Report: Applied Deep Learning

Domain Adaptation of CLIP model using DANN (adversarial learning)

Department of Statistics
Ludwig-Maximilians-Universität München

Dileep Vemuri, Anjali Sarawgi, Duc-Anh Nguyen

Munich, February 27, 2026

Contents

1 Introduction: Experimental Setup

In this project, we study performance of foundation model before and after fine-tuning and Domain Adaptation. To do this, we choose the CLIPmodel (Radford et al., 2021). First we create a baseline with Zero Shot learning for this dataset, and further fine-tune the model, and lastly use Domain Adaptation with the DANN method (Ganin et al., 2016) of adversarial learning.

1.1 Data

Here, we use the **DomainNet dataset** (Peng et al., 2019), which contains approximately 0.6 million images distributed among 345 categories and has 6 distinct domains. They include: Clipart, Infograph, Painting, Quick Draw, Real, S ketch. For this project, we use the **real** class as the source domain / the training set data, and the evaluation was focused on **Clipart** and **Infograph**.

1.2 The three stages

The first step of this project was to **evaluate the CLIP model adaptation using zero-shot learning** to check the pre-trained CLIP model's performance on the DomainNet dataset before it was trained on any of the images from this dataset. This is also our baseline for all the experiments ahead, especially to evaluate performance gain when CLIP encoder is fine-tuned, and Domain Adaptation is performed for the same.

For **fine-tuning**: we focus on the CLIP vision encoder as the DomainNet dataset has image classes as its labels. For our experiments, we fine-tune the entire CLIP encoder on the DomainNet dataset for the **Real** domain images, which caused **catastrophic forgetting of pre-trained knowledge**. We evaluate this using both with and without **label smoothening**. Further, we also **extract the feature embeddings** from the vision encoder of the CLIP model and attach an **MLP head of 2-3 layers** for the classification layer. Finally, for the fine-tuning stage, we also try using **Ensemble Methods** (Vahidi et al., 2023) with 3 subnets, where we evaluate model performance with and without diversity loss.

Finally, we use the **DANN method for Domain Adaptation of the CLIP encoder** on the **DomainNet dataset** (Peng et al., 2019). For our experiments, we again use the **Real** domain as the **source domain**, and the **Clipart** domain as the **target domain**, and perform Domain Adaptation for these two classes. Similar to the fine-tuning stage, we evaluate model performance both with and without freezing the CLIP encoder. We also experiment with the Lambda values for the Gradient Reversal Layer (GRL) (Ganin and Lempitsky, 2015), where we use the adapted lambda like in the main paper, and a constant lambda for evaluation purposes.

2 Results

2.1 Stage 1: Zero Shot Learning

Two steps: We first take class labels of the DomainNet dataset and use them to make text embeddings, then we extract image features using the CLIP image encoder. After that we use cosine similarly between these encodings to find which word vector (labels) is closest to the image. This helps us evaluate the CLIPs performance on the DomainNet dataset based on what it already knows from the pre-training pipeline.

Domain type	**Accuracy (Test Set)**
Real	0.78
Clipart	0.62
Painting	0.59
Sketch	0.55
Infograph	0.40

Table 1: Results from zero shot learning for 5 domains of the DomainNet dataset. This is evaluated for the entire dataset of each domain.

Here, we achieved 0.78 as the accuracy for Real, and 0.62 for Clipart, which form our baselines for the next steps.

2.2 Stage 2: Fine-tuning

In this stage, we have three distinct training setups to evaluate two different questions. They are:

2.2.1 Fine-tuning the full CLIP model

Domain	**lr**	**ls**	**Accuracy**
Real	0.001	yes	**0.86**
Clipart	0.001	yes	**0.65**
Infograph	0.001	yes	**0.38**
Real	0.001	no	0.85
Clipart	0.001	no	0.63
Infograph	0.001	no	0.37

Table 2: Results from fine-tuning for the full clip model. The results are evaluated on the test set. Note: ls = label smoothing. We train each model for 10 epochs.

