$: pseudo-label for the target sample
- $p(y|\tilde{x}_{st})$: model's predicted class distribution



Why Use Pseudo-labels?

Since the target data is unlabeled, we can't use true labels. So, we use predictions from a model (usually the ensemble or averaged predictions from SDM and TDM) as **pseudo-labels**, assuming they're close enough to the real ones.

Stabilizing Training

Because mixup blends both input and labels, and uses fixed ratios, this loss allows the model to **learn smoother decision boundaries** and become more robust to domain shift, all while being trained in a **controlled and stable way**.

Confidence Based Learning and Self Penalization

To utilize the two models as bridges from the source domain to the target domain, we propose a confidence-based learning where one model teaches the other model using the positive pseudo-labels or teach itself using the negative pseudo-labels.

- **Bidirectional Matching with positive pseudo-labels**. When one network assigns the class probability of input above a certain **threshold τ** , we assume that this predicted label as a **positive pseudo-label**.

$$\mathcal{L}_{bim} = \frac{1}{B}\sum_{i=1}^{B} \mathbb{1}(max(p(y|x_i^t) > \tau)\hat{y}_i^t \log(q(y|x_i^t)),$$

  - **p** and **q** are probability distribution of two models (**source and target dominant**)
  - **B is batch size** .

$$\hat{y}_i^t = \text{argmax}\, p(y|x_i^t)$$

  - Basically , here we train the entire peer network to match it's predictions with the positive pseudo labels via a **standard cross entropy**.

- **Self-penalization with negative pseudo-labels**. The network also learns through negative pseudo labels.
- Here, the negative pseudo-label indicates the most confident label (top-1 label) predicted by the network with a confidence lower than the **threshold τ**
- Since the negative pseudo-label is unlikely to be a correct label, we need to increase the probability values of all other classes except for this negative pseudo-label.

$$\mathcal{L}_{sp} = \frac{1}{B}\sum_{i=1}^{B} \mathbb{1}(max(p(y|x_i^t) < \tau)\hat{y}_i^t \log(1 - p(y|x_i^t))$$

  - The hyperparameter T is not fixed and keeps on changing adaptively by sample mean and standard deviation of the mini batch.

Consistency Regularization

- Encourage **stable convergence** of SDM and TDM.
- Ensure both models produce **consistent predictions** on intermediate (mixed) inputs.
- **Improves generalization** to target domain.

How It Works?

- Mix source and target inputs equally to form **intermediate samples** using fixed mixup ratio λ_{sd} and λ_{td} set to 0.5
- Both SDM and TDM predict on these shared inputs.
- Apply **L2 consistency loss** to match predictions:

$$\mathcal{L}_{cr} = \frac{1}{B} \sum_{i=1}^{B} \left\| p(y|\tilde{x}_{st}^{i}) - q(y|\tilde{x}_{st}^{i}) \right\|_2^2$$

- B: batch size
- p, q: predictions from SDM and TDM
- $\tilde{x}_{st}$: mixed input

The loss measures **how different the two predictions are** — we want this to be **small.**

