Domain Generalization on Small Models

Ben Monnig
Georgia Institute of Technology
bmonnig3@gatech.edu

Rachel Yuh
Georgia Institute of Technology
ryuh6@gatech.edu

Giovanni Hsu
Georgia Institute of Technology
ghsu31@gatech.edu

Joseph Kiezulas
Georgia Institute of Technology
jkiezulas@gatech.edu

Abstract

Many practical vision systems rely on small, resource-constrained models, yet these models often struggle when the visual domain at test time differs from the one seen during training. We investigate whether lightweight domain-generalization strategies (such as feature-statistics mixing and late-layer feature alignment) can improve the robustness of compact neural networks under the standard leave-one-domain-out setting. Our goal is to determine whether methods originally developed for large architectures continue to offer benefits when applied to much smaller models without increasing their computational footprint.

Across PACS, Office-Home, and VLCS, we find little to no consistent improvement over a small CNN trained with standard empirical risk minimization. On PACS, MixStyle improves mean target-domain accuracy from 28.7% to 29.3% over ERM, while Late Feature Alignment (LFA) slightly improves VLCS accuracy (51.6% to 52.2%) but degrades PACS performance (28.7% to 25.5%). While some techniques yield minor gains on specific domains, their effects are unstable and do not generalize reliably across datasets or held-out splits. These results suggest that many current DG strategies do not readily transfer to low-capacity architectures and highlight the need for approaches explicitly designed with small-model limitations in mind.

1 Introduction

Domain Generalization (DG) develops models that can perform well on data from previously unseen domains. Although many recent studies have shown strong results using large, high-capacity architectures, practical systems often rely on much smaller models due to constraints on computation, memory, and latency. These compact networks are more prone to overfitting and struggle to learn representations that remain stable across differing visual conditions, making DG especially challenging in this case.

In this project, we ask the following question: Can a small convolutional network be equipped with simple, compute-efficient DG mechanisms that reliably improve performance on unseen domains under standard leave-one-domain-out evaluation? We define success as achieving higher target domain accuracy than a size-matched ERM baseline on PACS, Office-Home, and VLCS, without increasing parameter count or inference-time FLOPs. This matters for applications such as mobile vision, embedded perception, and robotics, where robustness to domain shift is desirable but large models are impractical. If successful, our findings would offer concrete, low-cost strategies for improving robustness in small models and inform the design of reliable on-device perception systems.

To investigate this question, we evaluate lightweight DG techniques (including feature regularization, style-based mixing, and late-layer alignment) adapted from methods traditionally developed for larger

architectures. Our models take RGB images from multiple source domains (photos, sketch, cartoons) and predict categorical object labels. Training uses all but one domain, and the remaining domain serves as the target test split, providing a clear measure of how well each technique handles unseen distribution shifts.

By analyzing these methods within a unified small-model framework, we aim to understand which DG strategies remain effective under strict capacity limits and where they fail. These insights highlight practical pathways toward deploying compact, domain-robust models in resource-constrained settings.

Success is defined as achieving higher target domain accuracy than a size-matched ERM baseline under the leave-one-domain-out protocol, without increasing parameter count.

2 Related Work

Domain Generalization (DG) focuses on training models that generalize to unseen domains without ever seeing the target domain data. Foundation methods have explored feature invariance and different meta-learning strategies. For example, Invariant Risk Minimization tries to find representations where a single classifier works well across all of the source domains [Arjovsky et al., 2020]. Meta-Learning approaches, like MLDG, simulate the domain shifts during training by splitting each batch into train and test domains. This leads the model to make updates that improve performance not just on the training domain but also on other unseen domains [Li et al., 2017]. Other methods, like Representation Self-Challenging (RSC), suppress the most dominant features during training so the model is forced to learn the more nuanced, generalizable features [Huang et al., 2020]. These different strategies have mostly been tested on large, overparameterized networks, and it is unclear how well they will work on smaller models that can be easily over fitted.