2.2.2 Freezing the CLIP model and adding an MLP head (2 layers, 5 layers) for the classification layer

Domain	lr	Layers	Scheduler	Accuracy
Real	0.001	2	StepLR	**0.84**
Clipart	0.001	2	StepLR	**0.59**
Infograph	0.001	2	StepLR	**0.36**
Real	0.001	2	Cosine	**0.84**
Clipart	0.001	2	Cosine	**0.59**
Infograph	0.001	2	Cosine	**0.36**
Real	0.001	5	StepLR	0.73
Clipart	0.001	5	StepLR	0.43
Infograph	0.001	5	StepLR	0.23
Real	0.001	5	Cosine	0.73
Clipart	0.001	5	Cosine	0.42
Infograph	0.001	5	Cosine	0.20

Table 3: Results from fine-tuning with MLP heads, along with two schedulers. We train each model for 10 epochs.

2.2.3 Freezing the CLIP model and using a 3 model ensemble setup with 2-layer MLP heads

Domain	lr	Layers	Scheduler	Accuracy
Real	0.001	2	Cosine	**0.82**
Clipart	0.001	2	Cosine	**0.53**
Infograph	0.001	2	Cosine	**0.33**

Table 4: Results from fine-tuning with 2 MLP heads using 3 ensembles. We train each model for 10 epochs. We chose 2 layers because we achieved the best performance for 2 layers in Table ß3. We do this for without diversity loss.

2.3 Stage 3: Domain Adaptation using DANN

For the Domain Adaptation part, we use the adversarial learning method DANN. We closely follow the DANN implementation from the main paper, which includes the depth of the MLPs for the domain, class, and adapter sections with 2, 2, and 3 layers, respectively (Ganin et al., 2016). We also use an adaptive lambda, like in the main paper, but for additional experiments, we used a constant lambda of 0.5 as well. Further, it is important to note that DANN implementation in the main paper is not based on fine-tuning a pre-trained foundational model; it trains the model as it adapts to the domain. Due to resource capacities and time constraints, we decided to focus our experiments on freezing the CLIP encoder model and using MLP heads for adaptation. Furthermore, we also did a few experiments in the end for the unfrozen CLIP encoder, inspired by the main paper, just to explore it further, but the model's performance was very poor (more results below).
Here, we perform DANN for 2 classes, which is Real (source) and Clipart (target).

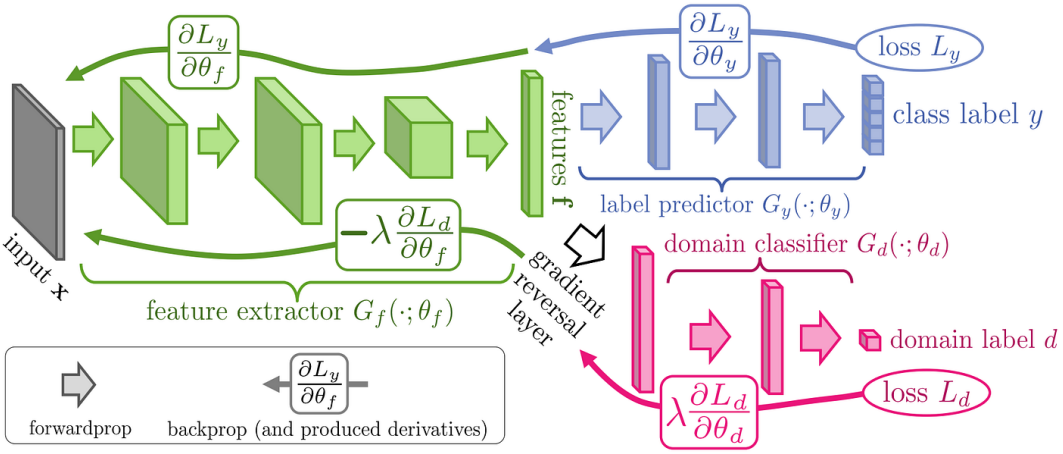

Figure 1: Green: deep feature extractor (here CLIP), blue: deep label predictor. We implement unsupervised domain adaptation by adding a domain classifier (red) connected to the feature extractor via a gradient reversal layer. Source: (Ganin et al., 2016)

