PA 2: Domain Adaption and Generalization

Muhammad Hamza Habib [1] **Abdul Samad** [2] **Rumaan Mujtaba** [3]

Abstract

In this report, we analyzed different techniques used to adapt to new domains and generalize to unknown domains. For the domain adaptation task, the key finding is the severe performance degradation caused by the domain shift. Among the domain generalization techniques, we found that SAM performs better than IRM and GroupDRO, as it tries to find the geometrically flat minima without explicitly trying to optimize for a specific domain or applying restrictive penalties on the model, which can lead to a loss of class discriminability. Moreover, in the case of CLIP, it successfully adapted across multiple domains with prompt tuning, which led to the preservation of CLIP's generalization ability. However, this also introduced overconfidence and open set robustness in CLIP underscoring the importance of domain adaptation and generalization techniques.

1. Introduction

Out of distribution data has always been a challenging problem in the field of Machine Learning. A model is trained on huge amount of data, and it gives state of the art performance in the controlled environment. But the same model fails when it is deployed outside the controlled environment. This drastic drop accuracy is common for machine learning models and it happens because models fail to perform adequately on out of distribution data. Models learn domain specific features while helps them to perform better on the in-distribution data, but these features lead the model to failure when exposed to out of distribution data. This report aims to investigate several algorithms used in **domain adaptation** (when out of distribution data is available but unlabeled) and **domain generalization** (when out of distribution data is not available), which are designed to improve these models' performance on out of distribution data. The codebase of the experiments conducted is available on this link.

2. Methodology

2.1. Task 1

The experiment is started by importing the required libraries and downloading the **PACS** dataset. The downloaded images are then transformed using ImageNet which perform the required standard preprocessing steps for models pretrained on ImageNet. Dataloader instances are created for each domain and all loaded datasets and dataloaders are organised using the domain data dictionary. The model backbone to be used in all the subtasks is defined. The feature extractor uses a pretrained **Resnet-50** model with the final classification layer removed, consistent output feature dimension is ensured followed by flattening. The forward function is used to extract a high dimensional feature vector of the input image. The classifier has a fully connected layer, outputs 7 classes for **PACS**, has logits for classification.

For the first subtask which requires source only model training, the name of the source domain is set to *photo* and the name of the target domain is set to *art_painting*. The model is initialized by creating instances of FeatureExtractor and Classifier, which are optimized using the Adam optimizer and a learning rate of 0.001, and the cross entropy loss function is used. The model is trained over 20 epochs, where in each epoch the optimzer gradients are zeroed, image features are extracted followed by logit generation, loss is computed between the true labels and the logits, backpropagation is performed and model parameters are updated. The loss of each epoch in printed.

For the domain alignment based adaptation task, we start by loading the previously trained source only model. MMD loss is defined which initializes parameters for the RBF kernel and computes the Gaussian kernel matrix between the features. The DAN model is initialized using the same optimizer and learning rate as the previous subtask, with the weight for MMD loss set to 1.0. Training is done over 20 epochs with the objective of combining classification loss on source domain using cross entropy loss and MMD loss on target domain, both the source and target datsets are simultaneously iterated. The total loss is computed as the sum of the classification loss and MMD loss. Proxy distance for domain alignment is computed and the results are stored.

For the DANN, a GRL (gradient reversal layer) is implemented and the DANN model architecture is defined which integrates a shared feature classifier, a task classifier and a domain classifier. Features are first extracted from input and passed through the task classifier which gives class output then GRL is applied to features before entering domain classifier which distinguishes between source and target domain features. The model is the initialized and trained using an adversarial. Domain loss is computed as sum of loss on source domain and loss on target domain. Total loss is sum of cross entropy loss in classification and domain loss.

The CDAN model makes the decision of the domain discriminator dependent on both the image features and the predicted class probabilities. The model extracts image features,predicts image classes and outputs logits which are converted to probabilities using softmax. The conditional input is a vector which is sum of image features and computed logit probabilities, GRL is applied to this conditional input and passed to domain classifier which attempts to find domain specific features for a certain class prediction. The hyperparameter for the model are the weight for domain loss and weight for entropy loss both set to 1.0. The model is trained to minimize 3 losses, classification loss, domain adversarial loss and entropy loss.We also compute the negative transfer effect for three rarest classes in target domain relative to our source-only baseline. This is done for each of the three alignment techniques implemented, including our source-only baseline

For the self training task, new feature extractor and classifier are created and initialized with weights from the source only model, Adam optimizer and cross entropy loss are used with 0.0001 learning rate. Pseudolabels are generated and a confidence threshold is applied, only pseudolabels higher than threshold are kept and then used for fine tuning in which model computes cross entropy loss against its own generated pseudo labels. The model is then evaluated on the source and target datsets.