Recent work has explored lightweight strategies that are helpful for smaller models. MixStyle mixes feature statistics from different domains to generate synthetic variations, helping models generalize better [Zhou et al., 2021]. NormAUG mixes up features through alternative normalization statistics to increase generalization in smaller networks [Qi et al., 2024]. Studies also analyze the architecture, such as Angarano et al., show that backbone choice highly affects DG performance, and that small networks often under perform on their own [Angarano et al., 2023]. To address this issue, ALOFT introduces a compact MLP network with low-frequency perturbations, showing that the architectural design can improve DG for smaller models [Guo et al., 2023]. Similarly, RRLD uses self-distillation and augmentations to improve generalization in small vision transformers [Singh and Jayavelu, 2024]

Benchmarking studies, such as DomainBed, show that even a simple empirical risk minimization (ERM) can show competitive results when tuned well Gulrajani and Lopez-Paz [2020].
Our project will build on these insights by evaluating classical and lightweight DG techniques on small CNNs. We plan on combining previous methods and test them systematically on standard benchmarks to identify strategies that improve domain generalization in resource efficient models.

Table 1: Comparison of representative DG methods and how our work differs.

Method Family	**Assumptions**	**Requirements**	**Gap We Addresses**
IRM / invariance	Invariant features exist across domains	High compute; large backbones	Hard to satisfy with small models
Meta-learning (MLDG)	Episodic simulation approximates domain shifts	High compute; episodic training; large models	Too expensive for compact architectures
Feature suppression (RSC)	Dominant feature maps are informative	Medium compute; mid-sized CNNs	Weak effect with low-dimensional feature maps
MixStyle	Style statistics are rich enough to mix	Low compute; easy to integrate	Unclear behavior on tiny CNNs
Our work	Lightweight DG for small models	Low–medium compute; small CNNs	Which DG ideas still help under capacity limits

Existing DG methods differ widely in their assumptions, computational demands, and dependence on high-capacity backbones. Approaches such as IRM and MLDG require strong invariance assumptions or episodic domain simulation, which become difficult to satisfy in compact architectures with

limited feature diversity. Regularization-based methods such as RSC assume the presence of rich, high-dimensional feature maps, and lightweight strategies such as MixStyle typically rely on mid- or high-capacity backbones where intermediate feature statistics are expressive.

In contrast, our work focuses explicitly on DG on smaller models. Rather than implementing the full set of classical DG algorithms, we draw inspiration from them and evaluate three practical, lightweight variants suited to compact networks: (1) ERM on a small CNN, (2) the same CNN augmented with MixStyle, and (3) a new late-layer feature-alignment loss designed to impose cross-domain consistency without adding significant computation.

3 Method/Approach

3.1 Hypotheses

H1. SmallCNN Baseline Capacity. We expect ERM with SmallCNN to underperform standard DG results because the backbone is intentionally capacity-limited ($\sim$ 30k parameters). This highlights the challenge of DG in the area of small models.

H2. MixStyle Improves Style Robustness. We expect MixStyle to outperform ERM. By perturbing feature statistics, MixStyle exposes the model to synthetic style shifts, improving robustness to unseen domains.

H3. LFA Achieves the Best Cross-Domain Alignment. We hypothesize that aligning class-conditional feature means at the final convolutional layer improves DG the most. Late alignment avoids over-constraining early features while enforcing domain-invariant class structure.

3.2 Model Architecture

All experiments use a single lightweight backbone, **SmallCNN**, designed to evaluate DG on small models. The network consists of three convolutional blocks with BatchNorm and ReLU activations, each followed by 2×2 max pooling. An adaptive average pooling layer reduces the spatial dimension before a final linear classifier.

