EE5102/CS6302 - Advanced Topics in Machine Learning – Fall 2025

Assignment 2 — *Release Date: 2 October 2025*

Instructions: Please read the following instructions carefully and abide by while preparing your submissions:

- This is the second graded assignment of the course which counts towards 7% of your final assignment aggregate.
- We need only one submission per group. If you submit the report for your only task and/or not in the required format, we will assign you zero marks.
- This assignment is due by **Thursday, 16 October 2025 on LMS**.
- As a submission, you need to prepare and submit you deliverables according to the instructions in **Task 5 of this assignment**.
- **AI Usage Policy:** You are not allowed to use any generative AI model—including LLMs—to write any part of your PDF report. The language/text and analysis must be entirely your own, in the report.

  For the coding part, you ***may*** use public libraries, built-in functions, or even get help from an LLM if needed. That's acceptable.

  However, once you submit your assignment, you are fully responsible for everything in it—text and code both. If your submission overlaps significantly with another group's work, you cannot later claim that some part was LLM-generated. That will not be accepted as an excuse.

Overview & Motivation: In this programming assignment, you will explore Domain Adaptation (DA) and Domain Generalization (DG) – key challenges in making machine learning models robust to distribution shifts. We will start with fundamental DA techniques and progress to advanced DG methods, ultimately combining insights from both. By the end, you will have hands-on experience with:

- **Unsupervised DA:** Reducing distribution divergence between a labeled source domain and an unlabeled target domain, and understanding the trade-off between domain alignment and model discriminativeness.
- **DG:** Training on multiple source domains to generalize to unseen domains, using invariant learning (e.g., IRM), advanced techniques like sharpness aware minimization and robust optimization (e.g., Group DRO for worst-case performance).

Dataset Choices and Setup: You will choose one or more benchmark datasets for evaluating DA/DG algorithms. Recommended options include *PACS*, *Office-Home*, and *DomainNet*, summary in Table 1. These datasets provide multiple distinct domains for the same classification task, enabling both adaptation and generalization experiments.

Setup: It is recommended to use a **pretrained backbone** (e.g., ResNet-50 pretrained on ImageNet) for efficiency, fine-tuning it for domain tasks—this focuses the assignment on domain shift effects rather than learning low-level features. You may use any existing libraries or codebases (e.g., PyTorch, `torchvision` models, DomainBed `https://github.com/facebookresearch/DomainBed`, or other DA/DG toolkits) to implement algorithms, or write your own implementations. Using CLIP models with prompt learning will be helpful for the later parts. Ensure all experiments are reproducible: fix random seeds where possible and document any external code usage.

Evaluation Metrics: For each experiment, evaluate model performance on:

- **Target (unseen) domain accuracy:** the primary metric for domain adaptation/generalization success.
- **Source domain accuracy:** to observe if source performance drops when adapting/generalizing.
- **Domain-wise accuracy:** accuracy on each domain separately (when applicable) to see how performance varies.
- **Worst-group accuracy:** in multi-domain scenarios, the accuracy on the worst performing domain (reflects robustness to the most difficult domain shift).

Table 1: Summary of recommended datasets for DA/DG experiments.

Dataset	Domains	Classes	Total Images	Notes
PACS	4	7	9,991	Large style differences. Often used for DG by training on 3 domains and testing on the 4th.
Office-Home	4	65	∼15,500	Office object images in different domains (clipart vs. real photos). Used for both DA and DG.
DomainNet	6	345	∼0.6M	Very large and diverse; suitable for ambitious experiments.

Where relevant, also compute the average accuracy across domains. Use tables or plots to present these results clearly. If comparing methods, a summary table of each method's performance on each domain (and worst-case) is very helpful.

Now, proceed through the tasks below. Each task contains multiple parts; follow the instructions and gather the answers for analysis to be included in the report. Provide intuition and connect your findings to concepts from class and the literature. Citations to the provided papers and any additional sources you use are encouraged when explaining your reasoning.

Task 1: Unsupervised Domain Adaptation Basics: In this task, you will perform Unsupervised Domain Adaptation (UDA) from one labeled source domain to one unlabeled target domain. This will introduce the challenge of distribution shift and basic adaptation techniques. You may select any source and target domain of your choice, but **keep them consistent** throughout all Domain Adaptation experiments. Similarly, you may use any model as your feature extractor, but it should be consistent throughout all experiments.

