Few-shot Adaptation to Distribution Shifts By Mixing Source and Target Embeddings

Yihao Xue [1] **Ali Payani** [2] **Yu Yang** [1] **Baharan Mirzasoleiman** [1]

Abstract

Pretrained machine learning models need to be adapted to distribution shifts when deployed in new target environments. When obtaining labeled data from the target distribution is expensive, few-shot adaptation with only a few examples from the target distribution becomes essential. In this work, we propose MixPro, a lightweight and highly data-efficient approach for few-shot adaptation. MixPro first generates a relatively large dataset by mixing (linearly combining) pre-trained embeddings of large source data with those of the few target examples. This process preserves important features of both source and target distributions, while mitigating the specific noise in the small target data. Then, it trains a linear classifier on the mixed embeddings to effectively adapts the model to the target distribution without overfitting the small target data. Theoretically, we demonstrate the advantages of MixPro over previous methods. Our experiments, conducted across various model architectures on 8 datasets featuring different types of distribution shifts, reveal that MixPro can outperform baselines by up to 7%, with only 2-4 target examples.

1. Introduction

Modern machine learning models often struggle to generalize well, when deployed in domains where the data distribution significantly differs from their source training data distribution (Quinonero-Candela et al., 2008). Thus, before deployment in a new domain, they need to be adapted to the target distribution. When abundant data from the target domain is available, one can simply fine-tune the model on the target data to improve its performance. Nevertheless,

in many real-world scenarios only a limited number of examples from the target domain is available. For example, data for training autonomous vehicles in severe weather conditions, are not only rare in certain geographical areas but also pose safety risks for data collection. In medical diagnosis, collecting data for rare diseases is often challenging. In such scenarios, fine-tuning on the small target data fails, by overfitting the few available examples instead of learning their features in a generalizable manner. *Few-shot adaptation* of a model to a new domain requires developing data-efficient methods that can effectively adapt the model with only *a few* examples from the target domain.

Recent studies revealed that neural networks learn versatile features from the training data in their penultimate layer (Kirichenko et al., 2022; Izmailov et al., 2022; Lee et al., 2022a; Mehta et al., 2022; Rosenfeld et al., 2022), and retraining only the last layer using the target data can effectively re-weight features to improve generalization on the target distribution. This approach is extremely lightweight and outperforms end-to-end fine-tuning of the whole model on the target data (Appendix C.1, Kirichenko et al. 2022), and therefore has recently become a popular technique to deal with different kinds of distribution shift (Kirichenko et al., 2022; Mehta et al., 2022; Rosenfeld et al., 2022; Izmailov et al., 2022; Chen et al., 2023; Qiu et al., 2023). Nevertheless, last-layer retraining is not the most data-efficient technique and can perform poorly for few-shot adaptation (Chen et al., 2023). To address this, Chen et al. (2023) proposed PRO^2 that finds a diverse set of features from the source data and re-weights them by training a linear model on the few available target data. But, as finding the set of compact features is done independent of the target data, this approach may miss capturing important relevant features for the target domain, and yields sub-optimal performance.

In this work, we develop a highly data-efficient method, MixPro, that takes advantage of the few available target examples in addition to the abundant source data, to re-weight the last layer of a pre-trained model, effectively adapting it to the target distribution. MixPro first constructs a large new dataset by *mixing* embedding of every source examples with that of a randomly chosen target example in the same class, via taking their weighted linear combination. Then, it trains

[1]Department of Computer Science, University of California, Los Angeles [2]Cisco Systems Inc.. Correspondence to: Yihao Xue <yihaoxue@g.ucla.edu>.

a linear model on the new dataset of mixed embeddings. In the new dataset, every example is a combination of a target example with a distinct source example. Therefore, the model trained on the mixed embeddings learns the target information, without overfitting (particular noise in) the few target examples. Besides, it can take advantage of the relevant information of the source data. In doing so, MixPro effectively adapts the model to the target domain.