- Block 1: Conv(3→16, 3×3), BatchNorm, ReLU, MaxPool
- Block 2: Conv(16→32, 3×3), BatchNorm, ReLU, MaxPool
- Block 3: Conv(32→64, 3×3), BatchNorm, ReLU, MaxPool
- Head: AdaptiveAvgPool → FC(64→ C)

MixStyle layers (if enabled) are inserted after each convolutional block. The full model contains approximately **30k parameters** (depending on # classes).

3.3 Proposed Method: Late Feature Alignment (LFA)

Late Feature Alignment (LFA) is a lightweight alignment module operating on the output of the third convolutional block. For each class c and domain d, let $\mu_d^{(c)}$ denote the mean feature vector over samples from (c, d). LFA minimizes the pairwise distance between these class-conditional means:

$$\mathcal{L}_{\text{align}} = \sum_{c} \sum_{d_i \neq d_j} \left\| \mu_{d_i}^{(c)} - \mu_{d_j}^{(c)} \right\|_2^2 .$$

The full training loss is:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \, \mathcal{L}_{\text{align}}.$$

Why it should work. Class-conditional alignment encourages domains to share a common feature structure for each class, reducing domain shift. Performing alignment late retains domain-specific low-level features while enforcing high-level semantic consistency.

3.4 Baselines and Metrics

Baselines. All methods are evaluated under the same backbone and leave-one-domain-out method: for each dataset, we train on three source domains and test on the held-out target domain. We consider the following training strategies:

- **ERM** (Empirical Risk Minimization): A standard supervised baseline that groups all source domain samples and minimizes cross-entropy, without any explicit domain generalization techniques. The model can never see the target domain data while training.
- **MixStyle**: A lightweight domain generalization method that adds MixStyle layers after convolutional blocks. During training, these layers randomly mix simple feature statistics (per-channel means and variances) between samples from different domains. This creates new domains and encourages the model to focus less on domain specific style.

Metrics. For each dataset, we use the leave-one-domain-out method. We set one domain d_{tgt} as the target, train on the remaining domains, and select the checkpoint with the best mean validation accuracy on the training domains. We then evaluate this checkpoint on the target domain.

Let $\text{Acc}_{d_{\text{tgt}}}$ be the test accuracy on the left out domain and $\text{Val}_{d_{\text{tgt}}}$ the corresponding mean validation accuracy on the source domains. Across all choices of target domain, we calculate:

$$\overline{\text{Acc}} = \frac{1}{|\mathcal{D}|} \sum_{d_{\text{tgt}} \in \mathcal{D}} \text{Acc}_{d_{\text{tgt}}},$$

as the mean target domain accuracy. To capture how much performance drops when moving from validation (seen domains) to test (unseen domain), we also compute the domain generalization gap:

$$\text{DG Gap} = \frac{1}{|\mathcal{D}|} \sum_{d_{\text{tgt}} \in \mathcal{D}} \left(\text{Val}_{d_{\text{tgt}}} - \text{Acc}_{d_{\text{tgt}}}\right).$$

In Table 2, each entry is reported as $\overline{\text{Acc}}$ (DG Gap), so higher accuracy and smaller DG Gap are better.

3.5 Training Procedure

All methods use the same training configuration:

- Optimizer: Adam
- Batch size: 32
- Epochs: 10
- Scheduler: None
- Model selection: best validation accuracy
- Hardware: NVIDIA T4 GPU in Google Colab
- Framework: PyTorch 2.2

Data augmentation follows DomainBed-style transformations (RandomResizedCrop, RandomHorizontalFlip, Normalize).

Pseudocode.

```
for each target domain d_t:
    load source domains D \ {d_t}
    for epoch in 1..10:
        train_one_epoch(model, train_loader)
        evaluate(model, val_loader)
    load best checkpoint
    test on domain d_t
```

3.6 Hyperparameter Search

We perform a per-domain Optuna search for each method with the following search spaces:

- Learning rate: log-uniform in $[10^{-5}, 10^{-2}]$
- MixStyle: $p \in [0, 1]$, $\alpha \in [0.05, 5]$
- LFA: $\lambda \in [10^{-4}, 10^{-1}]$

Each of the 50 trials trains for 5 epochs, selects the best hyperparameters using validation accuracy and evaluation on the target domain.