Scenario: You have labeled examples in the source domain and **no labels in the target domain**, except for evaluation. The task is to train a classifier that performs well on the target domain without seeing target labels. Select a pair of domains from your chosen dataset from a standard DA benchmark. Train models with each method on the source domain(s) and evaluate on the target. **Ensure that all methods use the same backbone for fairness.**

1. **Source-Only Baseline**: Train a model only on the source domain data (standard empirical risk minimization on source labeled data). Evaluate its accuracy on the source test set and on the target domain test set. This represents the no-adaptation baseline.

   Analysis: How large is the drop in accuracy from source to target? This illustrates the effect of domain shift. Connect this observation to the theory: a large shift between source and target distributions can severely degrade performance. In your report, describe the nature of the domain shift (e.g., differences in style, lighting, etc.) and why the source-trained model struggles on target data.

2. **Domain-Alignment Based Adaptation**: Implement or use a library implementation to compare three domain adaptation approaches that enforce domain invariance in different ways: (1) Deep Alignment Network (DAN) – a statistical alignment method using kernels/MMD to match feature distributions; (2) Domain-Adversarial Neural Network (DANN) – an adversarial alignment method using a gradient reversal layer to confuse a domain discriminator; and (3) a class-aware alignment method such as CDAN or another approach that conditions alignment on class labels. Quantify any negative transfer effects – e.g., does aligning distributions cause the classifier to confuse rare classes more? One direction to investigate this could be by calculating the F1-score for the three rarest classes in the target domain and comparing it to the source-only baseline.

   Analysis: Does domain-alignment based training improve target accuracy compared to the source-only baseline? Discuss the results: did reducing the divergence between domains lead to better target performance as expected by theory? Conversely, note any impact on source-domain accuracy or class accuracy – for instance, did forcing alignment hurt the model's ability to capture discriminative features? Refer to the trade-off reported in literature: aligning domains too much can remove important class-specific features. Include any evidence of this trade-off from your experiment (e.g., confusion matrix changes or drop in certain class accuracy). Can you come up with a proxy-distance between source and target feature distributions for each method – this gives a sense of how well the domains were aligned. For example, you could use the error of a small classifier trained to distinguish between source and target features as a proxy for domain distance.

3. **Self-Training on Target (Pseudo-Labeling)**: As an alternative adaptation approach, perform self-training: use the source-trained model to predict labels on the target domain, then fine-tune the model on those pseudo-labeled target samples (optionally filtering by confidence). This leverages the target data structure without true labels. Evaluate the updated model on the target domain.

   Analysis: Compare this to DANN. Often, a simple self-training on target can significantly boost performance, especially with a strong pre-trained model. Did you observe an improvement and why? Discuss the potential reasons: for example, the model's initial predictions on target might be noisy, but incorporating even some correct pseudo-labels can refine the decision boundary for target data. Relate your findings to the observation that fine-tuning on pseudo-labeled target data alone can nearly match more complex UDA methods. What are the pitfalls of self-training (consider confirmation bias on incorrect labels)?

4. **Concept Shift (Mild):** Design experiments to simulate label/concept/semantic shift and rare-class scenarios and observe the impact on each method. For label shift: you can downsample or remove certain classes in the target domain or oversample some classes in source so that $P_s(Y) \neq P_t(Y)$. For a rare-class scenario: ensure one class is under-represented in the target. Measure how all these considered DA methods perform under these conditions. This will demonstrate when invariance breaks down. Quantify any negative transfer effects – e.g., does aligning distributions cause the classifier to confuse rare classes more?

For this task, you can get a number of visual results which might help your understanding:

- 2D t-SNE embeddings of feature representations for source vs. target samples (different colors) under each method. This will tell you whether the distributions overlapping (aligned) or separate? Color points by class to see if class clusters are preserved or mixed up.

- Confusion matrices on the target data for each method, especially in label shift scenarios – this can reveal if invariant models are misclassifying specific classes.
- A heatmap of class distribution shift (e.g., comparing source vs target label frequencies) and mark which classes had high or low accuracy under each method.

Task 2: DG via Invariant & Robust Learning Now assume we do not have any target data at training time – not even unlabeled. This is the DG scenario: the model must generalize to a completely unseen domain after training on multiple source domains.