We theoretically validate the effectiveness of our approach in two scenarios. Firstly, we demonstrate that when important features for the target distribution appear irrelevant in the source distribution, PRO^2 (Chen et al., 2023) fails to learn these features, resulting in poor generalization on the target. In contrast, MixPro effectively learns the important features in a data-efficient manner. Secondly, we adopt a model for subpopulation shift (Sagawa et al., 2020) and examine MixPro's performance in high-dimensional asymptotics. We show that MixPro enables effective learning from the target distribution while using the source data to prevent overfitting the small target. In doing so, MixPro outperforms last-layer training on only target (Kirichenko et al., 2022) or source data. Moreover, our analysis demonstrates how the severity of the shift, the noise level in the target data, and the target sample size influence the optimal mixing weight.

Empirically, we conduct extensive experiments on 8 datasets, including 3 subpopulation shift datasets – Waterbirds (Sagawa et al., 2019), UrbanCars (Li et al., 2023), bFFHQ (Kim et al., 2021) – and 5 domain generalization datasets – Camelyon17(Koh et al., 2021), PACS (Li et al., 2017), VLCS (Fang et al., 2013), Office-Home (Venkateswara et al., 2017) and Terra Incognita (Beery et al., 2018). We show that MixPro outperforms existing baselines, achieving superior performance even with very few (2 to 16 per class) target data, across various datasets and model architectures. It can achieve a maximum improvement of 7%, and on average, outperforms baselines by 4.3%/3.9% for 2-shot/4-shot adaptation across datasets. Finally, we show that MixPro remains the best method when hyperparameters are selected using cross validation with only a few target data.

2. Related Work

Distribution shift. Distribution shift, or domain shift, refers to a scenario where a model is trained on data from one distribution but is expected to generalize to test data from different distributions. Prior work primarily focuses on two settings: zero-shot generalization, which involves training a classifier on source data without seeing target data and expecting it to perform well on the target distribution (Tzeng et al., 2014; Ganin et al., 2016; Zhai et al., 2019; Yosinski et al., 2014; Sagawa et al., 2019; Arjovsky et al., 2019; Creager et al., 2021; Kornblith et al., 2019; Zhang & Ré, 2022; Wortsman et al., 2022; Sharif Razavian et al., 2014; Nam

et al., 2020; Oquab et al., 2014; Liu et al., 2021; Kumar et al., 2022; Pagliardini et al., 2022; Lee et al., 2022b); and test time adaptation, where the trained model is additionally updated upon seeing unlabeled test data from the target distribution (Sun et al., 2020; Varsavsky et al., 2020; Iwasawa & Matsuo, 2021; Wang et al., 2020; Zhang et al., 2021; Gandelsman et al., 2022). These differ from our setting since we consider a few-shot case where labeled target data are available but very few. Additionally, these works do not take advantage of the recent discovery that naively training a neural network can already learn generalizable features in the representation layer, thus do not address the newly identified bottleneck regarding last layer retraining, which we discuss in the next paragraph.

Last layer retraining for distribution shifts. A common intuition, shared in aforementioned works, is that the failure of deep networks to generalize to out-of-distribution data stems from their inability to learn generalizable features from their training data. However, this notion has been recently challenged by works including (Kirichenko et al., 2022; Izmailov et al., 2022; Lee et al., 2022a; Mehta et al., 2022; Rosenfeld et al., 2022). These studies demonstrate that the model trained on source data has already learned rich features, and that simply retraining the last linear layer (also called DFR (Kirichenko et al., 2022)) with target data can already achieve excellent performance. This suggests that the real bottleneck is not in feature learning, but rather in how to effectively re-weight the last layer features. However, in these works, the last layer is retrained with a large data from the target distribution, making them impractical in many real-world scenarios where target data are typically scarce. Our work specifically targets this direction, seeking more data-efficient solutions than standard linear probing.

