Continual Unsupervised Domain Adaptation

A Project Report Submitted by

Divye Dixit

in partial fulfilment of the requirements for the award of the degree of

Computer Science and Engineering

Indian Institute of Technology Mandi

SCEE

May, 2025

Declaration

I hereby declare that the work presented in this Project Report titled Continual Unsupervised Domain Adaptation submitted to the Indian Institute of Technology Mandi in partial fulfilment of the requirements for the award of the degree of Computer Science and Engineering, is a bonafide record of the research work carried out under the supervision of Parimala Kancharla. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.

Signature

Divye Dixit

B21045

Certificate

This is to certify that the Project Report titled Title of the Project Report, submitted by Divye Dixit (B21045) to the Indian Institute of Technology Mandi for the award of the degree of Computer Science and Engineering, is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Signature

Parimala Kancharla

Acknowledgements

This project would not have been possible without the invaluable guidance and support of several individuals. First and foremost, we would like to express our heartfelt gratitude to our project advisor, Parimala Kancharla, for their constant encouragement, insightful feedback, and unwavering support throughout the course of this research. Their expertise and mentorship have been instrumental in shaping the direction and success of this work.

We are also deeply grateful to the faculty and staff at SCEE department of IIT Mandi for providing a conducive environment for research and for offering their assistance whenever required.

Additionally, we extend our appreciation to our friends and families, whose encouragement and understanding have been a source of motivation during challenging times. Finally, we acknowledge the resources provided by IIT Mandi, without which this work would not have been feasible.

To all who have supported us in this journey, we offer our sincerest thanks.

Abstract

Continual Unsupervised Domain Adaptation (CUDA) is a challenging problem that integrates aspects of domain adaptation with continual learning in an unsupervised setup. This study aims to deal with the problem of domain shift wherein there is a labeled source domain and several unlabeled target domains with different distributions of data. We attempt to tackle two main problems of the setup, unsupervised adaptation on the unlabeled domains and the forgetting happening on previously adapted domains while the unsupervised shift. Traditional Unsupervised Domain Adaptation (UDA) tend to lose specific valuable information from the attained domain while enforcing generalization to other domains, therefore not suitable for a continual setup (Continual Domain Adaptation - CDA).

Such prospects encourage this work to present three novel approaches, both of which use the Mean Teacher framework for the unsupervised adaptation and different techniques for mitigating catastrophic forgetting in CDA. The first approach uses feature-level replay using a diffusion model to mitigate forgetting. The second approach creates Expert Modules for each domain by training and using a LoRA adapter to adapt the base model trained on the source domain (labeled). The third approach achieves State of the art results on a low compute budget and utilizes components of SHOT, Fixmatch, LoRA adaptors and Diffusion Models

Experimental results on benchmark dataset demonstrate the effectiveness of the frameworks achieving the best results recorded yet in cross-domain deployment with lower run-time cost.

Contents

Continual Unsupervised Domain Adaptation

1 Introduction and background

The problem for the MTP project was chosen to be Continual Unsupervised Domain Adaptation. To understand this problem, we first have to understand the individual problems which are domain adaptation, continual domain adaptation, and unsupervised domain adaptation.

1.1 Domain Adaptation

Domain adaptation is a problem in machine learning where a model trained on one dataset (the source domain) is applied to a different but related dataset (the target domain). Since the two datasets are from different distributions, there is said to be a domain shift between the datasets due to which a machine learning model trained on the source dataset is not able to perform well on the target dataset. Taking the example of a computer vision dataset, a few practical reasons for a domain shift can be:

1. Changes in lighting condition between the source and target domain
2. Change in camera quality
3. Weather conditions are different such as fog, rain, etc.
4. Object visibility, different viewing angles, etc.

The main objective of the domain adaptation problem is to transfer knowledge from the source domain to the target domain while minimizing the need for labeling the data in the target domain. Some methods for dealing with domain shift include learning features from the source and target datasets and trying to bring them closer so that the model performs better on the target domain.

Another problem with domain adaptation problems is that of catastrophic forgetting wherein, when a model adapts to a new domain, it is not able to perform well on the source dataset as the weights are now tuned for the target dataset.

In summary, the major problems in domain adaptation are:

1. **Distribution Shift:** Differences in data distributions make it difficult for a model trained on one domain to generalize to another.
2. **Label Scarcity in the Target Domain:** Labels are often unavailable or expensive to obtain in the target domain, so methods must adapt using primarily unlabeled data.
3. **Catastrophic Forgetting:** When adapting to new domains, models might forget knowledge learned from the source domain.

1.2 Unsupervised Domain Adaptation

Unsupervised Domain Adaptation (UDA) is a specific type of domain adaptation problem where we aim to transfer knowledge from a labeled source domain to an unlabeled target domain. The main features between the two domains remain the same, so we have to leverage the labeled source domain to train a model which performs on the unlabeled target domain.

The lack of labeled data in the target domain makes UDA particularly challenging, as most supervised learning algorithms rely on labels to guide model learning. The primary goal of UDA is to reduce the distribution gap between the source and target domains, so the model trained on the source domain data can generalize effectively to the target domain. Common applications of UDA include tasks in computer vision, natural language processing, and healthcare, where obtaining labeled data in every environment or for every dataset variant is often impractical due to cost or privacy concerns.

In summary, the problems with unsupervised domain adaptation are:

1. **Domain Discrepancy:** Differences in data distributions between source and target domains can degrade target performance (e.g., a model trained on daytime images may struggle with nighttime scenes).
2. **Absence of Target Labels:** Lacking target domain labels, models rely on indirect methods to adapt, like aligning source and target feature distributions.
3. **Feature Alignment:** UDA models must learn domain-invariant features to bridge the source-target gap, often using techniques like adversarial training or distribution matching.
4. **Avoiding Source Overfitting:** With labeled data only in the source domain, models risk overfitting to source-specific features, harming target domain performance.

1.3 Continual Domain Adaptation

Continual Domain Adaptation (CDA) extends the concept of domain adaptation to scenarios where a model must adapt to multiple new target domains in sequence, each with its own unique distribution. Unlike standard domain adaptation, which focuses on adapting from one source to one target domain, CDA handles a continuous stream of evolving domains without retraining from scratch. In each new domain, the model must adapt while retaining the knowledge gained from previous domains.

The main issue with this type of domain adaptation problem is that of **catastrophic forgetting**, wherein the model stops performing well on the source dataset or any of the previously seen datasets upon being exposed to a new dataset.

2 Literature survey

We sought out to solve the Continuous Unsupervised Domain Adaptation using diffusion models. There was not much literature or any available papers which used diffusion to solve problems in this domain, so we had to look into different architectures for inspiration. The idea was therefore to pick up two state-of-the-art models in the following domain:

1. Unsupervised Domain Adaptation - Unsupervised Domain Adaptation via Domain-Adaptive Diffusion [1] - DAD Module
2. Continual Domain Adaptation - Continual Domain Adaptation through Pruning-aided Domain-specific Weight Modulation [2] - PaCDA

2.1 Unsupervised Domain Adaptation via Domain-Adaptive Diffusion

2.1.1 Problem and Motivation

When trained on extensive labeled datasets, Deep Neural Networks (DNNs) have demonstrated impressive performance in a variety of visual identification tasks. However, when used in domains with radically different distributions, their performance frequently deteriorates—a phenomenon known as domain shift. By transferring information from a labeled source domain to an unlabeled target domain, Unsupervised Domain Adaptation (UDA) aims to close this domain gap. Direct adaptation is challenging due to the significant distribution gap between the two domains.