In concept shift, a speceific class is chosen to be the rare class index and instances of this class in target domain are notably reduced, corresponding new dataset and dataloaders are created and all previously trained models are evaulated again, overall classification accuracy is calculated and confusion matrices are generated for each method on the updated target domain to see their performance on the rare class. F1 score for the rare class is calculated for each method and total target accuracy on modified dataset is recorded. A distribution heatmap is generated to compare how original class distribution differs from modified class distribution.

For each subtask a t-SNE visualisation is generated to observe the change in feature space compared to the source only baseline.

2.2. Task 2

This part of report aims to explore several Domain Generalization techniques aimed at improving model's performance, the goal was to analyze several domain generalization techniques like IRM, GroupDRO, and SAM which promise to improve performance on unseen domains.
A **Resnet-50** backbone, that was pre-trained on ImageNet, was used throughout these experiments to ensure consistency across multiple experiments conducted. All models were trained for multi-class classification across the benchmark dataset **PACS**. Throughout each experiment, models were trained on three of the source domains (Art painting, Photo, and Cartoon), and tested on one target domain (sketch). Moreover, 50 images from each source domain were already isolated for evaluated trained models on source domains. For reproducibility, the random seed used during all these experiments was 42. During all these experiments, all images were resized to 224×224 pixels and normalized using ImageNet's mean and std.
ERM (Baseline Experiment): A pretrained Resnet-50 model was downloaded from PyTorch library and fine-tuned using an Adam optimizer using a learning rate of 1×10^{-4}, along with $Loss_{CE}$ to note down the baseline accuracy. The batch size used during the training was 32, and the model was fine-tuned for 15 epochs.
IRM: For IRM, the same Resnet-50 backbone was downloaded and fine-tuned with Adam loss function using a learning rate of 1×10^{-4}. IRM training was drastically unstable, and this learning rate worked best to avoid the penalty collapse. The IRM implementation was taken from DomainBed (Gulrajani & other contributors, 2020), but the penalty computing term was replaced with the IRM v1 penalty, which seemed to be more stable. The original DomainBed implementation computes the IRM penalty by splitting each domain's data into two groups and measuring the variance of gradients across these splits. It was replaced with original IRM v1 penalty computation (See Equation 1).

$$\mathcal{P}_{\text{IRM-v1}}(\mathbf{\Phi}, w^*) = \left(\frac{\partial \mathcal{L}(w^* \odot \mathbf{\Phi}(\mathbf{x}), \mathbf{y})}{\partial w^*} - 1.0 \right)^2 \quad (1)$$

GroupDRO: For performing experiment with GroupDRO, the implementation of GroupDRO was taken again from (Gulrajani & other contributors, 2020) repository. GroupDRO follows a min-max pattern that focuses on the worst group performance. A pretrained Resnet-50 was employed, replacing its classification head with a task specific linear layer outputting 7-class predictions. An Adam optimizer with learning rate of 5×10^{-5} was used along with weight decay of 1×10^{-4}. At each iteration, one batch from each domain was sampled to ensure multi-domain training.

Sharpness Aware Minimization: To run this experiment, we applied SAM on a pre-trained Resnet-50 backbone to

evaluate it on PACS domain generalization benchmark. Implementation of sharpness aware minimization was taken from (, davda54). The classification head was replaced with a Linear layer with 7 output nodes. The hyperparameters used during training were perturbation magnitude of 0.05, base optimizer was Adam along with $Loss_{CE}$ function.

2.3. Task 3

We start by creating simple prompts (i.e. *a photo/sketch/cartoon/painting of a* $\{class\}$*)* for all the domains of our loaded **PACS** dataset. Using these prompts and our dataloaders for each domain, we compute the zero-shot classification accuracy of **CLIP**. This is done by multiplying our encoded prompts/text features with the encoded image features to get **CLIP's** prediction and compute top-1 accuracy.

Then for our fine-tuning approach, we take one domain of **PACS** dataset as target domain, and all the other as source domains to get a combined feature and label tensor from our feature extractor - each domain acts as the target domain once. The feature extractor does this by by encoding and normalizing the images in that dataloader and returns the concatenated features along with their corresponding labels. Then for the training step, we initialize the linear classifier as one fully connected layer, the cross entropy loss function, and Adam optimizer. For *epochs = 20* and *learning rate = 0.001*, we pass the combined source domain features, from the feature extractor, to the linear classifier to get the logits. These are used to compute our loss - using our defined loss function - and backpropagate. Then we evaluate our linear classifier by using logits of our target domain features to get predictions and compute evaluation accuracy on each domain.

Then for our domain adaptation task, we use *Photo* domain as source and *Art painting* domain as target of **PACS** dataset. We base our implementation on CoOp (Zhou, 2022), and create a TextEncoder and PromptLearner class. The PromptLearner class first concatenates the randomly initialized normalized learnable tokens for each class with the frozen CLIP token embeddings for each class name to create class prompts. These class prompts are then padded to ensure consistent shape, and passed to the TextEncoder class to encode these and return the normalized encoded text features. The TextEncoder class encodes these prompts by adding positional embeddings and passing them through text transformer of **CLIP's** model. These are normalized and projected to **CLIP's** embedding space to return the final text feature vectors.