Few-shot adaptation and sample efficiency. In scenarios where labeled target data are available but scarce, adapting the last layer to the target distribution in a data-efficient manner presents a significant challenge. The study most relevant to our paper is (Chen et al., 2023), which considers the same setting as ours, where the number of available target data for linear probing is very small (e.g., 2 to 32 per class). They propose PRO^2 that first learns a projection that maps the source data embeddings to orthogonal informative directions. Then, it passes the target data embeddings through the learned projection and perform linear probing using these projected embeddings. (Teney et al., 2022)'s method shares a similar intuition but does so with an additional loss term instead of explicitly enforcing orthogonality. However, we find that in both methods, the projection layer learned on the source data may miss directions in the embeddings that are important for the target distribution, as we will theoretically demonstrate. In contrast, our method does not suffer from this issue. We include these two methods as baselines in our experiments. (Zhu et al., 2022) and (Zhang et al., 2022)

also utilize both source and target data to achieve few-shot adaptation, albeit in different ways. However, their methods modifies the pretrained encoder, which could lead to overfitting on the limited dataset and degrades representation quality. We provide a comparison and further discussion of these methods in Appendix C.4.

3. Problem Formulation

We consider adapting a model to make accurate predictions on a shifted distribution in the few-shot setting, where we have access to a large source data but only a few examples from the shifted target distribution.

Formally, we have a source distribution $\mathscr{P}^s$ and a target distribution $\mathscr{P}^t$. The two distributions differ, for instance, due to data being collected at different times, from various regions, or in distinct environments. Similar to (Chen et al., 2023), we assume that the supports of the labels are the same in both distributions, while the supports of the inputs may or may not be the same. This encompasses both scenarios of subpopulation shift and domain generalization, as defined in (Koh et al., 2021). The source dataset $\mathcal{D}^s = \{\boldsymbol{x}_i^s, \boldsymbol{y}_i^s\}_{i=1}^n$, is composed of n examples from a set of classes $\mathcal{C}$ drawn from the source distribution $\mathscr{P}^s$, while the target dataset $\mathcal{D}^t = \{\boldsymbol{x}_i^t, \boldsymbol{y}_i^t\}_{i=1}^m$ consists of m examples from $\mathcal{C}$ drawn from the target distribution $\mathscr{P}^t$. In the few-shot setting we consider, $m \ll n$, where m is very small (e.g., in our experiments, we consider 2 to 32 examples per class). After adaptation, the model is evaluated on a held-out test set from the target distribution.

Note that our setting differs from prior works on zero-shot generalization (Sagawa et al., 2019; Kumar et al., 2022; Wortsman et al., 2022) under distribution shift, where the model is exclusively trained on the source data $\mathcal{D}^s$ and directly evaluated on the target distribution. A very small amount of target data that can be realistically obtained in many settings, is often essential to deal with arbitrary distribution shift, as we will confirm in our work. Closer to our setting are (Kirichenko et al., 2022; Izmailov et al., 2022; Lee et al., 2022a), which perform linear probing with target data on a pre-trained model that is fine-tuned on the source data. However, such methods are not very data-efficient, and may perform poorly when target data is very small. We specifically consider the adaptation with *a few* target examples, which is also recently considered in (Chen et al., 2023).

4. Mix & Probe (MixPro): Data-efficient Few-shot Adaptation to Target Domains

In this section, we first discuss the challenges and considerations of few-shot adaptation to distribution shift, and then introduce our method, MixPro, to overcome these challenges.

Challenges & considerations. Adapting a model to new domains with only a few examples from the target domain is very challenging for the following two reasons: First, while last-layer retraining on large target data outperforms end-to-end fine-tuning (Kirichenko et al., 2022; Izmailov et al., 2022; Rosenfeld et al., 2022; Yang et al., 2023), this approach is not very data-efficient and poses a risk of overfitting when only *a few* examples from the target distribution are available (*c.f.* (Chen et al., 2023), and our figures in Sec 6 and C). Second, adaptation to the target domain should take maximum advantage of the few available target examples to achieve optimal performance. Indeed, the prior work that first find a diverse set of features from the source and re-weight them using the target data (Teney et al., 2022; Chen et al., 2023) may yield sub-optimal performance. This is because some features that are important for the target domain, but seem unimportant in the source data, may be missed. We will confirm this theoretically in Sec. 5.