Motivated by Denoising Diffusion Probabilistic Models (DDPMs), which are excellent at progressively changing distributions, the authors suggest a new framework that makes use of diffusion models' progressive distribution conversion capabilities. The suggested approach makes adaptation easier and more efficient by breaking down the domain gap into smaller, more manageable steps.

2.1.2 Proposed Framework

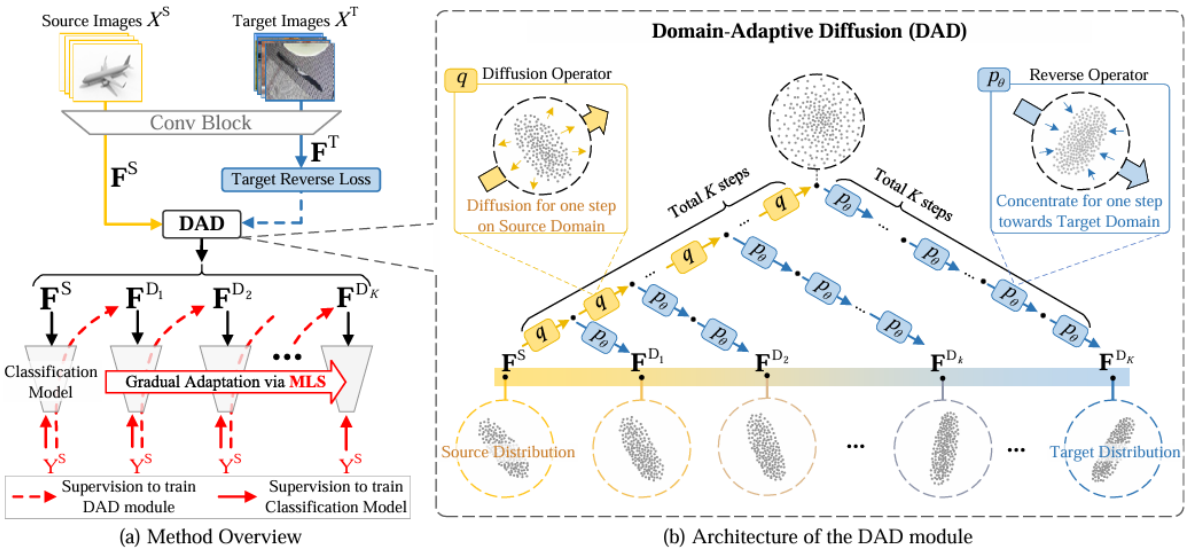

Figure 2.1: Step-by-step diffusion and MLS stratergy

The proposed framework consists of two main components: the Domain-Adaptive Diffusion (DAD) module and the Mutual Learning Strategy (MLS).

Domain-Adaptive Diffusion (DAD) Module The DAD module uses a combination of a diffusion operator q and a reverse operator p_θ to simulate a progressive distribution transition from the source domain to the target domain.

The diffusion operator q perturbs the source feature distribution F_S over K steps:

$$q(F_k|F_{k-1}) = \mathcal{N}(F_k; \sqrt{1-\beta_k}F_{k-1}, \beta_k I),$$

where β_k controls the variance of the noise added at each step, and F_{k-1} represents the feature map after $(k-1)$ diffusion steps.

The reverse operator p_θ eliminates perturbations to concentrate the distribution back toward the target domain:

$$p_\theta(F_{k-1}|F_k) = \mathcal{N}(F_{k-1}; \mu_\theta(F_k, k), \beta_k I),$$

where $\mu_\theta(F_k, k)$ is the mean of the output feature distribution:

$$\mu_\theta(F_k, k) = \frac{1}{\sqrt{\alpha_k}}\left(F_k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}} f_\theta(F_k, k)\right),$$

and $\alpha_k = 1 - \beta_k$, $\bar{\alpha}_k = \prod_{s=1}^{k} \alpha_s$, and f_θ is a learnable neural network within the reverse operator.

The DAD module iteratively applies q and p_θ to simulate the cross-domain distribution transition process:

$$F_{D_k} = p_\theta(q(F_S, k), k),$$

where F_{D_k} represents the feature map at the k-th transitional distribution.

Mutual Learning Strategy (MLS) To preserve feature semantics and enhance adaptation, the MLS alternates learning between the classification model and the DAD module.

1. **Classification-to-DAD (C→D) Learning**: The classification model, trained on prior distributions, guides the DAD module to preserve semantics in the next transitional distribution. The loss function is:

$$\mathcal{L}_{C\to D} = \mathcal{L}_{cls}(f_C(F_{D_k}), Y_S) + \mathcal{L}_{TR},$$

where $\mathcal{L}_{cls}$ is the cross-entropy loss between the predicted and ground-truth labels, and $\mathcal{L}_{TR}$ is the target reverse loss:

$$\mathcal{L}_{TR} = \mathbb{E}_q\left[-\log p_\theta(F_k) - \sum_{k=1}^{K} \log \frac{p_\theta(F_{k-1}|F_k)}{q(F_k|F_{k-1})}\right].$$

2. **DAD-to-Classification (D→C) Learning**: The updated DAD module generates transitional features to train the classification model on new distributions:

$$\mathcal{L}_{D\to C} = \frac{1}{k}\sum_{i=0}^{k-1} \mathcal{L}_{cls}(f_C(F_{D_i}), Y_S) + \mathcal{L}_{cls}(f_C(F_{D_k}), Y_S).$$

Through alternating C→D and D→C learning, the classification model progressively adapts to the target domain.

2.1.3 Training and Inference

During training, the backbone network (e.g., ResNet-50) is initialized with ImageNet-pretrained parameters and trained on the source domain using the source training loss:

$$\mathcal{L}_{ST} = \mathcal{L}_{cls}(f_C(f_E(X_S)), Y_S),$$

where f_E is the feature extractor and f_C is the classification head.

After training, the Conv Block is frozen, and the DAD module is trained with the target reverse loss $\mathcal{L}_{TR}$. The MLS is then employed to alternately train the DAD module and classification model.

During inference, the DAD module is removed, and the Conv Block and classification model are used to predict labels for target domain data.

2.1.4 Results and Analysis

The proposed method achieves state-of-the-art performance on benchmarks like Office-31, Office-Home, and VisDA-2017. Ablation studies demonstrate the effectiveness of the multi-step transition design in DAD and the MLS for semantic preservation. The training introduces minimal computational overhead, while inference efficiency is on par with other methods.

2.2 Generative Feature-driven Image Replay for Continual Learning

2.2.1 Problem and Motivation

Continual learning (CL) systems often suffer from *catastrophic forgetting*, where neural networks lose previously acquired knowledge upon learning new tasks [3]. Traditional methods mitigate this by storing and replaying past data, which may not be feasible due to privacy concerns or storage limitations. Generative replay approaches attempt to address this by synthesizing past data; however, existing methods either replay images—requiring complex generative models—or replay features—limiting the adaptability of the entire classifier. There is a need for a method that combines the advantages of both approaches to effectively prevent forgetting without storing real data.

2.2.2 Proposed Framework

The authors introduce **Genifer** (GENeratIve FEature-driven image Replay), a novel framework that synergizes image and feature replay. Genifer employs a conditional Generative Adversarial Network (GAN) [4] to synthesize images that, when passed through the classifier's feature extractor, produce feature representations closely matching those of real data. This approach ensures that the generated images are tailored to the classifier's learned features, facilitating effective knowledge retention. The key components of Genifer include:

- **Feature-driven Generation**: The GAN is trained not to replicate the exact image distribution but to generate images that elicit similar feature activations in the classifier as real images.
- **Adversarial Distillation**: A combination of adversarial loss and image distillation loss is used to prevent forgetting in the generative model itself, ensuring the quality and relevance of generated samples over time.
- **Data Augmentation**: Leveraging image-space augmentations enhances the diversity of generated samples, improving the robustness of the classifier and mitigating overfitting.