Dataset: Choose a dataset with multiple domains such as PACS. Use multiple domains as sources and hold one domain out as the unseen target. For example, with PACS you may train on *Art*, *Cartoon*, *Photo* and test on *Sketch*. Remember to keep your choice of source and target domains consistent throughout your experimentation. A more rigorous analysis would rotate the held-out target, but this is not required for this assignment. All source domains have labels during training; the target domain is used only for evaluation. You may use any backbone of your choice, as long as its consistent across all tasks below.

1. **ERM Baseline:** Merge all source-domain data and train a standard classifier (ERM) on this combined set. Evaluate on the unseen target domain. Also evaluate on the source domains; report average source accuracy or each domain's accuracy) This serves as the baseline for DG.

   Analysis: How well does ERM perform on the held-out domain? It often sets a tough-to-beat benchmark. Note down if the model overfits to source idiosyncrasies e.g., high average source accuracy but much lower target accuracy. Discuss why a model trained on diverse source data might still fail to generalize—for instance, it might latch onto features present in all sources but absent in the target i.e. spurious correlations. You may highlight any evidence of this: if known spurious features exist in the data, or by analyzing feature importance across domains.

2. **Invariant Risk Minimization (IRM):** Implement IRMv1 or use an available implementation e.g., from the DomainBed codebase. IRM seeks a data representation such that the optimal classifier on top of it is the same for all training domains. In practice, IRM adds a regularization term that penalizes the variance of gradients of each domain's loss (ensuring that the classifier's optimality condition holds across domains). Train a model with IRM on the same multi-source setup as in previous part. Keep in mind that a model can easily reduce the IRM penalty to zero by learning a trivial, non-discriminative representation (e.g., mapping all inputs to a constant). To diagnose this, report not just the target accuracy but also the final value of the IRM penalty term. A near-zero penalty coupled with low accuracy is a clear sign of a collapsed, trivial solution.

   Analysis: Compare IRM's target performance to ERM. Did IRM improve generalization to the new domain? Or does it slightly improve worst-case domain performance? You will notice IRM can be hard to optimize and sometimes underperforms ERM. Report results and discuss. Check if IRM reduced source-domain training accuracy—a sign of trading off fit to enforce invariance. Did performance across source domains become more balanced? If IRM did not help, explain possible reasons such as optimization difficulty, trivial solutions, need for stronger penalties or more domains etc.

3. **Group DRO (Worst-Case Training):** Another approach is to explicitly optimize for worst-case performance across domains. Implement a Group DRO strategy; at each training step, compute the loss for each domain and re-weight or update more on the domain with highest loss. This ensures the model doesn't neglect the "hard" domain. Train a model with a group DRO objective on the source domains. Evaluate on the target domain.

   Analysis: Does focusing on worst-case i.e. hardest source domain improve performance on an unseen domain? Examine your results: for source domains, check if the performance gap between domains reduced i.e., the model balanced them better than ERM did. Also, report the worst-source-domain accuracy during training to confirm that Group DRO was indeed optimizing it. In your report, discuss how Group DRO relates to distributionally robust optimization theory – the model is optimizing a min-max objective to guard against worst-case domain shifts. Can you connect this to the concept of spurious correlations?
   Note: naively applying DRO on overparameterized networks requires strong regularization to truly improve worst-group generalization; if you needed to use measures like early stopping or stronger regularizers to get Group DRO to work well, then cite this insight, if applicable.

4. **Apply Sharpness-Aware Training (SAM) to ERM:** Standard training may converge to a sharp minimum (where the loss increases quickly if parameters change slightly), which can lead to poorer generalization under domain shift. Techniques like SAM add a perturbation step to find parameters that minimize loss in a neighborhood (i.e., a flatter solution).

   Keep the same the multi-source training setup; this time, train your model using a SAM optimizer (you can use an open-source implementation or write a wrapper that for each batch first computes gradients, then updates weights in the worst-direction and computes gradients again for the update). Essentially, you minimize a perturbed loss $L_{\text{SAM}}(\theta) = \max_{|\epsilon| \leq \rho} L(\theta + \epsilon)$, which penalizes sharpness. After training with SAM, evaluate

on the target domain.