4.1. Linear probing on mixed source and target embeddings

We present our method, MixPro that effectively adapts a model with only a few data from the target distribution.

Key idea. The key idea of our method is to take advantage of the large source data and a few target examples to *construct* a new *large* dataset that contains generalizable information about the target domain. If this can be done, then last-layer retraining (linear probe) on the new constructed data achieves a superior performance as: (1) it does not overfit the (noise in the) few target examples, and (2) as the new constructed data contains information of both the source and target domains, the linear probe can take advantage of all the relevant information during training to achieve superior performance on the target distribution.

To construct a large dataset that contains information about the target data, our main idea is to leverage the large available source data and incorporate information about the target into it. To do so, we use a pre-trained backbone model $f : \mathcal{X} \to \mathbb{R}^d$ to map the source and target examples to a d-dimensional embedding space. Then, we create a new large embedding dataset by taking every source example and *mix* its embedding with the embedding of a randomly-chosen target example in the same class. For mixing the embeddings, we simply take linear combinations of the embeddings of the source and target examples. In the new dataset, every example is a combination of a target example with a distinct source example. Therefore, the model trained on the mixed embeddings learns the target information in a more generalizable manner, without overfitting (particular noise in) the few target examples. In addition, the new dataset still contains the information from the source data. Hence, the valuable information of the source data that are

relevant to the target domain but do not present in the few target examples can be easily leveraged by the linear probe to achieve superior performance.

Formal description. Formally, for source dataset $\mathcal{D}^s = \{\boldsymbol{x}_i^s, \boldsymbol{y}_i^s\}_{i=1}^n$ with n examples and a target dataset $\mathcal{D}^t = \{\boldsymbol{x}_i^t, \boldsymbol{y}_i^t\}_{i=1}^m$, with $m \ll n$ examples, MixPro has the following two steps:

(1) Mixing source & target. we construct a new dataset of embeddings with their labels, expressed as follows:

$$\mathcal{E}_{\text{mixed}} = \{(1-s)f(\boldsymbol{x}_i^s) + sf(\boldsymbol{x}_{j_i}^t), \ y_i^s\}_{i=1}^n, \quad (1)$$

where $j_i \overset{\text{unif.}}{\sim} \{j \mid y_j^t = y_i^s, j \in [m]\}$, meaning that each j_i is uniformly randomly sampled from the indices of the target data whose labels equal y_i^s. That is, for each source example, we randomly select one target example with the same label and take a weighted average of their embeddings to create a new embedding, where s is the weight for the target example and serves as a hyperparameter.

(2) Linear probe on mixed embeddings. Finally, we train a linear classifier g on the mixed embeddings by minimizing loss function $l(\cdot, \cdot)$:

$$\min_g \hat{\mathbb{E}}_{(\boldsymbol{z},y)\in\mathcal{E}_{\text{mixed}}} l(g(\boldsymbol{z}), y), \quad (2)$$

where $\hat{\mathbb{E}}$ denotes the empirical expectation.

MixPro vs (manifold) Mixup. We highlight the difference between MixPro and (Manifold) Mixup (Zhang et al., 2017; Verma et al., 2019). Mixup improves the in-distribution generalization by training the model on mixed *inputs and labels* of pairs of examples, using *randomly sampled weights*. In contrast, our method is specifically designed for data-efficient adaptation and involves mixing embeddings of source and target data with a *fixed* weight, *without* mixing labels of examples in difference classes. We will show MixPro outperforms Mixup in our experiments in Section 6.

Using a fixed s. We note that randomly sampling s, e.g. from a Beta distribution as is done in Mixup, is not effective, as s determines the proportions of source and target data in the mixture. Hence, it should be set appropriately for every dataset. When s is randomly selected from a Beta distribution, its expected value is 0.5, implying equal weights for source and target data, which can lead to suboptimal performance (*c.f.* Figure 3).