Next, we start our training process by freezing **CLIP's** backbone, initializing our learned prompt features - using the PromptLearner class -, defining cross entropy loss function and Adam optimizer. Using *epochs = 10* and *learning rate = 0.001*, we iterate over the corresponding batches of the source and target domain dataloaders. Using the **CLIP** model, we encode images of our source and target dataloaders and compute their respective logits using our text features/learned prompt features. For the source domain loss, we use the defined loss function, but for the unlabeled target domain, we use the entropy minimization loss from (Grandvalet & Bengio, 2005). At the end of our training process, we have the trained prompt learner. Finally, we evaluate our prompt-learner adaptation by computing top-1 accuracy of the trained prompt learner on the target domain (*Art painting* domain of **PACS**).

For the gradient conflict and alignment task, we take the case of two source domains, specifically, **PACs** *Cartoon* and *Sketch* domain. We will be using the same prompt learner, loss function, and optimizer as in the previous task. With *epochs = 15* and *learning rate = 0.001*, we train our prompt learner with two different scenarios. In one case we train it without alignment, and in the next one, we do it with PCGrad alignment from (Tseng, 2020). During training, for both scenarios, we iterate over the corresponding batches of both source domain dataloaders, compute image features, and then logits using the prompt learner text features. For both domains, we use the defined loss function and total loss becomes the sum of these individual losses.

Then, after every three epochs we want to calculate the cosine similarity for the gradient vectors of both domains. So, we start by computing gradients of our objectives (losses of domains) with respect to our parameters (context vectors/normalized learnable tokens of our prompt learner). The list of gradient per parameter for each domain is flattened and concatenated to get concatenated gradients for each domain. The cosine similarity is then computed between these gradients, and losses are backpropagated. In addition to this, in the second scenario only, using the PCGrad technique, we project one gradient onto the normal plane of the other if they are conflicting (negative dot product ≤ 0). The cosine similarity is then computed again using the projected gradients and backpropagation is done manually using the mean gradient that is applied to parameters. Metrics like cosine similarity and training losses are saved. For the second scenario only, cosine similarity before and after projection is saved. These are used to compute the average cosine similarity for each scenario/method, top-1 accuracy of each method on each source domains, and a graphical plot of cosine similarity against epochs for each method.

For the open-set and generalization task, we ,first, use *Cartoon* domain of **PACS** dataset for the open-set

experiment. By randomly choosing 80% of **PACS** classes for training and 20% for evaluation, we create dataloaders for the seen (used for training) and unseen classes (used for evaluation). For training, we use the same prompt learner, loss function, and optimizer as in the previous task. Using *epochs = 10* and *learning rate = 0.001*, the prompt learner is trained by iterating over the batches of *Cartoon* domain of **PACS** dataset. The top-1 accuracy of the tuned prompt learner is first computed on seen classes dataloader, and then for evaluation on the unseen dataloader, the mean entropy and maximum softmax probability (msp) is computed.

Then, to compare zero-shot **CLIP** vs tuned prompts in terms of recognizing unseen classes, we use simple prompts (*a cartoon of a* $\{class\}$) to, first, compute the zero-shot text features for the seen classes using **CLIP's** tokenizer and text encoder. Since we already have text features from the tuned prompt learner, we use these to compute msp and store corresponding labels for both seen and unseen dataloader. These scores and labels are then used to compute AUROC and FPR@95TPR for both zero-shot tuned prompt **CLIP**. Finally, to check how similar learned prompts from source and target domain are, we use *Sketch* domain of **PACS** dataset as source and *Cartoon* domain as target. We get the learned prompt vectors for both of these, flatten them, and compute the cosine similarity.

3. Results

3.1. Task 1

The accuracy of our **ResNet** trained on source domain data (*Photo*) on our source domain testset is computed as 98.98% while on target domain (*Art Painting*) is computed as 25.59% for our no-adaptation baseline. The drop in accuracy from source to target domain, from the above accuracies, is calculated to be 73.3%.

For the Domain-Alignment based adaptation techniques, the accuracy on the source and target domain of each technique is give in **Table 1**. The results demonstrate that two of the domain alignment techniques (DANN and CDAN), both increase model's accuracy on target domain dataset as compared to source-only baseline.