Analysis: Compare SAM-trained model's target accuracy with the normal ERM model. Did SAM yield an improvement on the unseen domain? Often, SAM (or weight averaging methods) improve both in-domain and out-of-domain generalization. Check also the source domain performance – SAM sometimes maintains or even improves source accuracy while improving OOD performance, by finding a broadly effective solution. Discuss the results. If possible, provide an anecdotal measure of flatness: for example, you could plot training loss vs. parameter perturbation magnitude for the SAM vs ERM model to illustrate which is flatter. Summarize how such a flat minimum can act like a form of regularization that makes the model less sensitive to domain-specific perturbations. Connect to theory: a flatness-aware solution minimizes the worst-case loss in a neighborhood (analogous to a robust optimization). Also, reflect on any downsides: SAM requires more computation per step (gradient of gradient), and may need tuning of the sharpness radius ρ.

Finally, consider: could sharpness-aware training be combined with earlier methods (IRM or DRO)? There is ongoing research in mixing these ideas. You don't need to implement this, but briefly speculate: e.g., "Would applying SAM on top of IRM help mitigate IRM's optimization difficulties by avoiding sharp solutions that overfit to spurious correlations?" but how?

Create a summary table of target-domain accuracies (and worst-case source accuracies) for the methods you tried. Analyze:

- Which method performed best on the unseen target? Did any method significantly improve worst-case robustness among sources, and did that correlate with better unseen performance?
- If ERM matched or outperformed IRM and DRO, relate to recent results that well-tuned ERM can be competitive–look in the literature. If IRM or DRO helped, in what settings (e.g., target resembling a specific hard source)?
- It's known from prior benchmarks that IRM can be difficult to get right and sometimes fails to improve over ERM. You might find that IRM's performance is inconsistent (possibly even worse than ERM on PACS) – if so, discuss possible reasons (e.g. IRM might need very strong penalties or a simpler environment to succeed, and can converge to a bad solution if domains violate its assumptions). Provide an ablation or stability analysis: for IRM, try different penalty weights to see if it diverges or finds trivial solutions.
- Measuring Flatness: Can we somehow measure the flatness of our converged solution? Think about it – usually, cross-domain flatness can be quantified by how much a perturbation that worsens one domain's loss also worsens others – ideally, a truly domain-general solution has a broad region of low loss that covers all domains.
- Visualization of Loss Landscapes: Can we visualize the loss surface to see sharp vs flat minima, by picking a 2D subspace for visualization?
- Finally, consider the concept of domain invariance vs. discriminability: which approach seems more suitable given your data and why? Provide reasoning backed by your experiments.
- A broader research question guiding your analysis should be: *Can We Engineer Cross-Domain Flatness and Invariance for Domain Generalization?*

Task 3: Are Prompts a Stable Control Knob for DA/DG with CLIP? The final task is more open-ended and research-focused. You will examine a state-of-the-art approach that leverages pre-trained vision-language models (like CLIP) and addresses domain shifts via prompt learning and gradient alignment. The goal is to understand how conceptual advances can combine the best of both worlds: strong pre-trained features + domain-specific adaptation while ensuring consistency across domains.

CLIP provides a powerful feature space aligned with human concepts. Recent works propose prompt learning for DA/DG, where you tune text prompts to adapt CLIP to new domains. However, prompt tuning can be brittle: a slight prompt change can greatly affect predictions.

1. **CLIP Zero-Shot vs Fine-Tuned on Domains:** First, evaluate the out-of-the-box capability of **CLIP** on your chosen dataset. Using a pretrained CLIP model, perform zero-shot classification on different domains. For example, for PACS, use CLIP's text encoder with prompts such as *"a photo of a {class}"* for each class. Since CLIP is trained on the word "photo," you might also try alternative prompts like *"a sketch of a {class}"* for the sketch domain, etc. Measure CLIP's accuracy per domain without any fine-tuning. Next, try a simple fine-tuning approach: treat CLIP's image encoder as a feature extractor and train a linear classifier on the source domains (or apply small prompt tuning if feasible). Evaluate this model on the target domain.

   Analysis: How well does CLIP zero-shot perform across domains? Often, CLIP performs quite well on photographic domains but may struggle on very stylized ones (since its training distribution had many photos). Does fine-tuning on source make CLIP lose some of its generality (overfitting to source domain)? This can happen – discuss the adaptation vs. generality trade-off. You might find that CLIP's zero-shot is a strong baseline that is hard to beat on some domains due to its broad training.

2. **Prompt-Learning with CLIP:** Now simulate a domain adaptation scenario with CLIP. Suppose you have one source domain and one target domain (unlabeled). Implement a prompt-learning adaptation: initialize a set of learnable prompt vectors for CLIP's text input (instead of using the hand-crafted *"a photo of a {class}"*). Train these prompt parameters such that CLIP's predictions are accurate on the source domain and also account for target domain data. You can do this by:

   (a) Using source labeled data to push the prompt to classify correctly; just like training a classifier, but only optimizing the text prompt embedding while keeping CLIP frozen.