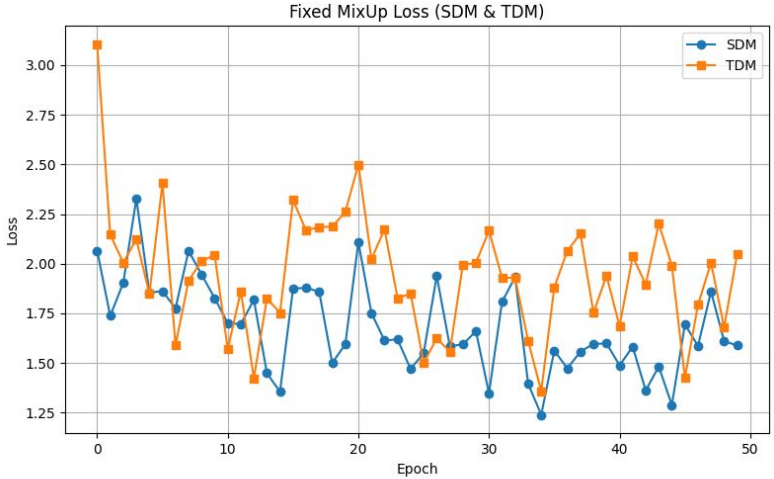
Results

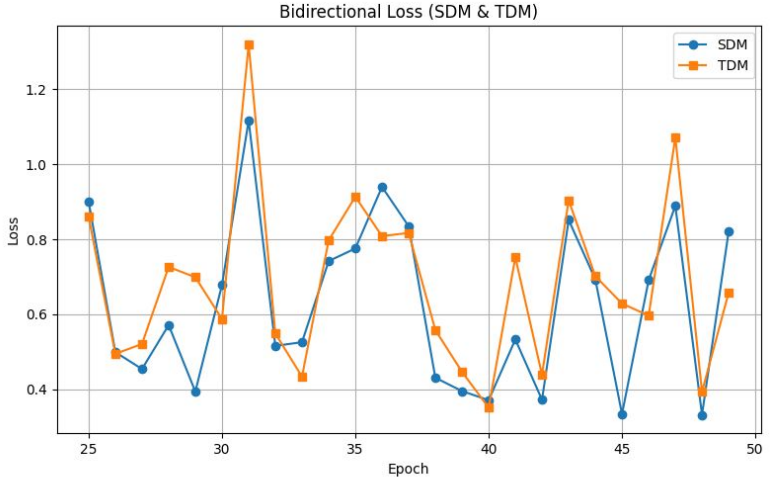
Office-31 Dataset

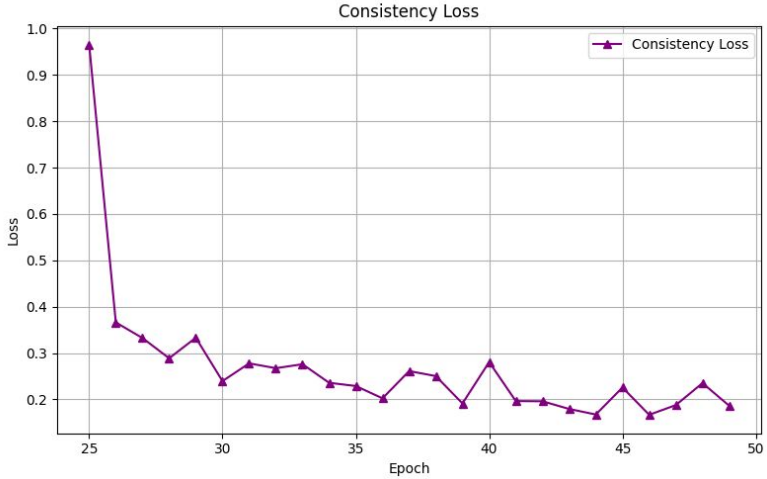
- **Dataset Used:** Office-31 (31 classes) [3] (subdivided into Amazom, DSLR, Webcam)
  - Source Domain : Office-31 Amazon
  - Target Domain : Office-31 DSLR
- **Base Model:** ResNet-50 (initialized with **DANN**[5] based training weights)
- **Results on Office-31** (A: Amazon, D: DSLR, W: WebCam) in Paper [1]

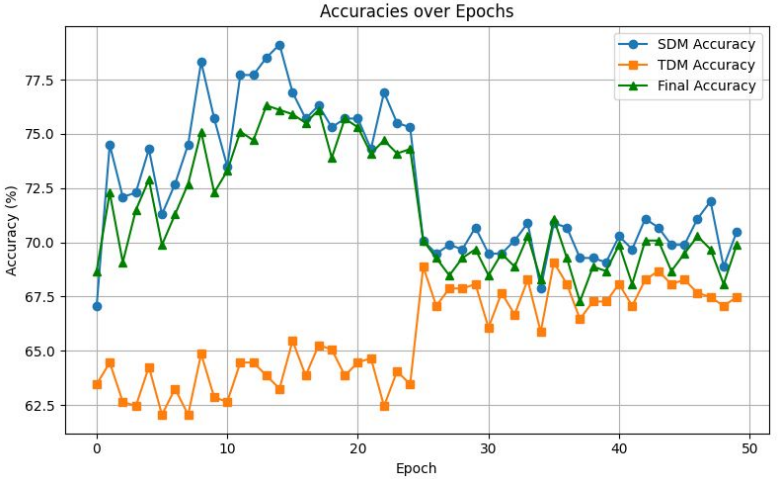
Method	A→W	D→W	W→D	A→D	D→A	W→A	Avg
FixBi (Ours)	**96.1±0.2**	**99.3±0.2**	**100.0±0.0**	95.0±0.4	**78.7±0.5**	**79.4±0.3**	**91.4**

- **Our Achieved Accuracy** (after 50 Epochs and warm-up: 25, as stated in experiments part of paper)
  - SDM: 70.48%, TDM: 67.47%
  - Together: 69.88%
- **Observations**:
  - Improved class-wise prediction consistency in SDM and TDM
  - Difference in paper stated results and our results is huge, maybe **due to different weights of DANN**[2] and **way smaller batch size of 8 (as compared to 32)** due to memory issue in Google Colab.
  - This issue was also observed by other users also, see: https://github.com/NaJaeMin92/FixBi/issues/12









Note: 1st epoch is marked as 0 in graphs

References

[1] J. Na, H. Jung, H. J. Chang and W. Hwang, "**FixBi: Bridging Domain Spaces for Unsupervised Domain Adaptation,**" 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2021, pp. 1094-1103, doi: 10.1109/CVPR46437.2021.00115.

[2] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. "**Domain-adversarial training of neural networks.**" J. Mach. Learn. Res. 17, 1 (January 2016), 2096–2030.

[3] **Office-31 Dataset** : https://paperswithcode.com/dataset/office-31

[4] **FixBi Code GitHub Repository** : https://github.com/NaJaeMin92/FixBi

[5] **DANN Implementation and Weights** : https://github.com/NaJaeMin92/pytorch-DANN

[6] S. Xie, Z. Zheng, L. Chen, and C. Chen. "**Learning semantic representations for unsupervised domain adaptation.**" In Thirty-fifth International Conference on Machine Learning (ICML), 2018