Method	Source Accuracy (%)	Target Accuracy (%)
DAN	25.87	21.92
DANN	97.66	29.25
CDAN	93.95	29.59

Table 1. Source and Target Domain Accuracies of Each Alignment Technique

Then, to quantify any negative transfer effects, we calculate the F1-score for the three rarest classes in target domain across our domain-adaptation techniques, relative to Source-only baseline. The results are summarized in **Table 2**

Table 2. F1-scores for the three rarest target classes across domain-adaptation techniques

Method	Guitar (3)	Horse (4)	Elephant (1)
Source-Only	0.2407 (0.0000)	0.1739 (0.0000)	0.1988 (0.0000)
DAN	0.0000 (–0.2407)	0.0000 (–0.1739)	0.0000 (–0.1988)
DANN	0.2658 (+0.0251)	0.1201 (–0.0538)	0.1860 (–0.0128)
CDAN	0.1935 (–0.0471)	0.1166 (–0.0573)	0.2304 (+0.0316)

In addition to this, the proxy-distance between feature distribution of source and target domain is given in **Table 12**.

Method	Proxy-Distance
DAN	0.4492
DANN	0.1396
CDAN	0.1547

Table 3. Proxy-Distance between Source and Target Domain Feature Distributions

For the self-training task where we fine-tuned our model on pseudo-labels obtained from the source-trained model predictions, the accuracy on the source (*Photo*) is computed as 95.81%, while on the target (*Art Painting*) is calculated as 25.34%. In addition to this, the t-SNE visualization map for the feature-space alignment between source and target domain after after self-training is demonstrated in **Figure 1**

Method	Target Accuracy (%)	F1-Score
Source-only Baseline	26.03	0.00695
DAN	26.32	0.0
DANN	30.25	0.0513
CDAN	30.83	0.0714
Self-Training	27.90	0.2790

Table 4. Concept-shift Accuracy and F1-Score of Each Method on Target Domain

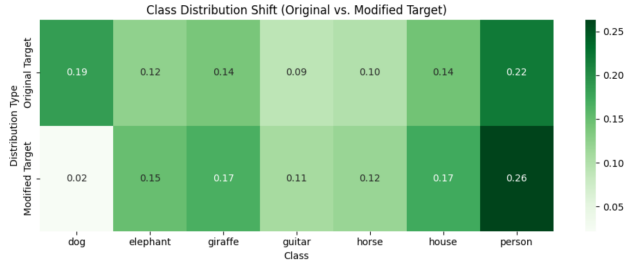

Figure 1. Heatmap Representing Change in Class Distribution Before and After Applying Concept Shift

Finally, for the concept shift, the accuracy of each technique/method implemented above on the target domain (*Art Painting*) after application of semantic shift is computed. The F1-score for negative transfer effect is also computed, and results are summarized in **Table 4**. At the same time, **Figure 1** illustrates a heatmap representing class distribution before and after applying concept shift.

3.2. Task 2

ERM (Baseline Experiment): The baseline experiment was performed to evaluate how a discriminative model like Resnet-50 trained with empirical loss minimization performs when evaluated on an unseen domain of data. The model was also evaluated on an unseen subset of source domains to evaluate. Through these evaluations, model was performing significantly well on all on sources with an average accuracy of above 80%, but fell down to an average accuracy 43.8% on unseen target domain (sketch). The performance matrices for experiment are given in Figure 2. This fall of accuracy on unseen data domains is common for models trained via ERM.

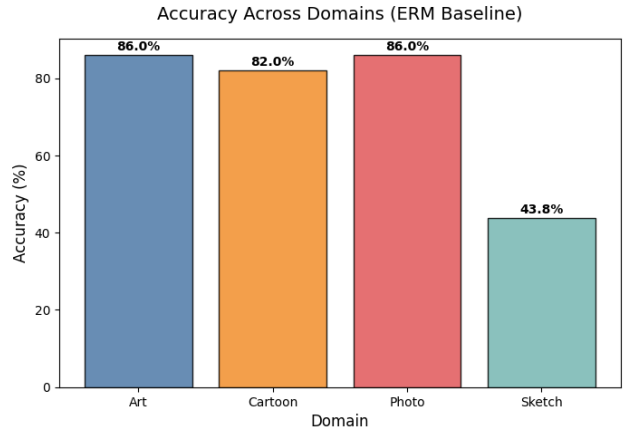

Figure 2. Resnet-50 trained via ERM performance on different PACS domains

IRM (Invariant Risk Minimization): IRM experiment was run multiple times experimenting with the penalty, as it was prone to penalty collapse. The model was evaluated on both source domains and target domains. Throughout multiple runs, IRM was unable to outperform ERM on unseen target domain getting an accuracy drop of 10% to 15%. The final values for the experiment run where IRM did manage to outperform ERM are given in the graph 3, along with a final average penalty value of 0.2. IRM outperformed ERM by a margin of 20%, along with improvement in accuracy over source domains.