2.3.1 DANN on CLIP with frozen CLIP encoder

Domain	lr	Layers	Lambda	Scheduler	Accuracy
Real	0.01	2, 2, 3	Adaptive	Cosine	0.83
Clipart	0.01	2, 2, 3	Adaptive	Cosine	0.58
Real	0.001	2, 2, 3	Adaptive	Cosine	**0.84**
Clipart	0.001	2, 2, 3	Adaptive	Cosine	**0.59**

Table 5: Results from fine-tuning with MLP heads, along with two schedulers. We train each model for 10 epochs. The Adaptive Lambda is set for a maximum of 0.8(scaled).

2.3.2 DANN with CLIP with unfrozen CLIP visual encoder

In this part, we test the model performance for DANN with a two-stage finetuning. We do this for two reasons: (1) to stabilise the training (2) to clearly test whether adding domain loss actually improves target performance. We follow the two setups:

- **Setup 1:** The entire CLIP model is frozen only for the first stage (6 epochs). In the second stage (from epoch 7 to 20), the CLIP visual encoder is unfrozen, and the domain loss is not used.
- **Setup 2:** Same as above but here the **domain loss is used**.

From the results, we see that when Domain Loss is added, the performance improves for the Target domain (Clipart), from 0.58 to 0.60, and thereby also reduces the Domain Generalization Gap by two points. This indicates that the DANN adversarial learning is working by reducing the domain generalization gap. Although the improvement is small (2 percentage points), it shows that adding domain loss helps the model transfer slightly

Domain	Lambda	Domain Loss Added	Accuracy	Domain Gap
Real	Adaptive (max 0.3)	No	0.84	-
Clipart	Adaptive (max 0.3)	No	0.58	26 pt.
Real	Adaptive (max 0.3)	Yes	0.84	-
Clipart	Adaptive (max 0.3)	Yes	0.60	24 pt.
Real	Adaptive (max 0.6)	Yes	**0.83**	-
Clipart	Adaptive (max 0.6)	Yes	**0.62**	**21 pt. ***

Table 6: Results from two-stage finetuning, with and without domain loss.

better to the target domain, while keeping the source performance the same. Finally, with Adaptive Lambda strength scaled up to 0.6, we achieve the best results with the least domain gap of 21 points, and the highest performance for the target dataset at 62%.

2.3.3 Visualisation Feature Embeddings using TSNE

Here, since we implement Domain Adaptation, we use TSNE to visualise how the embedding space changes before and after the adaptation.

Before Domain Adaptation, we can see clear clusters in the tsne plots reflecting the model's inherent understanding of the domain classes for different domains. For example, in the figure below, the blue dots show the source domain (real) and the orange points show the target domain (clipart).

(i) TSNE plot before Domain Adaptation:

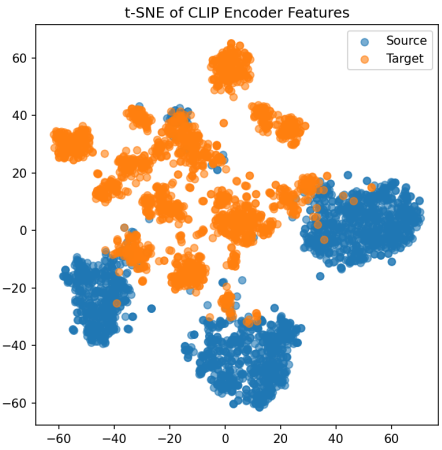

Figure 2: TSNE for feature embeddings before Domain Adaptation.