2.2.3 Training and Inference

Training in Genifer alternates between updating the classifier and the generative model:

1. **Classifier Training**: The classifier is trained on both real data from the current task and generated data representing previous tasks. Feature distillation is applied to preserve knowledge from earlier tasks.
2. **Generator Training**: The generator is updated to produce images that, when processed by the classifier, yield feature representations similar to those from previous tasks. Adversarial distillation ensures that the generator maintains its ability to produce relevant samples over time.

This alternating training scheme allows both the classifier and the generator to adapt continually, preserving past knowledge while integrating new information.

2.2.4 Results and Analysis

Genifer was evaluated on two benchmark datasets: CIFAR-100 [5] and CUB-200 [6], under various class-incremental learning settings. The results demonstrate that Genifer outperforms existing exemplar-free methods and even surpasses exemplar-based approaches in certain scenarios. Key findings include:

- **Superior Performance**: Genifer achieved higher average incremental accuracy compared to state-of-the-art methods like Fusion and iCaRL [7], particularly in settings with a large number of tasks.

- **Effective Knowledge Retention**: The combination of feature-driven generation and adversarial distillation effectively mitigates catastrophic forgetting, maintaining classifier performance over extended task sequences.
- **Scalability**: Genifer's approach scales well to complex datasets without the need to store real exemplars, making it suitable for real-world applications where data storage is constrained.

These results highlight Genifer's efficacy in continual learning scenarios, offering a promising direction for developing robust, scalable CL systems without relying on stored exemplars.

2.3 Transformer^2: Self-Adaptive Large Language Models with Expert Vectors and a Two-Pass Mechanism

2.3.1 Problem and Motivation

Traditional parameter-efficient fine-tuning (PEFT) methods such as LoRA require a separate low-rank adapter for every downstream task, quickly inflating storage and hindering compositionality. *Transformer*^{2}, proposed by Sakana AI in 2025, instead decomposes each weight matrix of a pre-trained LLM via singular-value decomposition (SVD) and learns a compact *z-vector* that rescales only the singular values relevant to a given task [8]. Each z-vector functions as a lightweight **expert** (typically $<0.1\%$ of model parameters), enabling the base model to acquire new skills without retraining or catastrophic interference.

2.3.2 Proposed Framework

The framework has two distinct stages:

(1) Expert Vector Generation (Offline)

- **Singular-Value Fine-Tuning (SVF).** For every transformer weight matrix W, an RL policy learns a task-specific scaling vector $z \in \mathbb{R}^{\text{rank}(W)}$. The adapted weight becomes $W' = U\,\text{diag}(\sigma \odot z)\,V^\top$, where $U\Sigma V^\top$ is the SVD of W and $\odot$ denotes element-wise multiplication.
- **Reinforcement Learning Objective.** SVF maximizes task reward directly (e.g. GSM8K accuracy), promoting strong specialization while keeping parameter cost minimal.

(2) Two-Pass Inference (Online) At test time the model adapts *on the fly* via a lightweight dispatch–adapt pipeline:

1. **First pass — Task Detection.** The base LLM runs once to analyze the prompt and obtain task cues. Three interchangeable dispatchers are proposed: *prompt-based*, *classifier-based*, and *few-shot optimization*.
2. **Second pass — Expert Composition.** The dispatcher selects or linearly mixes the relevant z-vectors, producing a composite vector z^*. All weight matrices are then modulated by z^*, and the final answer is generated.

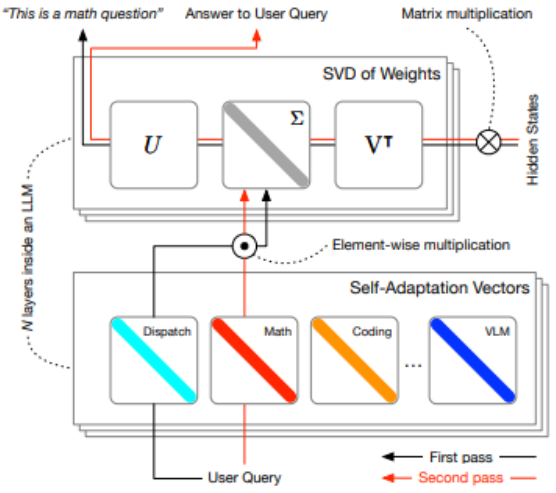

Figure 2.2: Transformer two-pass approach

2.3.3 Training and Inference

During training only the z-vectors (and, optionally, a lightweight task classifier) are updated; the base LLM remains frozen, keeping compute economical. Inference cost is a single extra forward pass for task detection, after which the model runs normally with the adapted weights. Because z-vectors are broadcast to every layer, adaptation remains *full-rank* yet parameter-efficient.

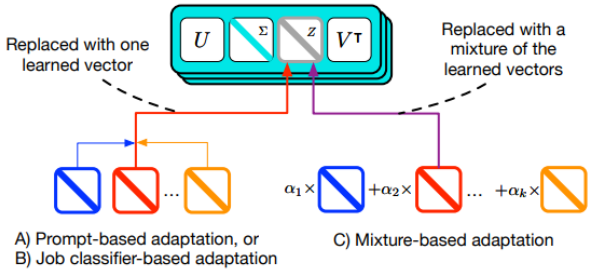

Figure 2.3: Singular Value Fine Tuning

2.3.4 Results and Analysis

Across Llama-3-8B and Mistral-7B backbones, Transformer^2's SVF experts outperform LoRA by 1.5–3.2 pp on GSM8K, MBPP-Pro, and ARC-Challenge while using $\sim$10$\times$ fewer trainable parameters [8, 9]. On unseen tasks (e.g. MATH, HumanEval), the two-pass strategy yields further gains of +2 pp (prompt-based), +4 pp (classifier), and up to +7 pp with the few-shot mixer. Notably, learned experts transfer across architectures—z-vectors trained on Llama-3 improve Mistral-7B by 0.8–1.4 pp without retraining, suggesting strong modularity.

2.4 Low-Rank Adaptation (LoRA)

2.4.1 Problem and Motivation

Large pre-trained transformers are costly to fine-tune when every task requires duplicating all parameters. **LoRA** [10] tackles this by *freezing* the backbone and learning a low-rank update

$$\mathbf{W}' = \mathbf{W}_0 + \Delta\mathbf{W}, \qquad \Delta\mathbf{W} = \mathbf{B}\mathbf{A},$$

where $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$, $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$, and $r \ll \min(d_{\text{in}}, d_{\text{out}})$. A scaling factor α/r controls the update magnitude. The approach reduces trainable parameters by orders of magnitude while *adding zero inference latency* because $\mathbf{W}'$ is merged into the forward pass.

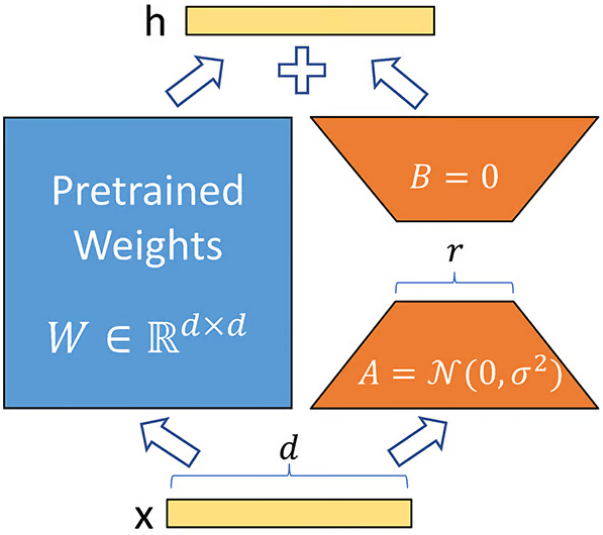

Figure 2.4: Low-Rank Adaptation

2.4.2 Proposed Framework

Adapter Injection. Low-rank pairs $(\mathbf{A}, \mathbf{B})$ are inserted in parallel with each linear projection in attention and MLP blocks. During back-propagation only these pairs are updated; the frozen backbone supplies stable features.