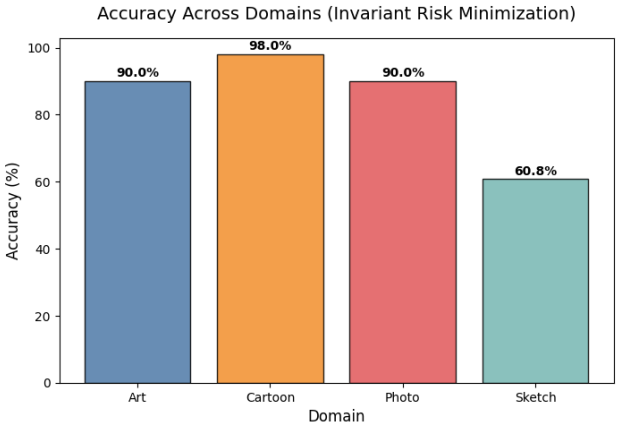

Figure 3. Resnet-50 trained via IRM performance on different PACS domains

Group DRO: While running the GroupDRO algorithm, the model improved training accuracy stably over 5 epochs. After training, the model was evaluated on an unseen subset of source domains. Model trained with GroupDRO achieved an average accuracy of 95.29% on the unseen subset of source domains. The model was then evaluated on the target domain (sketch) where it managed to achieve an accuracy of 67.93%. During training, the worst performing group was Art Domain, and GroupDRO managed to improve its accuracy by 2.69% (89.24% to 91.93%) over the course of 5 epochs (See Figure 3.2). The model achieved its best target domain accuracy (73.56%) on 4^{th} epoch, and reduced to 67% on the next epoch. This shows that GroupDRO could not stop the model from learning domain-specific spurious correlations.

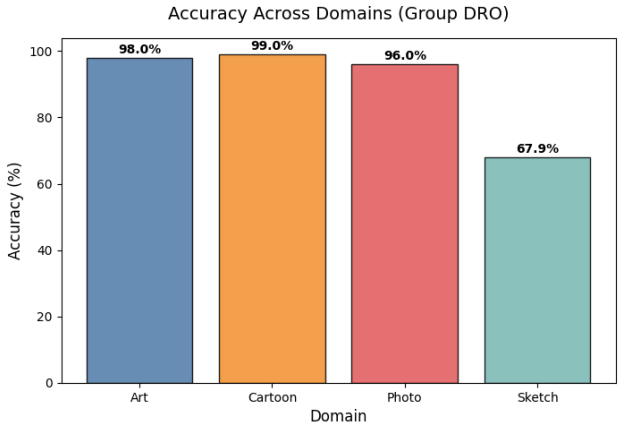

Figure 4. Resnet-50 trained via GroupDRO performance on different PACS domain

Sharpness Aware Minimization: After training, the model was evaluated on the held out source domain subsets and target domain. The model achieved an average accuracy of 98.67% on the held out subset of source domains. This high accuracy indicates that the model managed to learn the classification task on source domains successfully. The same trained model was then evaluated on the unseen target domain (sketch dataset) where the model scored an average accuracy of 72.05% (See Figure 5) While this degradation of 26.02% is significant, the inherent challenge of domain generalization is significant, the sketch domain carries a completely different visual representation compared to realistic and stylized images.

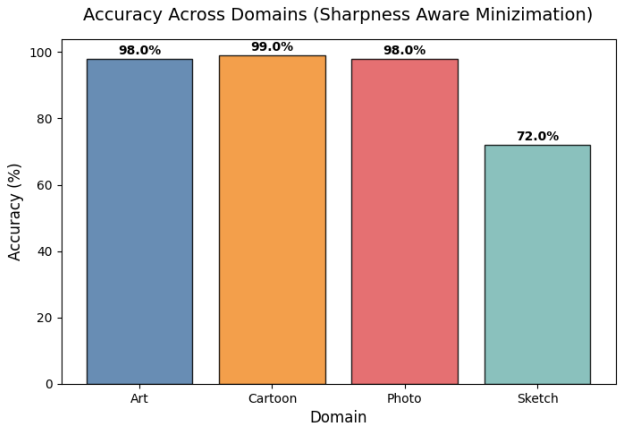

Figure 5. Resnet-50 trained via IRM performance on different PACS domains

An overview of all the experiments run, and their average accuracies is also given in 5. The worst case performance among all of the source domains and target domain is given for comparison.

Table 5. Table Showing worst case source domain performance vs target domain performance

Algorithm	**Worst Source Accuracy (%)**	**Target Domain Accuracy (%)**
ERM	82.0	43.8
IRM	90.0	60.8
GroupDRO	96.0	67.9
SAM	98.0	72.0

3.3. Task 3

The zero-shot accuracy of **CLIP** on all domains of **PACS** dataset for simple prompts (i.e. *a photo/painting/sketch/cartoon of a* $\{class\}$*)* is given in **Table 5**.