4 Data

Our experiments use standard domain-generalization benchmarks (PACS, Office-Home, and VLCS) each containing 4 visual domains with consistent label spaces.

4.1 Datasets

PACS PACS [Xu et al., 2019] is a domain generalization dataset where each domain corresponds to a different visual style of image: *Photo*, *Art*, *Cartoon*, and *Sketch*. The label space consists of 7 object classes (dog, elephant, giraffe, guitar, horse, house, person). The domains are imbalanced with *Sketch*, for example, has more images than the other three combined domains. The primary domain shift is stylistic: all domains depict the same set of object categories, typically as centered single objects, but rendered with different drawing/photographic styles.

Office-Home Office-Home [Venkateswara et al., 2017] contains everyday office and home object categories across four domains: *Art*, *Clipart*, *Product*, and *Real World*. The label space comprises 65 categories (e.g., keyboard, printer, bike, chair). The dataset has approximately $15{,}000$ images with unbalanced classes: some categories and domains have substantially more samples than others.

VLCS VLCS [Torralba and Efros, 2011] is constructed by combining subsets of four existing photographic datasets, which are the different domains: *VOC2007 (V)*, *LabelMe (L)*, *Caltech101 (C)*, and *SUN09 (S)*. All images are natural photos, so domain shift arises from dataset-specific collection pipelines, scene types, and backgrounds rather than stylization. The label space consists of the 5 classes shared across all four source datasets: bird, car, chair, dog, and person.

All datasets are only RGB image modalities, stored in common formats (JPG/PNG). Labels are object classes, and the label space is shared across domains. Because these datasets were collected from mixed sources, they show biases such as class imbalance, uneven per-domain representation, and domain-specific artifact patterns.

4.2 Train/Validation/Test Construction

We used the leave-one-domain-out evaluation scheme usually used in work on domain generalization. For a dataset with domain set $\mathcal{D} = \{d_1, \ldots, d_K\}$, we choose one domain $d_{\text{target}} \in \mathcal{D}$ as the target, and use the remaining domains as sources:

$$\mathcal{D}_{\text{source}} = \mathcal{D} \setminus \{d_{\text{target}}\}.$$

All target samples are used for the test set in that experiment. For example, in PACS with *Sketch* as the target, there are roughy $4{,}000$ test images and $6{,}000$ training images from the remaining three domains.

Train/validation split within sources. Within the source samples, we create the train and validation sets by a fixed random split:

- We shuffle the indices of the source samples, using a fixed NumPy random seed for reproducibility.
- We use 10% of the source samples as validation and the remaining 90% as training.

4.3 Normalization and augmentation

All models share the same pixel level pre-processing pipeline. This makes comparisons between methods and architectures fair and avoids confounding the results with different data augmentation policies.

Image size and normalization. All images are resized to a square resolution of 224×224.

- During training, images are first resized slightly larger than the target resolution, then a random 224×224 crop is taken, followed by a random horizontal flip with probability 0.5.
- For validation and test, images are resized in the same way but center-cropped instead of randomly cropped, and no random flips are applied.

After spatial processing, all images are converted to tensors and normalized using the standard ImageNet statistics, with mean $\mu = (0.485, 0.456, 0.406)$ and standard deviation $\sigma = (0.229, 0.224, 0.225)$.

Summary In summary, all three benchmarks are used in their standard multi-domain classification form, leave-one-domain-out train/validation/test splits, and a shared normalization and augmentation policy across architectures. This setup is designed to isolate the effect of domain generalization techniques on the small models.