   (b) Incorporating target data in training via an unsupervised loss. For example, you might enforce that the prompt does not produce highly conflicting predictions on target data, or use a consistency regularization, or even use pseudo-labels for target.

   There are two very famous prompt learning schemes: CoOp (Context Optimization) and CoCoOp (Conditional CoOp).

   Analysis: Report the target performance and compare it to prior approaches. Did prompt learning effectively balance domain-specific and domain-invariant information? Discuss why prompt tuning might be advantageous for domain shift. Also consider the challenges (hint: prompts can overfit or be brittle). Ground your discussion in the context of the literature – for example, prompt tuning is more lightweight and robust than fine-tuning the full model, but a naive prompt approach still faces brittleness, hence the need for gradient alignment.

3. **Gradient Conflict and Alignment:** To understand gradient alignment, perform a smaller analysis experiment. Take a case of two source domains or one source + target pseudo-labeled, that you train simultaneously. At a given training step e.g., after initialization or a few epochs, compute the gradient of the loss for domain A and the gradient for domain B with respect to the model parameters; you can focus on the prompt parameters or last-layer weights to make this tractable. Calculate the angle or cosine similarity between these gradient vectors. Do this at several points during training. Often, when there is a conflict; one domain's improvement hurts the other, the gradients will be pointing in very different (sometimes opposite) directions (cosine similarity ≤ 0). Now, conceptually, a gradient alignment algorithm like PGA will try to increase the cosine similarity between domain gradients – i.e., update in a direction that is more agreeable to all domains. There are known techniques such as GradCos or PCGrad that project gradients to avoid interference.

   Analysis: Report the degree of conflict you observed: were the gradients from different domains often conflicting or mostly aligned? Identify moments of high conflict – these likely correspond to the model encountering features that help one domain but not the other. Discuss how gradient alignment could improve training here. For instance, if domain A and B gradients conflict, aligning them by modifying their directions slightly or re-weighting, can find a compromise that improves both.

   Can you come up with your own approach to align the gradients whenever there is a conflict? This part

is exploratory; in your report, explain in your own words why aligning gradients leads to learning domain-invariant features.

4. **Open-Set and Generalization Analysis:** One concern is how prompt tuning affects open-set performance. CLIP is naturally an open-set recognizer i.e. zero-shot for any class via text. When you tune prompts on a closed set of classes, you might inadvertently reduce the model's ability to recognize new classes or detect out-of-distribution samples. To test this, design an open-set experiment: for instance, train or tune prompts for only a subset of classes (say 80% of PACS classes) and then evaluate on both seen and unseen classes.

   Analysis: Compare the zero-shot CLIP vs tuned prompt in terms of recognizing the unseen classes. You can measure calibration or openness by metrics like the false-positive rate on unseen classes at a given threshold, or use maximum softmax probability (MSP), confidence entropy, etc. It's hypothesized that a tuned prompt will be highly confident on its trained classes but might give arbitrary (and high) scores to an out-of-distribution input, thus hurting open-set detection. Also, compute the cosine similarity of prompt embeddings across domains: if you learn a prompt for source and one for target with a few-shot target tuning, how similar are they? If there is a large difference, what does it indicate?

Task 4: Synthesis of results & Report Writing Guidelines Now that you have conducted the experiments, the final task is to synthesize your findings into a coherent analysis and present it in a professional manner.

Deliverables:

- GitHub Repo: containing code (with documentation), any saved models or data subsets, and a README explaining how to run your analysis.
- PDF Report: ICML style, 6-10 pages including figures, addressing all points above. Treat it as a professional paper – clarity, structure, and correctness are key. Guidelines are given below. If you have additional results, you may add them in appendix after 10 pages. Write clearly and succinctly. Use bullet points or subheadings only if necessary.