Domain	**Accuracy** (%)
Photo	17.13
Art Painting	17.33
Sketch	11.94
Cartoon	16.77

Table 6. Zero-Shot Classification Accuracy of CLIP on Domains of PACS Dataset

Then, for the fine-tuning part, the classification accuracy of our trained linear classifier on each target domain is given in **Table 6**. These results reinforce the argument that due to **CLIP's** broad training, the zero-shot baseline is hard to beat on some domains.

Target Domain	**Accuracy** (%)
Photo	11.44
Art Painting	20.23
Sketch	22.78
Cartoon	14.99

Table 7. Classification Accuracy of Trained Linear Classifier on Target Domains

Next, for our domain adaptation task, where we train our prompt parameters on the source domain (*Photo*) and also incorporate the target domain (*Art painting*) data via an unsupervised loss, the evaluation accuracy of **CLIP** on the target domain data using the trained prompt features is provided in **Table 7**, along with comparisons with our previous approaches performance on the same target domain.

Approach	Accuracy (%)
Zero-Shot Classification	17.33
Fine-Tuning with Linear Classifier	20.23
Prompt-Learning Adaptation	22.17

Table 8. CLIP Evaluation Accuracy for Different Approaches

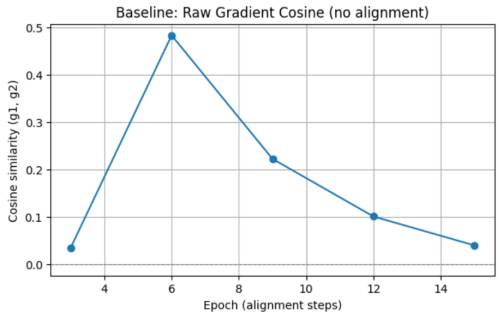

Figure 6. Cosine Similarity of Gradient Vectors Without Alignment During Training

Then, for the gradient conflict task, we measure the cosine similarity of the gradient vectors of the two source domains (*Cartoon* and *Sketch*) every three epochs. The average cosine similarity over all epochs between these gradient vectors with and without alignment is calculated to be 0.3337, and 0.1761, respectively. The cosine similarity graph of the gradient vectors during training without alignment is given in **Figure 1**. At the same time, the cosine similarity graph of the gradient vectors for when PCGrad alignment is applied during training is given in **Figure 2**. From **Figure 1** it can be observed that gradient vectors align in the beginning, but then, as training progresses, the gradients diverge and cosine similarity starts to fall to zero. On the other hand, **Figure 2** shows frequent conflict in gradients, since they're often negative. But then the green points clustering in the positive region imply that PCGrad is removing conflict such that the projected gradients point in the same direction.

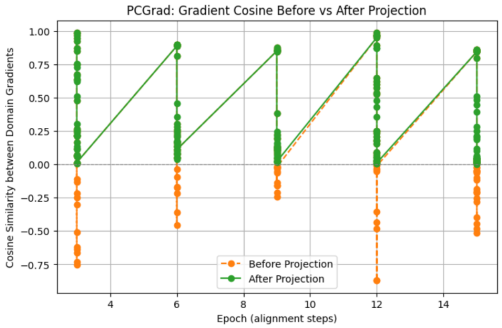

Figure 7. Cosine Similarity of Gradient Vectors With PCGrad Alignment During Training

For the open-set task, we first compute the top-1 accuracy of **CLIP** on the seen classes dataloader using the trained prompt learner. This is calculated as 50.39% because the prompt learner was also trained on the seen classes dataloader. Then, we compute the mean msp and entropy of **CLIP** on the unseen classes dataloader using our trained prompt learner. Mean msp, which represents the confidence on unseen data, is calculated to be 0.8614, while mean entropy, which represents uncertainty on unseen data, is calculated to be 0.3985. Then, for the comparison of tuned prompts (trained on seen classes) and zero-shot prompts (of seen classes) on unseen classes data, the results are summarized in **Table 5**.

Methodology	AUROC	FPR@95TPR
Zero-Shot	0.52	0.95
Prompt-Learning Adaptation	0.56	0.95

Table 9. Performance Comparison of CLIP on Unseen Classes with Different Methodology

Finally, the cosine similarity of the prompt embeddings of source (*Sketch*) and target (*Cartoon*) domain is calculated to be 0.0132, suggesting that the prompt embeddings are not quite similar.

4. Discussion

4.1. Task 1

The large accuracy drop (73.3 percent) from source to target domain in the source only baseline shows the effect of domain shift. The model which is highly accurate on photographic images performs very poorly on art paintings. The nature of this domain shift seems to be mainly stylistic as photographic images are more realistic in texture and lighting while paintings are more abstract and vary in texture and colour. The model is unable to properly generalise the features it learned from the source domain to the target domain.

In domain alignment based training the DANN and CDAN (which have lower proxy distances than DAN) show slightly improved target accuracy whereas DAN shows reduced target accuracy, as the reduce divergence between source and target model distributions this shows that by aligning feature distributions models become more robust to domain shift however they have reduced source accuracy as compared to source only baseline which shows that aligning domains too much can cause a removal of important features which are class specific and can cause degradation of performance over source domain.