5 Experiments and Results

We conduct a thorough empirical study across three standard domain-generalization benchmarks: PACS, Office-Home, and VLCS. All experiments follow the leave-one-domain-out protocol, where models are trained on all but one domain and evaluated on the held-out target domain. We then report test accuracy on the held-out target domain. For each method and dataset, Table 2 summarizes the mean target domain accuracy across all choices of held-out domain, together with the corresponding domain generalization gap (DG Gap) in parentheses.

Baselines. We compare three small-model variants that share the same lightweight CNN backbone and training pipeline:

- **Small CNN (ERM)**: a classical baseline that trains the CNN with standard empirical risk minimization on source domains.
- **Small CNN + MixStyle**: the same CNN with MixStyle layers after convolutional blocks. During training, these layers randomly mix simple feature statistics (per-channel means and variances) between samples from different domains.
- **Small CNN + LFA**: our Late Feature Alignment (LFA) variant, which adds a loss on per-class, per-domain feature centroids at a late layer to encourage class-consistent representations across domains.

MixStyle probability, mixing strength, and the LFA weight λ_{align} are tuned on source-domain validation performance for a fair comparison.

Ablations for H1–H3. We designed our experiments to test the three hypotheses in Section 3.1.

H1 (SmallCNN baseline capacity). We use the SmallCNN trained with ERM as our base model and compare its target domain accuracies on PACS, Office-Home, and VLCS to typical results reported for larger backbones on the same benchmarks in prior DG work. This will serve as the reference point for the remaining models.

H2 (MixStyle improves style robustness). To see if perturbing feature statistics improves generalization, we compare Small CNN (ERM) and Small CNN + MixStyle under the same training. MixStyle layers are put in after the convolutional blocks, and we sweep their activation probability and mixing strength using the source domain validation performance to select the best configuration. We then analyze changes in mean target domain accuracy and DG Gap across all leave-one-domain-out splits.

H3 (LFA has the best cross-domain alignment). To test if Late Feature Alignment has the strongest DG improvements, we introduce Small CNN + LFA, which adds a loss on class conditional feature means at a late convolutional layer. We compare Small CNN + LFA to both ERM and MixStyle in relation to mean target domain accuracy and DG Gap, as well as per-domain behavior in Tables 3–5.

Quantitative Results. Table 2 shows that all three of the small models achieve relatively low absolute accuracies compared to large backbone results in the literature. Within the small models, the effects of MixStyle and LFA are mixed and depend on the dataset:

- **PACS.** Small CNN + MixStyle slightly improves mean target domain accuracy over ERM (28.7 $\rightarrow$ 29.3) while reducing DG Gap (20.6 $\rightarrow$ 15.8). However, Small CNN + LFA under performs ERM (25.51) and shows the largest DG Gap (24.26), indicating over constraint on this dataset. Per-domain results in Table 3 reveal that MixStyle helps on *Photo*, *Art*, and *Cartoon*, but hurts *Sketch*. LFA amplifies this loss in accuracy more on *Cartoon* and *Sketch*.
- **Office-Home.** All methods are clustered in a 7–12% range (Table 4), and the mean accuracies in Table 2 differ by less than 1 percent. MixStyle slightly reduces mean accuracy relative to ERM but also reduces DG Gap, while LFA is roughly on par with ERM in both mean accuracy and DG Gap. None of the variants show a clear winner across all domains.
- **VLCS.** Small CNN + LFA has the best mean target-domain accuracy (51.59 $\rightarrow$ 52.21) and a slightly smaller DG Gap (5.77 $\rightarrow$ 5.29) compared to ERM. Per-domain results in Table 5 show that LFA improves *LabelMe*, *VOC2007*, and *SUN* but hurts *Caltech*. MixStyle only clearly helps on *SUN* and hurts *Caltech*, *LabelMe*, and *VOC2007*.

Overall, there is no best small model variant. MixStyle and LFA provide some gains on certain domains and datasets but can also hurt performance on others. Improvements are usually in the range of 0.5–3 percent and come with changes in DG Gap, showing a trade off between robustness and an over constraint in the small models.