Instructions:

- **Organize Your Code and Results:** Ensure your scripts for each task are clean, well-documented, and placed in the GitHub repo. Include instructions or scripts to install requirements and run the experiments. If some experiments are computationally heavy, provide saved outputs (e.g. learned model weights, logged metrics, or sample images) so the TAs/instructor can verify results without rerunning everything. The repository should be structured logically (perhaps a folder for Task1, Task2, etc., each with code and maybe a short README of its own).
- **Prepare Figures and Tables:** From Tasks 1–4, you likely have several plots, images, and metrics. Select the most meaningful ones to include in your report. DO NOT ADD JUST RANDOM FIGURES. Use clear captions and refer to them in the text. Ensure every figure is legible (use sufficient resolution as needed) and every table is properly labeled.
- **Writing the Report:** The report should roughly include:
  - **Abstract:** A short summary of what you did and key findings.
  - **Introduction:** Introduce the problem of DA/DG. State the objectives of this assignment that you will explore. Motivate why this study is important. You can briefly preview your approach and findings. Cite relevant background in proper academic citation format.
  - **Methodology/Experiments:** This can be structured by your tasks.
  - **Results:** Present the findings for each experiment, ideally intertwining the quantitative results with analysis. This is where you include those figures and tables.
  - **Discussion:** This is a crucial part to demonstrate deep reflection. Also discuss the interplay of architecture and data-driven biases and potential future designs.
  - **Conclusion:** A short paragraph wrapping up.
  - **Citations:** Throughout the report, if needed, cite sources in academic style. Make sure to cite any claim that is not your own result. A References section should list all cited works. Compare your findings with known literature to show deep understanding.
  - Finally, reflect on the process in a brief note (could be in the report discussion or a separate markdown in the repo): What surprised you? Did any results conflict with your expectations or published results?

References

[1] DAN Paper: Long, Mingsheng, et al. "Learning transferable features with deep adaptation networks." International conference on machine learning. PMLR, 2015.
https://proceedings.mlr.press/v37/long15

[2] DANN Paper: Ganin, Yaroslav, and Victor Lempitsky. "Unsupervised domain adaptation by backpropagation." International conference on machine learning. PMLR, 2015.
https://proceedings.mlr.press/v37/ganin15

[3] CDAN Paper: Long, Mingsheng, et al. "Conditional adversarial domain adaptation." Advances in neural information processing systems 31 (2018). https://arxiv.org/abs/1705.10667

[4] Cross-Domain Contrastive Learning Paper: Wang, Rui, et al. "Cross-domain contrastive learning for unsupervised domain adaptation." IEEE Transactions on Multimedia 25 (2022): 1665-1673. https://arxiv.org/abs/2106.05528

[5] Domain Adaptation via Prompt Learning Paper: Ge, Chunjiang, et al. "Domain adaptation via prompt learning." IEEE Transactions on Neural Networks and Learning Systems (2023). https://arxiv.org/abs/2202.06687

[6] PGA Paper: Phan, Viet Hoang, et al. "Enhancing domain adaptation through prompt gradient alignment." Advances in Neural Information Processing Systems 37 (2024): 45518-45551. https://arxiv.org/abs/2406.09353

[7] IRM Paper: Arjovsky, Martin, et al. "Invariant risk minimization." arXiv preprint arXiv:1907.02893 (2019). https://arxiv.org/abs/1907.02893

[8] G-DRO Paper: Sagawa, Shiori, et al. "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization." arXiv preprint arXiv:1911.08731 (2019). https://arxiv.org/abs/1911.08731

[9] DRO At Feature Level Paper: Li, Chenming, et al. "Cross contrasting feature perturbation for domain generalization." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023. https://arxiv.org/abs/2307.12502

[10] SAM (SWAD) Paper: Cha, Junbum, et al. "Swad: Domain generalization by seeking flat minima." Advances in Neural Information Processing Systems 34 (2021): 22405-22418. https://arxiv.org/abs/2102.08604

[11] SAM (Shaping Loss Landscapes) Paper: Li, Aodi, et al. "Seeking consistent flat minima for better domain generalization via refining loss landscapes." Proceedings of the Computer Vision and Pattern Recognition Conference. 2025. https://openaccess.thecvf.com/content/CVPR2025/html/Li_Seeking_Consistent_Flat_Minima_for_Better_Domain_Generalization_via_Refining_CVPR_2025_paper.html

[12] SAM Paper: Luo, Haocheng, et al. "Explicit eigenvalue regularization improves sharpness-aware minimization." Advances in Neural Information Processing Systems 37 (2024): 4424-4453. https://proceedings.neurips.cc/paper_files/paper/2024/hash/0845e3f7447ff9e36a26033bb7334674-Abstract-Conference.html