It is possible that aligning distributions cause the classifier

to confuse and perform even poorly, which indicates negative transfer. DAN shows negative transfer for all three rare classes while DANN nad CDAN show mixed results showing that the models do not priortize correct classification of rare classes over overall alignment.

In self training, improvement in target accuracy as compared to source only baseline was not observed.Potenial reasons for this could be noisy initial pseudo labels, simple self training method or the pretrained model not being strong enough.

Self training with the highest F1 score shows it has the highest negative transfer effect, this tells that not handling label noise properly can be very detrimental to model performance. CDAN and DANN show moderate negative transfer showing that while these models give better overall target accuracy they might still introduce harmful biases that can show in concept shift. DAN and source only have lowest negative transfer showing that they did not produce harmful domain specific bias

4.2. Task 2

This section aims to discuss the results in previously mentioned experiments. A primary goal of any domain generalization model lies in learning stable features across domains while minimizing the effects of domain specific spurious correlations, and achieving the best accuracy on unseen target domains. This section quantitatively investigates the performance of all 4 tested algorithms, identifying the top performer on the target domain, sketch images. Moreover, this discussion also analyzes the worst case performance among source domains. The ultimate test of a domain generalization method is its accuracy on an unseen target domain. 5 compares the performance of each algorithm on target domain against the worst performing source domain.

Empirical Risk Minimization: The baseline ERM model achieved an accuracy of 43.8% on unseen target domain. The baseline established the difficulty of domain generalizations. The significant drop in ERM's accuracy from source to target domain accuracy indicates that ERM tends to learn unstable spurious correlations in training data. All three DG algorithms show a significant improvement over this baseline.

Invariant Risk Minimization: The experiment did prove that IRM is significantly difficult to train than ERM. It has been discussed in multiple studies that IRM is prone to overfitting with penalty collapse, when the models learns a trivial solution to minimize penalty. Another significant factor is the sensitivity of penalty weight (λ). To minimize this instability, a common practice is to use a penalty annealing schedule increases λ to a higher value after a significant number steps. DomainBed's implementation uses this hyperparameter, and tuning this factor did help with stabilizing IRM. To quantitatively analyze the role of λ in these experiments, its effect on model's performance was tracked throughout experiments. For a value of 0.1, IRM behaves exactly like IRM, and its accuracy falls around ERM accuracy. For a penalty of 1×10^5, the model collapsed and converged to a trivial solution. The model performed best on $\lambda = 1000$. The main challenge was to find the optimal values for λ and annealing schedule term to maximize the performance of IRM on target domain.

GroupDRO: In the case of GroupDRO, the algorithm works by improving the model performance over the worst performing group, the experiment did verify the results. Unlike IRM, which seeks to learn the true domain invariant features, GroupDRO aims to focus on worst performing group and tries to improve DG performance. During the GroupDRO run, the model improved the target domain accuracy and then the target accuracy fell on the next epoch while the worst performing group accuracy still managed to improve. This shows that GroupDRO focused on improving the performance on worst group, and learned some unstable domain specific correlations which led to decrease in performance on the target domain. This also suggests that optimizing for worst performing in-distribution data is not the best way to improve performance on out-of-distribution data.

Sharpness Aware Minimization: Unlike IRM and GroupDRO, SAM does not explicitly tries to model for a specific domain but instead tries to find a geometrically flat minima in the loss landscape which also minimizes loss in the neighbourhood. Among all the experiments done above, SAM outperformed both IRM and GroupDRO. This raises the question if geometric regularization is the more optimal and robust way to improve a model's performance on out of distribution data. SAM's outperformance does indicate that finding such a solution which is more invariant to perturbations effectively translates to improved performance on unseen shift in out of distribution data.

Invariance vs. Discriminability: The results highlight the tradeoff regarding invariance and discriminability. IRM's approach of enforcing strict domain invariance can be too restrictive causing the model to lose class discriminability for domain invariance. In contrast, SAM's geometric regularization finds a robust solution without explicitly discarding features or favoring any specific domain during training, thus preserving the model's discriminative ability.

Loss Landscapes: The loss landscapes of each model can be visualized by manually perturbating the weights of model on each side of the minimum and then visualizing the average loss of the model.

4.3. Task 3

The zero-shot accuracy of **CLIP** across domains demonstrates that it performs moderately well across domains, but struggles on stylized ones, such as Sketch (accuracy = 11.94%). This result can be attributed to **CLIP's** pretraining distribution which consists highly of photographic and natural images.This explain why it was able to generalize well to domains like *Photo* (accuracy = 17.13%) and *Art painting* (accuracy = 17.33%).