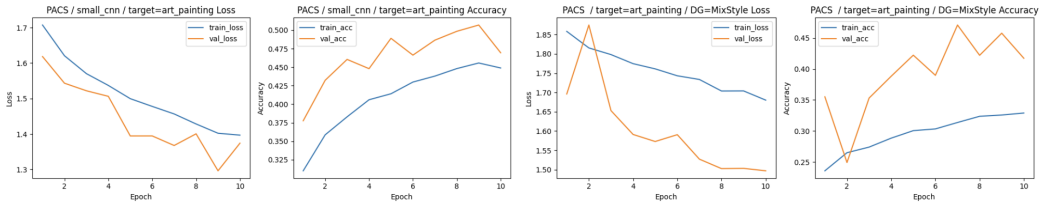

Figure 1: Training and validation curves on the target domain **Art Painting**. Left: loss/accuracy without MixStyle; Right: with MixStyle. MixStyle reduces overfitting and stabilizes training.

Table 2: Average target domain accuracy (%) for PACS, Office-Home, and VLCS under leave-one-domain-out evaluation. DG gap in parentheses.

Method	PACS	Office-Home	VLCS
Small CNN (ERM)	28.7 (20.6)	**10.4 (6.1)**	51.59 (5.77)
Small CNN + MixStyle	**29.3 (15.8)**	9.4 (3.6)	49.96 (6.14)
Small CNN + LFA	25.51 (24.26)	9.76 (6.62)	**52.21 (5.29)**

Table 3: target domain accuracy (%) on PACS (leave-one-domain-out).

Method	Photo	Art	Cartoon	Sketch
Small CNN (ERM)	35.97	22.95	27.70	**28.01**
Small CNN + MixStyle	**39.92**	**24.27**	**29.27**	23.89
Small CNN + LFA	39.39	22.41	19.89	20.35

Table 4: target domain accuracy (%) on Office-Home (leave-one-domain-out).

Method	Art	Clipart	Product	Real World
Small CNN (ERM)	**9.68**	**8.16**	**11.26**	**12.49**
Small CNN + MixStyle	8.18	7.66	9.44	12.23
Small CNN + LFA	9.44	8.07	10.88	10.65

Table 5: target domain accuracy (%) on VLCS (leave-one-domain-out).

Method	Caltech	LabelMe	VOC2007	SUN
Small CNN (ERM)	**74.51**	51.96	45.52	38.38
Small CNN + MixStyle	70.56	39.83	42.05	**47.41**
Small CNN + LFA	67.15	**54.74**	**45.78**	43.16

Qualitative Analysis. Training curves indicate that ERM converges quickly and often achieves higher validation accuracy, but with larger DG Gaps on the target domain. MixStyle typically flattens the validation curve and narrows the gap on PACS, while LFA can slow convergence and sometimes underfit. Visual inspection of success and failure cases from the target domains shows that errors frequently coincide with strong stylistic shifts (texture loss in sketches, unusual color palettes in clipart).

Hypothesis Evaluation.

- **H1** predicted that ERM-trained SmallCNNs would underperform due to limited capacity. Our experiments support this: accuracies are substantially lower than typical DG results reported for large backbones.
- **H2** predicted that MixStyle would improve robustness to style shift. Our results partially refute this: MixStyle improves performance on some PACS domains but hurts performance on others.
- **H3** predicted that Late Feature Alignment (LFA) would provide the strongest improvements. This hypothesis is not supported: LFA does not consistently outperform ERM or MixStyle and sometimes harms performance, suggesting that alignment over constrains small models.

Error Analysis. We observed that misclassifications concentrated on extreme style divergences, for example, sketches lacking texture. This suggests that small models cannot preserve domain-invariant features when low-level cues disappear. Future small-model DG should incorporate texture-preserving augmentations or mid-level contrastive objectives.