(ii) TSNE plot after Domain Adaptation:

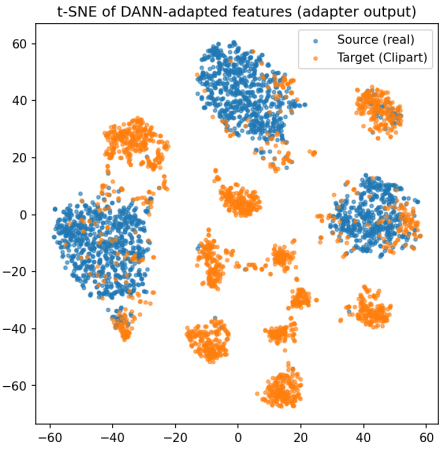

Figure 3: TSNE for feature embeddings after Domain Adaptation.

3 Discussion

From our results and experiments, the experimental setup for fine-tuning the full CLIP model gave us the best performance across all experiments. Here, we achieved a test accuracy of 0.86 on **Real** domain, 0.65 on **Clipart**, and 0.38 on **Infograph**. Nevertheless, when compared to the zero-shot learning performance, the **Infograph** domain accuracy reduces by about 2%. At the same time, we achieve higher gains for real and clipart domain datasets. A possible explanation to this is that **Infograph** domain dataset is highly different from the **Real** dataset. Additionally, when fine-tuned on the real dataset, we gain an 8% increase in performance, indicating that the model performance increases for the domain dataset fine-tuned on (the source domain **Real**), but does not improve performance as well as partly reduces on the other domains (**Infograph**, **Clipart**).

Furthermore, in Table 3, we have results for freezing the CLIP model while adding an MLP layer on top of the image encodings across different parameters, which include learning rate, number of layers in MLP head, and scheduler choices. We get the best results for learning rate at 0.001, 2-layer MLP, and a Cosine scheduler. Here, we can also see that the model performance does not improve more than the full fine-tuning stage as in Table 2; however, as compared to the zero-shot learning results on the source dataset, the model improves by 6%, again indicating that the model performance reduces for cross-domain generalization. Further, even for the experiments with Ensembles, our model performance improves by 4% on the source domain as compared to zero-shot, but still faces poor generalization on other domains.

Finally, when we implement Domain Adaptation for the two classes, i.e. Real as source, and Clipart as target, we see the performance does improve when compared to zero-shot by 5% on real datasets. Although it doesn't improve beyond the fine-tuning clip stage, even across different parameters. Finally, the two-stage training also indicates that, even though the gain is small, the adversarial training helps the model generalize a bit better on the target domain without reducing accuracy on the source domain. This task is also

challenging because the main paper of the DANN adaptation uses an architecture that doesn't involve using a pre-trained foundation model (Ganin et al., 2016). However, in our study, since we only used the image encoding, as we cannot re-train the CLIP model, and to also avoid catastrophic forgetting, this limitation may have shown less performance gain by using DANN over fine-tuning a CLIP model. Further, there may be other limitations that can be explored further in future experiments.

Moreover, from our results, we conclude that for this task, while the pre-trained model performs quite well by Zero Shot learning across 345 classes, both fine-tuning and DANN Domain Adaptation do yield some performance gain on the source dataset **Real**, and struggle to improve on the target dataset **Clipart**.

4 Electronic appendix

Data, code and figures can be found under this Github Repository:
https://github.com/DileepV17/Applied-Deeplearning

References

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation.
URL: *https://arxiv.org/abs/1409.7495*

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M. and Lempitsky, V. (2016). Domain-adversarial training of neural networks.
URL: *https://arxiv.org/abs/1505.07818*

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K. and Wang, B. (2019). Moment matching for multi-source domain adaptation, *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1406–1415.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G. and Sutskever, I. (2021). Learning transferable visual models from natural language supervision.
URL: *https://arxiv.org/abs/2103.00020*

Vahidi, A., Wimmer, L., Gündüz, H. A., Bischl, B., Hüllermeier, E. and Rezaei, M. (2023). Diversified ensemble of independent sub-networks for robust self-supervised representation learning.
URL: *https://arxiv.org/abs/2308.14705*