The results of fine-tuning show **CLIP** was able to adapt better to unfamiliar representations, as the performance on *Sketch* (accuracy = 22.78%) domain increased significantly. However, the drop in the accuracy for *Photo* domain (accuracy = 11.44%) indicate overfitting and reduction in generalizability. Possible cause for this drop could be catastrophic forgetting, where model specializes to stylized textures. This informs us of the adaptation vs. generality trade-off: fine-tuning on source-domain statistics allows **CLIP** to perform better on similar target domains, but at the expense of broader features that made zero-shot robust.

The target performance of prompt-learner adaptation approach (accuracy = 22.17%) exceeds the prior approaches (see **Table 7**), suggesting that **CLIP** adapts better via learned prompts. However, since the absolute gains are small, it is important to note that the additional unsupervised target loss in the prompt tuning approach could have contributed to this gain.

Since prompt tuning updates only a limited set of prompt vectors to steer **CLIP's** static language-vision alignment towards domain-specific cues, it allows integration of domain knowledge without significant alterations to the backbone. The performance improvement shown by prompt tuning, thus, demonstrates that it balances domain-specific adaptation and domain-invariant generalization well. In practical terms, prompt-tuning is light-weight than full mode fine-tuning (e.g. CoOp literature) and tends to maintain a large proportion of **CLIP's** pre-trained invariances. Also, since the image encoder is frozen, **CLIP's** broad pre-training leverages prompt tuning to align better with the target domain, making it advantageous for domain shifts. On the other hand, prompts are fragile. They are sensitive to prompt length, initialization, may overfit to target signals, not capture all domain-invariant features, and harm open-set detection. This brittleness urges the need for domain alignment techniques like PCGrad which make prompt updates more domain-agnostic, while keeping it light-weight.

As highlighted by **Figure 1**, the gradient vectors of the two source domains are heavily conflicting in the later stages of the training process. This typically implies that as training progresses, after the initial stages, there are hard examples and features that are specific to one domain such that improving one domain hurts the other (domains start to specialize). The average cosine similarity of (0.1761), without alignment, suggests that there is moderate conflict, such that gradient are somewhat aligned, but often diverge. On the other hand, average cosine similarity of (0.3337), with PCGrad alignment, shows how alignment nearly double the cosine similarity between gradient vectors. This is because it encourages gradient updates in the direction which is more agreeable for the multiple domains involved - allowing model to learn more domain invariant components. This implies that gradient vectors are now more consistently pointing in the same direction, meaning that PCGrad has successfully reduced interference.

For the open-set experiment, the comparison between performance of zero-shot and tuned prompts on seen classes data show that prompt tuning provides slight improvement (AUROC increasing from 0.52 to 0.56). This indicates that **CLIP** still struggles to identify unseen classes data. The tuned prompt model is too confident when presented with unseen class samples, as highlighted by the high mean msp (0.86) and low entropy (0.39). This suggests that tuning prompts on a subset of classes improves the model's confidence in its trained domain, but impairs its ability to recognize unknown inputs, which harm its open-set generalization.

Additionally, the prompt embeddings for the source and target domain are nearly orthogonal as demonstrated by the cosine similarity of 0.0132. This suggests that instead of learning generalizable representations, the prompts have become highly domain-specific rather than domain-invariant. This implies that prompt tuning has introduced brittleness and overconfidence under domain shift. Mitigation techniques like gradient alignment, and prompt regularization could encourage domain-invariant prompt directions

5. Conclusion

The experiments over domain adaptation techniques showed a major accuracy drop on domain shift. The drastic performance drop on domain shift indicates importance of domain adaptation techniques. The key results for domain generalization on discriminative models indicate that IRM penalty leads the model to lose class discriminability for domain invariance. Moreover, it is evident from the results that GroupDRO learned spurious features while trying to improve worst case performance. In case of SAM, the re-

sults indicates that geometrically regularizing the model led to a more robust and invariant solution. In case of CLIP, CLIP was effectively adapted across domains through prompt tuning, which enhanced performance on shifted and stylized data while maintaining a large portion of its generality. It did, however, also result in overconfidence and limited open-set robustness, underscoring the necessity of alignment methods such as PCGrad to encourage more domain-invariant learning.

6. Contributions

References

(davda54), D. S. Sam: Sharpness-aware minimization (pytorch implementation). https://github.com/davda54/sam, 2021. Accessed: October 17, 2025.

Grandvalet, Y. and Bengio, Y. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, volume 17, pp. 529–536, 2005.

Gulrajani, I. and other contributors. Domainbed: A pytorch testbed for domain generalization. https://github.com/facebookresearch/DomainBed, 2020. Accessed: October 17, 2025.

Tseng, W.-C. Pytorch-pcgrad. https://github.com/WeiChengTseng/Pytorch-PCGrad, 2020. Accessed: October 2025.

Zhou, K. Coop: Context optimization for vision-language models. https://github.com/KaiyangZhou/CoOp, 2022. Accessed: October 2025.