6 Conclusion

We conducted a systematic evaluation of lightweight DG techniques for compact CNNs under the leave-one-domain-out setting. While MixStyle and our proposed Late Feature Alignment regularizer modestly improved performance on isolated domains, none of the methods produced consistent gains over the ERM baseline. Our findings suggest that many DG methods, originally designed for large backbones, do not transfer reliably to capacity-limited models.

Limitations. Our models are intentionally small, which may limit the expressive power required for successful DG. Training for only 10 epochs, although consistent across methods, may underfit certain domains. Hyperparameter searches were constrained by compute, and we did not explore more advanced augmentations or pretraining strategies.

Future work. Future directions include designing DG methods that explicitly account for low capacity, incorporating information bottlenecks or parameter-efficient adapters, exploring feature-space contrastive objectives, and evaluating small transformers or mobile backbones under the same constraints.

7 Team Contributions

Table 6: Author Contributions

Person	Contributions
Ben Monnig	Implemented training pipeline; integrated MixStyle module; ran ERM and MixStyle experiments; performed validation and tuning for PACS; created quantitative result tables and plots; contributed to experimental design; analyzed performance trends; conducted debugging and reproducibility checks; wrote portions of Method and Results sections; edited the final report draft.
Rachel Yuh	Researched and built dataset pipelines; implemented dataset preprocessing and loaders; wrote major portions of the Abstract, Introduction, and Results sections; ran LFA experiments; managed Office-Home data processing; validated model checkpoints; generated qualitative visualizations; analyzed dataset biases and domain gaps; contributed to Related Work; reviewed and edited the final report draft.
Giovanni Hsu	Implemented core training and evaluation logic; developed the Late Feature Alignment (LFA) loss; ran LFA experiments; analyzed domain generalization metrics and per-domain behavior; produced learning curves and qualitative examples; debugged and profiled model stability issues; wrote Method, Experiments, and Discussion sections; prepared figures and equations; refined report organization and clarity; edited the final report draft.
Joseph Kiezulas	Implemented data augmentation and evaluation utilities; ran ERM and MixStyle experiments; performed H1–H3 ablation studies; automated result aggregation and metric computation; analyzed error and failure cases; tuned LFA regularization parameters; maintained scripts for reproducibility; edited Related Work and Limitations sections; generated plots and tables; edited the final report draft.

References

Simone Angarano, Mauro Martini, Francesco Salvetti, Vittorio Mazzia, and Marcello Chiaberge. Back-to-bones: Rediscovering the role of backbones in domain generalization, 2023. URL https://arxiv.org/abs/2209.01121.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2020. URL https://arxiv.org/abs/1907.02893.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization, 2020. URL https://arxiv.org/abs/2007.01434.

Jintao Guo, Na Wang, Lei Qi, and Yinghuan Shi. Aloft: A lightweight mlp-like architecture with dynamic low-frequency transform for domain generalization, 2023. URL https://arxiv.org/abs/2303.11674.

Zeyi Huang, Haohan Wang, Eric P. Xing, and Dong Huang. Self-challenging improves cross-domain generalization, 2020. URL https://arxiv.org/abs/2007.02454.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to generalize: Meta-learning for domain generalization, 2017. URL https://arxiv.org/abs/1710.03463.

Lei Qi, Hongpeng Yang, Yinghuan Shi, and Xin Geng. Normaug: Normalization-guided augmentation for domain generalization, 2024. URL https://arxiv.org/abs/2307.13492.

Ankur Singh and Senthilnath Jayavelu. Robust representation learning with self-distillation for domain generalization, 2024. URL https://arxiv.org/abs/2302.06874.

Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias, 2011.

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation, 2017.

Jiaolong Xu, Liang Xiao, and Antonio López. Self-supervised domain adaptation for computer vision tasks, 07 2019.

Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle, 2021. URL https://arxiv.org/abs/2104.02008.