Variants.

- **QLoRA** [11]: couples 4-bit NF4 quantisation with LoRA, enabling 65B models on a single GPU.
- **AdaLoRA** [12]: allocates rank dynamically via singular-value importance scores.
- **Serial LoRA** [13]: cascades multiple rank-1 adapters for ViTs, matching full fine-tuning on ImageNet.
- **DualLoRA** [14] and **C-LoRA** [15]: introduce orthogonal or routed adapters for continual learning.
- **ExPLoRA** [16] and **DC-LoRA** [17]: extend LoRA to domain-shift and domain-incremental settings.

2.4.3 Training and Inference

Training updates only $(\mathbf{A}, \mathbf{B})$ with standard optimizers; the memory footprint is $\mathcal{O}(r(d_{\text{in}} + d_{\text{out}}))$. At inference the adapters are fused into $\mathbf{W}_0$, so speed equals that of the frozen backbone.

2.4.4 Results and Analysis

- **Parameter Efficiency**: On GPT-3 175B, LoRA tunes just 0.1% of parameters while matching full fine-tuning [10].
- **Memory Savings**: QLoRA fits 65B models in 24 GB VRAM with no accuracy loss [11].
- **Vision Benchmarks**: Serial LoRA achieves 84.7% top-1 on ImageNet with only 0.22 M trainable parameters [13].
- **Continual Learning**: DualLoRA retains +6

2.5 FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence

2.5.1 Problem and Motivation

State-of-the-art semi-supervised learning (SSL) pipelines such as MixMatch and ReMixMatch combine several heuristics—mixup, sharpening, auxiliary rotation losses, distribution alignment, etc.—making them hard to tune and slow to train. **FixMatch** removes this complexity by coupling only two core ideas: *consistency regularisation* and *pseudo-labelling*. With just a confidence threshold and two levels of data augmentation, it attains or surpasses previous SOTA on CIFAR-10, CIFAR-100, SVHN and STL-10 while training fewer epochs and using $\sim 7\times$ less compute:contentReference[oaicite:0]index=0.

2.5.2 Proposed Framework

Notation. Let $\mathcal{D}_{\text{lab}} = \{(x_i, y_i)\}_{i=1}^{n}$ be labelled data and $\mathcal{D}_{\text{unlab}} = \{u_j\}_{j=1}^{m}$ be unlabeled data. FixMatch maintains a single neural network f_θ.

Step 1: Weak Pseudo-Label Generation. Each $u \in \mathcal{D}_{\text{unlab}}$ is first passed through a *weak* augmentation $\mathcal{A}_w$ (random flip + translation). The network prediction is $p_w = f_\theta(\mathcal{A}_w(u))$. If $\max(p_w) \geq \tau$ ($\tau = 0.95$ works best) the hard pseudo-label is $\hat{y} = \arg\max(p_w)$; otherwise the sample is discarded:contentReference[oaicite:1]index=1.

Step 2: Consistency on Strong Augmentations. The *same* image undergoes a *strong* augmentation $\mathcal{A}_s$ (RandAugment or CTAugment) and the network is required to predict $\hat{y}$:

$$\mathcal{L}_{\text{u}} = \frac{1}{|\mathcal{B}_{\text{conf}}|} \sum_{u \in \mathcal{B}_{\text{conf}}} \ell_{\text{CE}}\big(f_\theta(\mathcal{A}_s(u)), \hat{y}\big),$$

where $\mathcal{B}_{\text{conf}} \subseteq \mathcal{D}_{\text{unlab}}$ are confident samples.

Step 3: Supervised Cross-Entropy. Labelled data use the usual loss $\mathcal{L}_{\text{sup}} = \frac{1}{|\mathcal{B}_{\text{lab}}|} \sum_{(x,y) \in \mathcal{B}_{\text{lab}}} \ell_{\text{CE}}(f_\theta(\mathcal{A}_w(x)), y)$.

Total Objective. $\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda \mathcal{L}_{\text{u}}$ with $\lambda = 1$ by default. No additional auxiliary objectives are required:contentReference[oaicite:2]index=2.

2.5.3 Training and Inference

Training alternates mini-batches of labelled and unlabeled data; the network is updated with SGD-Nesterov. A single model is kept—no EMA teacher is necessary, simplifying memory and code. At inference the augmentation pipeline is disabled, yielding standard forward latency.

2.5.4 Results and Analysis

- **CIFAR-10**: 94.93% accuracy with only 250 labelled images (4 per class):contentReference[oaicite:3]index=3.
- **CIFAR-100**: 71.71% with 10
- **SVHN**: 96.87% using 1,000 labels:contentReference[oaicite:5]index=5.
- **Computational Cost**: Runs for 300 epochs on a single GPU; ablations show performance is robust to $\tau \in [0.9, 0.99]$ and $\lambda \in [0.5, 2]$:contentReference[oaicite:6]index=6.

2.6 Exponential Moving Average (EMA) Teacher

2.6.1 Problem and Motivation

Deep networks trained on unlabeled data via consistency regularisation (e.g. Π-model, Temporal Ensembling) suffer from noisy targets and stale updates, especially on large datasets. *Mean Teacher* instead maintains an **EMA** of model weights as a "teacher," providing smoother, more accurate targets for the student to match at every step:contentReference[oaicite:0]index=0. This yields better intermediate representations and faster convergence compared to Temporal Ensembling, which updates targets only once per epoch:contentReference[oaicite:1]index=1.

2.6.2 Proposed Framework

Student–Teacher Setup. Let θ_t be the student parameters at step t and θ'_t the teacher parameters. After each gradient update on the student, the teacher is updated via:

$$\theta'_t = \alpha\,\theta'_{t-1} + (1-\alpha)\,\theta_t,$$

where $\alpha \in [0,1)$ (e.g. 0.99–0.999) controls the smoothing rate:contentReference[oaicite:2]index=2.

Consistency Loss. For each input x (labelled or unlabelled) with two augmentations η, η',

$$\mathcal{L}_{\text{cons}} = \mathbb{E}_{x,\eta,\eta'} \left\| f_\theta(x;\eta) - f_{\theta'}(x;\eta') \right\|^2,$$

encouraging the student's prediction f_θ to match the teacher's target $f_{\theta'}$:contentReference[oaicite:3]index=3.

Combined Objective. On labelled data (x, y) add cross-entropy,

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda\,\mathcal{L}_{\text{cons}},$$

with λ tuned per dataset (e.g. $\lambda = 1$ for CIFAR/SVHN):contentReference[oaicite:4]index=4.

2.6.3 Training and Inference

- **Online update.** Teacher weights are updated every step, avoiding the epoch-level staleness of Temporal Ensembling:contentReference[oaicite:5]index=5.
- **No separate teacher forward.** Both student and teacher compute one forward pass each, incurring minimal overhead.
- **Inference.** Use the *teacher* model θ' for final predictions for better generalisation; no consistency term is applied.

2.6.4 Results and Analysis

- **SVHN (250 labels):** Error reduced to 4.35%, beating Temporal Ensembling's 6.28%:contentReference[oaicite:6]inde
- **CIFAR-10 (4k labels):** Achieves 6.28%, a +4% improvement over prior methods:contentReference[oaicite:7]index=
- **ImageNet-2012 (10% labels):** Top-5 error drops to 9.11%, compared to 35.24% with only supervised training:contentReference[oaicite:8]index=8.
- **Scaling.** Mean Teacher scales to large datasets and modern architectures (ResNet, ViT) without modification, and is the backbone of many recent SSL/UDA pipelines:contentReference[oaicite:9]index=9.
- **Variants.** Robust-MT for medical imaging:contentReference[oaicite:10]index=10, two-stage teacher schedules for vision transformers:contentReference[oaicite:11]index=11.

3 Problem Definition and Objective

3.1 Problem Definition

The central aim of this study is to address the challenge of Continual Unsupervised Domain Adaptation (CUDA) by synthesizing components of domain adaptation, unsupervised learning, and continual learning. The CUDA relies on the machine learning model, thus allowing the transition from the labeled source domain to the series of target domains unlabeled using diverse data distributions.

This issue derives difficulties from the foundational domains:

1. **Domain Discrepancy**: In the case of applying a model trained in one domain to another, significant distribution differences between the source and target domains cause a lack of generalization.
2. **Label Scarcity**: Target domains lack sufficient labeled data, making supervised learning unfeasible. Models must leverage the labeled source domain and obtain information indirectly through distribution alignment approaches.
3. **Catastrophic Forgetting**: As the model adjusts to new target domains, it typically forgets knowledge from domains that it had learned earlier, causing a loss of performance in those earlier domains.

Conventional approaches to UDA address differences between domains, yet often fail in sequential settings. On the other hand, strategies for CDA retain prior learning but, in general, lack proper approaches to reconcile source and target distributions. The gap signifies the lack of an integrated solution.

3.2 Objective

This project aims to design an innovative framework with which to tackle the challenge of **Continual Unsupervised Domain Adaptation (CUDA)** by synergistically combining the two leading methodologies within that domain.

1. **Domain-Adaptive Diffusion (DAD):** Utilizes diffusion models to systematically reduce the gap between domains by decomposing domain shifts into feasible intermediary stages.
2. **Pruning-aided Domain-Specific Weight Modulation (PaCDA):** Utilize pruning in a manner that distributes domain-specific model capacity, preserving knowledge learned from previously experienced domains and avoiding catastrophic forgetting.

Specifically, the framework aims to:

1. Minimize the domain gap between source and sequential target domains through progressive alignment.
2. Allocating domain-related capacity in an efficient manner while information from all domains encountered is not lost.
3. Source-free constraint where it simply uses a pre-trained source model and unlabeled target domain data for adaptation.
4. It is computationally efficient during training and at inference-time, making the system scalable for real-world applications.

The proposed approach actually aims at a good performance in scenarios of Continual Unsupervised Domain Adaptation, hence combining the capabilities of diffusion models in having gradual distribution alignment with pruning-based knowledge management for efficient domain-specific knowledge.

4 Methodology

4.1 Approach 1: Feature-Level Diffusion Replay (FDR)

4.1.1 Problem Setting

We consider a sequence of domains $\{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_T\}$, where $\mathcal{D}_0$ is a **labeled** source domain and $\mathcal{D}_1, \ldots, \mathcal{D}_T$ are **unlabeled** target domains encountered *sequentially*. Each domain $\mathcal{D}_i = \{x_j^i\}_{j=1}^{N_i} \sim P_i(x, y)$, but labels y_j^i are available *only* for $\mathcal{D}_0$. The challenge is to adapt continually to every new $\mathcal{D}_i$ while *retaining* performance on all past domains. Unless otherwise stated we follow the OfficeHome protocol (**Art** → **Clipart** → **Product** → **Real-World**, 65 classes).

4.1.2 Overview of Approach

FDR couples a conditional diffusion model with a mean–teacher pseudo-labelling loop so that *features* from previous domains can be replayed during training on the current domain (Fig. **??**). The pipeline proceeds in four stages:

1. **Source pre-training.** A *frozen-backbone ResNet-50* feature extractor (f_θ, output dim d=2048) and a linear classifier $g_\phi \colon \mathbb{R}^{2048} \to \mathbb{R}^{65}$ are trained on $\mathcal{D}_0$ with cross-entropy.
2. **Source diffusion modelling.** A 1-D conditional DDPM $\mathcal{M}$ is trained to model feature vectors $h = f_\theta(x)$ conditioned on class y *and* domain ID d.
3. **Domain-i adaptation ($i \geq 1$).** For each new domain the student network is updated with confident pseudo-labels plus diffusion replay of features from $\{\mathcal{D}_0, \ldots, \mathcal{D}_{i-1}\}$.
4. **Diffusion update.** A fresh diffusion head $\mathcal{M}_i$ is fine-tuned on domain d_i and appended to the replay bank.

4.1.3 Loss Functions

With trade-off weights λ_{cons} and λ_{replay}, the overall objective is $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \lambda_{\text{cons}}\mathcal{L}_{\text{cons}} + \lambda_{\text{replay}}\mathcal{L}_{\text{replay}}$, where each component follows Eq. (1)–(4) in the draft.

4.1.4 Conditional Diffusion Model

We use an 8-block 1-D U-Net with group-norm and SiLU, taking as input (h_t, t, onehot(y), onehot(d)). The noise schedule is linear, $\beta_t \in [10^{-4}, 2{\times}10^{-2}]$ for T=1000 steps; the training loss is $\mathbb{E}\| \mathcal{M}(h_t, t, y, d) - \epsilon\|^2$.

4.1.5 Training Phase

Stage 0 — Source supervised training

- **Backbone.** ImageNet-pre-trained ResNet-50, all convolutional layers *frozen*; only the 2,048×65 classifier is learned.
- **Optimiser.** AdamW (lr = $1{\times}10^{-4}$, weight-decay $1{\times}10^{-4}$), cosine schedule with 1-epoch warm-up.
- **Augmentation.** RandomResizedCrop(224), RandAugment(N=2, M=9), ColorJitter, horizontal-flip, and standard Imagenet normalisation.
- **Batch / Epochs.** 32 images, E_{src}=10.
- **Runtime.** ≈42min on one Tesla P100-16GB (mixed precision); ≈17min on one A6000-48GB.

Stage 1 — Source diffusion pre-training

- Extract all 2048-D features of $\mathcal{D}_0$ and cache them to disk.
- Train $\mathcal{M}_0$ for S_{src}=2×10^5 steps (batch 256, AdamW, lr $= 1\times10^{-4}$, EMA 0.999, gradient-clip 1.0).
- GPU memory $\approx$6.1GB; throughput $\sim$1,300 features/s on P100.

Stage 2 — Domain-i adaptation loop ($i \geq 1$)

- **Teacher EMA.** $\alpha = 0.999$.
- **Confidence threshold.** τ_i increases linearly from 0.60 to 0.80 over the first 25,000 iterations.
- **Optimiser.** AdamW, identical hyper-parameters to Stage 0.
- **Replay ratio.** Each mini-batch mixes 75% target images and 25% replay features.
- **Loss weights.** $\lambda_{\text{cons}} = 5,\ \lambda_{\text{replay}} = 1$.
- **Iterations.** $N_i = 50{,}000$ for each $\mathcal{D}_i$, $_{(i=1\ldots3)}$.

Stage 3 — Diffusion head update

- Fine-tune a new head $\mathcal{M}_i$ for $S_i = 2\times10^4$ steps on pseudo-labelled features from $\mathcal{D}_i$.
- Append $\mathcal{M}_i$ to the replay bank; all heads are kept frozen henceforth.

4.1.6 Inference Phase

A test sample $x^\star$ from any previously-seen domain is processed in a single forward pass:

$$p = g_\phi\big(f_{\theta'}(x^\star)\big), \qquad \hat{y} = \arg\max p.]Latencyis$$

$\approx$4.3 ms / image on a NVIDIA A6000 (batch 32), identical to vanilla ResNet-50 because diffusion replay is *offline*.

4.2 Approach 2: LoRA Expert using EMA Teachers

This second approach sequentially adapts a pre-trained Vision Transformer (ViT) to new domains by combining *Low-Rank Adaptation* (LoRA) — for parameter-efficient tuning — with an *Exponential Moving Average* (EMA) Mean-Teacher framework for unsupervised learning.

4.2.1 Problem Setting

Identical to Approach 1, we observe a domain stream $\{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_T\}$ in which only the first domain is labelled. Experiments are conducted on Office-Home (65 classes) in the order Art → Clipart → Product → Real-World.

4.2.2 Methodology Overview

A. Backbone Freezing and Data Preparation

- An ImageNet-initialised ViT is adopted as the shared backbone.
- All backbone parameters are frozen, yielding a stable feature extractor that remains unchanged throughout training.
- Domain-specific data splits and standard augmentation pipelines are prepared for each domain.

B. Phase 1 — Training the Source-Domain Expert

1. **LoRA Injection**: Rank-8 LoRA modules are inserted into each query–key–value projection of the transformer's self-attention layers; the original weights stay frozen.
2. **Supervised Optimisation**: Only the newly added LoRA weights and a single-layer classification head are trained on the labelled source domain using cross-entropy.
3. **Result**: The learned LoRA parameters and head weights constitute the *source expert*.

C. Phase 2 — Unsupervised Adaptation to Each Target Domain For every subsequent unlabeled domain the procedure is repeated:

1. **Student Initialisation**:
   - A fresh copy of the frozen backbone is taken.
   - New LoRA modules are added and initialised from the most recent expert.
   - LayerNorm affine parameters and a new classification head are also initialised from the previous expert and set trainable.
2. **Teacher Construction**: A duplicate of the student network is created, kept frozen, and updated only through EMA with decay 0.999.
3. **Mean-Teacher Loop**:
   - The teacher produces pseudo-labels for target-domain images; only predictions whose confidence exceeds an adaptive threshold ($0.6 \rightarrow 0.8$) are kept.
   - The student is trained on these confident pairs while the teacher follows the student through EMA updates.
   - Early stopping retains the student snapshot giving the best target-validation accuracy.
4. **Outcome**: The final LoRA, LayerNorm, and head weights form the *expert* for this domain, and will seed adaptation to the next domain.

D. Multi-Expert Inference Pipeline

1. **Domain Identification**: A shallow domain classifier, trained on frozen-backbone CLS features, predicts the most probable domain for a test image.
2. **Expert Selection & Prediction**:
   2.1. If the domain prediction is highly confident, the corresponding expert alone generates the final class prediction.
   2.2. Otherwise, the top-K candidate experts are queried under lightweight test-time augmentation; their outputs are averaged, and — if still uncertain — a heavier augmentation set is employed.

4.2.3 Key Characteristics

- **Parameter Efficiency**: Each domain trains fewer than 1 M new parameters (LoRA, LayerNorm, head) while 86 M backbone weights remain frozen.
- **Unsupervised Target Learning**: The Mean-Teacher design leverages unlabelled data without catastrophic forgetting.
- **Continual Knowledge Transfer**: Every new student starts from the immediately preceding expert, accumulating knowledge over time.
- **Modular Experts**: Domain-specific LoRA+head bundles can be loaded or fused on demand during inference.
- **Stable Backbone**: A single frozen ViT provides consistent, high-level representations across all domains.

Overall, LoRA-EMA supports continual adaptation to an evolving sequence of domains, keeping memory and computation nearly constant while progressively improving performance on each new domain.

4.3 Approach 3: LoRA–DAD with FixMatch & SHOT

4.3.1 Problem Setting

We consider the same continual unsupervised domain adaptation problem:

$$\{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_T\},$$

where only the first domain $\mathcal{D}_0$ (e.g., Art) is labeled and all subsequent domains (Clipart, Product, Real-World) are unlabeled. Experiments are conducted on Office-Home (65 classes).

4.3.2 Methodology Overview

Approach 3 fuses three complementary components at the *feature* level:

(i) **Low-Rank Adaptation (LoRA)** for parameter-efficient tuning of a shared frozen ViT backbone.

(ii) **Domain-Adaptive Diffusion (DAD)** to gradually align CLS-token features across domains.

(iii) A **multi-head unsupervised objective** combining

- EMA pseudo-labeling (Mean-Teacher),
- FixMatch consistency,
- SHOT entropy regularization.

A. Backbone Freezing & LoRA Injection

- Start from an ImageNet-pretrained ViT-Base/16 (`vit_base_patch16_224`).
- Freeze all original ViT weights and remove the built-in classifier.
- Insert rank-16 LoRA adapters into each QKV projection of the 12 self-attention blocks, with scaling factor $\alpha/r = 32/16 = 2.0$ and dropout rate 0.05.
- Unfreeze all LayerNorm affine parameters.

B. Phase 1 — Source-Domain Expert Training

1. **Head & LoRA Training**: Train only the LoRA adapters, LayerNorm affines, and a new single-layer head on labeled source data.
2. **Optimization**: Use cross-entropy with label smoothing 0.1, AdamW (LR $= 5\times10^{-4}$, WD 0.05), and polynomial decay for 10 epochs.
3. **Result**: Save LoRA weights + LayerNorm + head as the *source expert*.

C. Phase 2 — Unsupervised Adaptation to Each Target Domain For each unlabeled domain $\mathcal{D}_i$, $i \geq 1$, repeat:

1. **Student & Teacher Initialization**:
   - *Student*: copy frozen ViT, inject new LoRA initialized from prior expert, warm-start LayerNorm and head from prior expert.
   - *Teacher*: deep copy of Student; frozen parameters updated by EMA ($\rho = 0.999$).

- *DAD Denoiser* p_θ: lightweight MLP on CLS features, initialized anew (hidden 1024, timestep embed 128).

2. **DAD Pre-training**:
   - Corrupt CLS features of unlabeled $\mathcal{D}_i$ with Gaussian noise over $T = 200$ steps ($\beta \in [10^{-4}, 2\times 10^{-2}]$).
   - Train p_θ for 5 epochs to predict noise via MSE.
3. **Diffusion-Guided Mutual Learning (MLS)** over $k = 0 \dots T-1$:
   (a) $C \to D$: update p_θ by reconstructing noisy features of previous-domain labeled data and classifying via student head.
   (b) $D \to C$: freeze p_θ, update student LoRA & head on reconstructed features from step (a).
4. **Unsupervised Target Learning**:
   - *EMA Pseudo-Labels*: teacher labels weakly-augmented images (threshold 0.7→0.9); train student on confident samples.
   - *FixMatch*: student generates pseudo-labels at 0.95 confidence on weak aug.; train on strong aug.
   - *SHOT*: minimize conditional entropy + maximize marginal entropy on weak aug., with head frozen.
5. **Student Update & EMA**:
   - Aggregate losses, step student optimizer, update LR scheduler.
   - EMA update teacher from student.
6. **Validation & Early Stopping**:
   - Every 20 steps (and final), evaluate student on $\mathcal{D}_i$ val split.
   - Save best student; stop if no improvement over 3 checks.
7. **Expert Finalization**:
   - Load best student as $\mathcal{D}_i$ expert (LoRA + LN + head).
   - Seed next domain's initialization.

D. Multi-Expert Inference Pipeline

1. Train a shallow domain classifier on frozen ViT CLS features.
2. At test time, select top-K experts by domain confidence:
   - If high confidence, use single expert.
   - Else, average predictions under lightweight TTA, escalating if uncertain.

4.3.3 Key Characteristics

- **Parameter Efficiency**: <1 M trainable parameters per domain (LoRA+LN+head).
- **Feature-Level Alignment**: DAD bridges domains in feature space, avoiding pixel-level overhead.
- **Robust Unsupervised Learning**: EMA, FixMatch, and SHOT jointly mitigate noisy pseudo-labels and confirmation bias.
- **Continual Modular Experts**: Each domain yields a plug-and-play expert bundle.

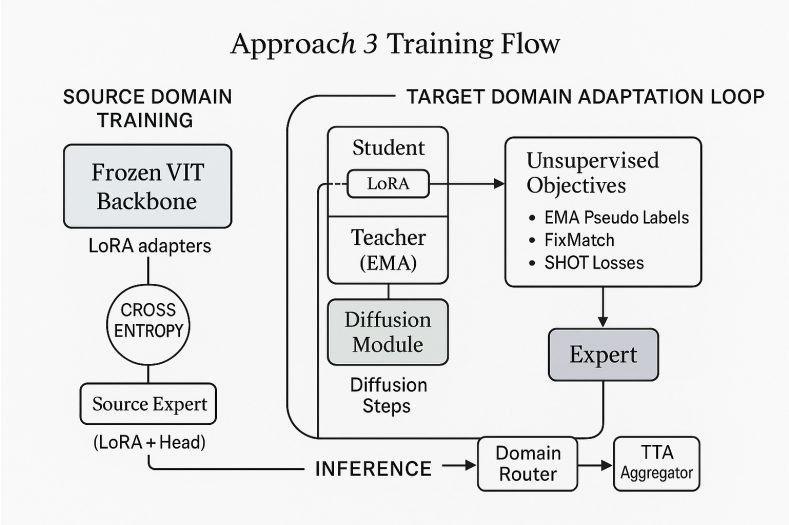

Figure 4.1: Flow for the approach

5 Theoretical/Numerical/Experimental Findings

5.1 Feature-based Diffusion Replay

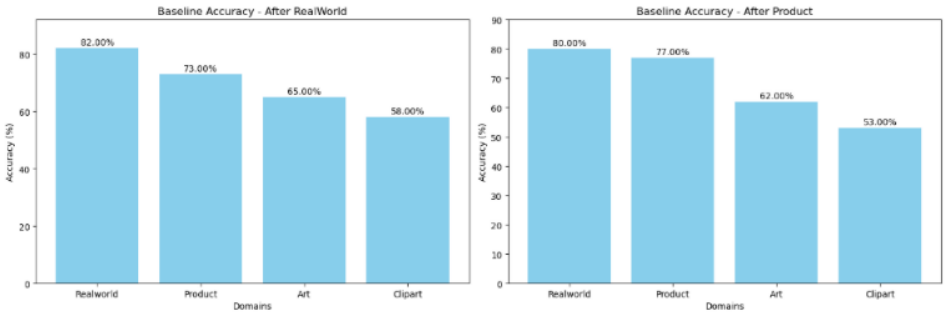

Figure 5.1: left - Zero Shot Baseline Accuracy after training on Real World. right - After adapting on Product

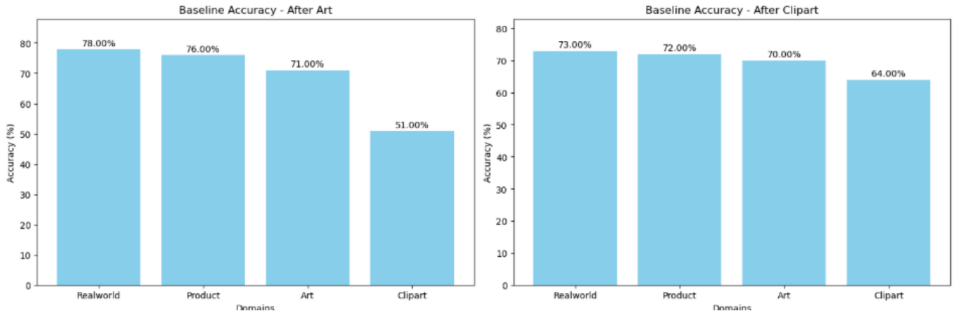

Figure 5.2: left - After adapting on Art. right - After adapting on ClipArt

5.2 LoRA Expert using EMA Teachers

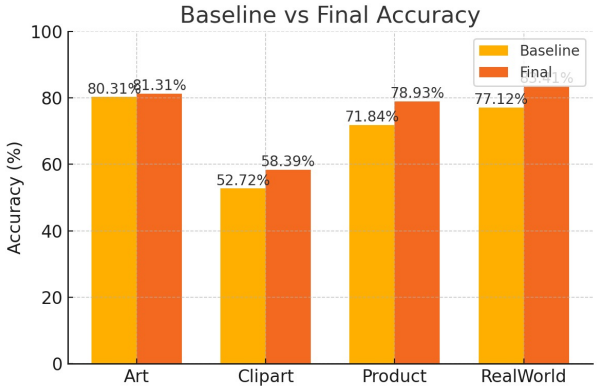

Figure 5.3: Baseline Accuracy after training on Art. right vs Final Accuracies without using Test Time Adaptation (TTA)

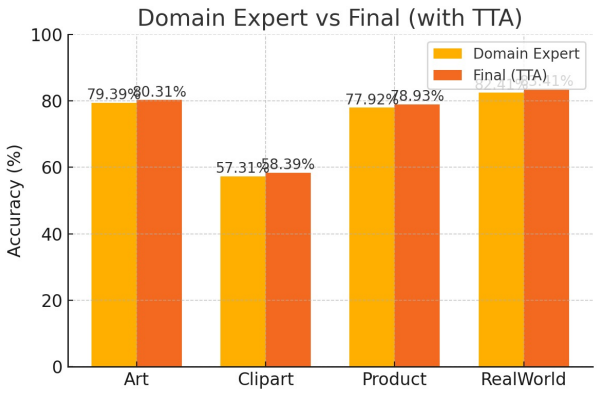

Figure 5.4: Comparison of Accuracies with and without TTA

5.3 LoRA–DAD with FixMatch & SHOT

Table 5.1: Comparison of continual UDA methods on OfficeHome (domain-incremental). Higher performance and no catastrophic forgetting via Experts compared to other methods

Rank	Method	Venue & Year	Source at Adapt.	Needs Domain-ID	Avg. Acc. (%)	BWT (%)
1	Ours: LoRA + Diffusion + FixMatch (ViT-B/16)	This work	✗ source-free	✗ entropy/router	81.1	+0.2 (no drop)
2	CoSDA (dual-speed teacher–student)	ICLR '24 (TMLR subm.)	✗	✗	70.76	–2.24
3	PaCDA (mask + BN router)	CVPR-CLVision '23	✗	✓ router	69.6 (ACPR)	–0.1 . . . –0.3
4	G-SFDA	ICML '22	✗	✓	∼66	–4
5	CoTTA	CVPR '22	✗	✗	62	–4.5
6	ConDA	ICCV '21	✓ buffer	✗	59	–11

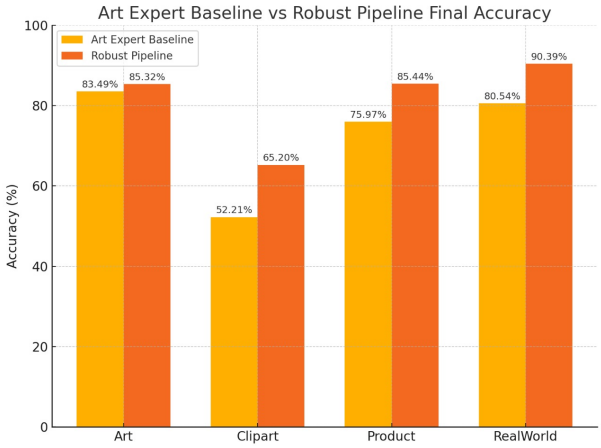

Figure 5.5: Art Expert Baseline Accuracy vs Robust Pipeline Final Accuracy

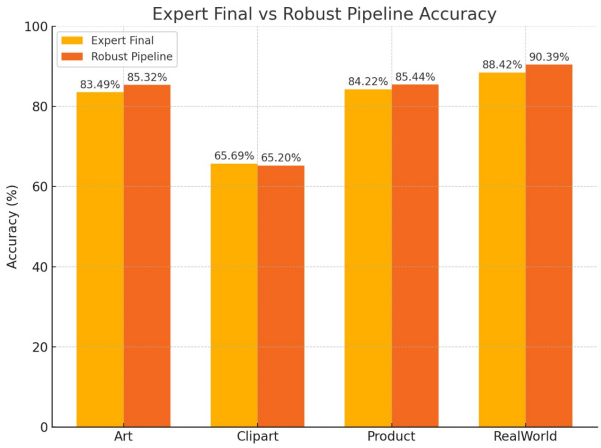

Figure 5.6: Final Expert accuracies vs Robust Pipeline Accuracies

6 Future Work

Although the framework presented here incorporates aspects of DAD, LoRA and various other UDA techniques, there are still a lot of opportunities for further research to develop the technique and broaden its application:

6.1 Uncertainty Estimation & Adaptive Triggering

1. Equip each expert with Bayesian LoRA (rank-dropout) or deep ensembles to quantify predictive uncertainty.
2. Use uncertainty thresholds to decide when to (a) invoke heavier test-time augmentations, (b) request human labels, or (c) spawn a new expert.

6.2 Memory-Constrained Expert Compression

1. Distil multiple domain experts into a shared Mixture-of-Experts backbone with sparse router gates, keeping per-domain LoRA only when essential.
2. Investigate rank-adaptive or block-sparse LoRA that shrinks inactive weights after a domain's relevance decays.

6.3 AutoML for DAD–LoRA Pipelines

1. Perform Bayesian optimisation / population-based training over diffusion steps K, LoRA rank r, FixMatch τ, and replay ratios to find hardware-specific Pareto optima.
2. Explore neural architecture search for ultra-light 1-D denoisers that maintain alignment quality while cutting parameters by $\geq 4\times$.

6.4 Fairness, Bias & Ethical Guarantees

1. Measure demographic bias drift across domain shifts; incorporate fairness-aware loss terms during diffusion alignment.
2. Derive theoretical upper bounds on disparate performance as a function of replay size and pruning ratio, guiding safe deployment in high-stakes settings.

6.5 Robust Test-Time Augmentations & Cross-Expert Knowledge Sharing

1. **Adaptive Augmentation Scheduler:** learn a policy (e.g. via reinforcement learning) that selects TTA intensity per sample, conditioned on expert-confidence and domain uncertainty.
2. **Augmentation Diversity Ensemble:** combine RandAugment, StyleMix, Fourier mixup and diffusion-based feature perturbations; use disagreement among experts as an augmentation-selection signal.
3. **Distillation Across Experts:** periodically distil a "shared student" from the pool of specialists under strong TTA, encouraging knowledge transfer without catastrophic forgetting.
4. **Feature-Level Replay Between Experts:** allow experts to exchange high-confidence pseudo-features through a small memory buffer, improving low-resource domains without accessing raw data.

6.6 Meta-Learning for Fast Domain Adaptation

1. **MAML-Style Initialisation:** meta-train LoRA parameters (and DAD denoiser weights) so that only 1–3 gradient steps are needed to specialise on a new domain.
2. **Task Construction for Meta-Training:** create pseudo-tasks inside the source domain (e.g. class, style, or resolution splits) to emulate domain shifts during the outer loop.
3. **Meta-Learned DAD:** update diffusion hyper-parameters in the outer loop, learning to pick the minimal number of steps K that still achieves alignment.

6.7 Modular & Composable Architectures

1. **HyperNetwork for LoRA Generation:** a domain-embedding encoder feeds a HyperNet that predicts LoRA (A, B) and head weights on-the-fly, enabling near zero-shot adaptation.
2. **Mixture-of-Experts (MoE) LoRA Layers:** each ViT block hosts multiple low-rank adapters; a lightweight gate chooses the top-k adapters per sample or per patch.
3. **Sparse Routing Loss:** encourage the MoE gate to reuse overlapping experts for similar domains, implicitly clustering them and yielding parameter sharing.

6.8 Generative & Ontology-Aware Paradigms

1. **Conditional Generative Alignment:** elevate DAD to a conditional diffusion model that translates ViT features across any pair of domains described by a learned code.
2. **Continual Self-Supervised Pre-training:** run an MAE/DINO phase on each new domain before supervised adaptation of LoRA, regularised by DAD to avoid feature drift.
3. **Domain Relationship Graph:** learn embeddings for domains and build a similarity graph; use graph distance to (i) initialise new experts, (ii) tune diffusion step budget, and (iii) schedule replay strength.

These guidelines aim to address deficiencies in the current framework and improve its functionality to meet more complex and diverse requirements under evolving scenarios of continual unsupervised domain adaptation.

References

[1] D. Peng, Q. Ke, Y. Lei, and J. Liu, "Unsupervised domain adaptation via domain-adaptive diffusion," *IEEE Transactions on Image Processing*, 2023. [Online]. Available: https://arxiv.org/abs/2308.13893v1

[2] P. B, S. Sanyal, and R. V. Babu, "Continual domain adaptation through pruning-aided domain-specific weight modulation," in *Proceedings of Vision and AI Lab, Indian Institute of Science*, 2023. [Online]. Available: https://arxiv.org/abs/2304.07560v1

[3] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of Learning and Motivation*, vol. 24, pp. 109–165, 1989.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: https://papers.nips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

[5] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep. TR-2009, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[6] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," California Institute of Technology, Tech. Rep., 2011. [Online]. Available: https://authors.library.caltech.edu/records/cvm3y-5hh21

[7] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010. [Online]. Available: https://arxiv.org/abs/1611.07725

[8] Q. Sun, E. Cetin, and Y. Tang, "Transformer^2: Self-adaptive llms," *arXiv preprint arXiv:2501.06252*, 2025.

[9] Sakana AI, "Transformer^2: Self-Adaptive LLMs (Blog Post)," 2025, https://sakana.ai/transformer-squared/.

[10] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[11] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *arXiv preprint arXiv:2305.14314*, 2023.

[12] X. Jin, X. Qiu *et al.*, "Adalora: Adaptive budget allocation for parameter-efficient fine-tuning," in *ICLR*, 2023.

[13] H. Zhong, S. Shen, K. Cai *et al.*, "Serial low-rank adaptation of vision transformers," *arXiv preprint arXiv:2503.17750*, 2025.

[14] J. Li, Y. Xie *et al.*, "Dual low-rank adaptation for continual learning with pre-trained models," *arXiv preprint arXiv:2411.00623*, 2024.

[15] Y. Chen, M. Zhao *et al.*, "Continual low-rank adaptation (c-lora)," *arXiv preprint arXiv:2502.17920*, 2025.

[16] H. Su, Y. Kalantidis *et al.*, "Explora: Parameter-efficient extended pre-training to adapt vision transformers," *arXiv preprint arXiv:2406.10973*, 2024.

[17] W. Zhang, C. Liu *et al.*, "Domain correlation low-rank adaptation for domain incremental learning," in *CVPR*, 2025.