The selection of s. Prior work (Kirichenko et al., 2022; Chen et al., 2023; Teney et al., 2022), involve hyperparameter tuning based on a hypothetical large validation set from the target distribution. Indeed, addressing hyperparameter selection in the few-shot scenario has not been explored before. In Sec. 6.2, we demonstrate that 2-fold cross-validation with the few available target examples, can lead to reasonable hyperparameter selection.

Our method outperforms prior work, whether using this strategy or a hypothetical large validation set.

5. Theoretical Analysis

In this section, we provide theoretical analysis in two scenarios that comprehensively showcase the advantages of Mix-Pro. Firstly, we compare MixPro with PRO^2 (Chen et al., 2023), which learns a projection of pre-trained embeddings of source data before performing linear probing on target data. We demonstrate that PRO^2 can overlook important features for the target distribution, whereas our method, which directly performs linear probing on a mixture of source and target data, does not have this issue. Secondly, we consider a subpopulation shift model and demonstrate the trade-off in selecting the mixing weight, as well as the advantage of our method over using solely source or solely target data.

5.1. A case study of domain generalization: the advantage of MixPro over prior work

Recent works (Teney et al., 2022; Chen et al., 2023) suggested leveraging the large source data to learn a projection from the original embedding space to a lower-dimensional space that preserves a small and diverse (orthogonal) set of features from the source data. Linear probing on the target domain using these orthogonal directions has been found to be more data-efficient. But, the drawback of this approach is that the projection, being learned solely from source data, may disregard directions that are less significant in the source but critical for the target distribution. Next, we show that such methods may result in sub-optimal performance.

In our analysis, we make assumptions about the distribution of embeddings on which different methods are applied. In the source distribution, each label y is uniformly drawn from $\{-1, 1\}$ and the corresponding embedding $\boldsymbol{z}$ is given by $\boldsymbol{z} = y\boldsymbol{v} + \boldsymbol{\xi}$, where $\boldsymbol{v} \overset{\text{uni}}{\sim} \{\boldsymbol{v}_1, \boldsymbol{v}_2\}$, $\boldsymbol{\xi} = \xi\boldsymbol{v}_3$ with $\xi \sim \mathcal{N}(0, \sigma)$. $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ are orthogonal unit vectors. Simply put, in the embedding space, the label information is carried in two directions, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, with each embedding encoding label information in one of these two directions, while having some noise in the third direction, $\boldsymbol{v}_3$.

To model the shift in distribution, we consider the following target data distribution where each embedding is given by $\boldsymbol{z} = y\boldsymbol{v} + \boldsymbol{\xi}$ where $\boldsymbol{v} \overset{\text{unif.}}{\sim} \{\boldsymbol{v}_2, \boldsymbol{v}_3\}$, $\boldsymbol{\xi} = \xi\boldsymbol{v}_1$ with $\xi \sim \mathcal{N}(0, \sigma)$. The shift is such that, in the target distribution, the label information is carried in $\boldsymbol{v}_2$, which is the shared part with the source distribution, but it differs in that it can also be carried in $\boldsymbol{v}_3$ and never in $\boldsymbol{v}_1$. Instead, $\boldsymbol{v}_1$ consists only of noise in the target distribution.

Assuming we have access to n source data embeddings in dataset $\mathcal{E}^s$ and m target data points in dataset $\mathcal{E}^t$. To model

the case where the source data is very large, we let $n \to \infty$.

This simple model allows seeing the failure mode of previous methods, including (Chen et al., 2023; Teney et al., 2022). Specifically, we consider PRO^2 (Chen et al., 2023), while noting that (Teney et al., 2022) shares a similar underlying mechanism, and is nearly equivalent to PRO^2 for linear models. We defer detailed discussion to Appendix A.1.

PRO^2 first learns a projection layer $\mathbf{\Pi}^* = [\mathbf{\Pi}_1^*, \dots, \mathbf{\Pi}_p^*] \in \mathbb{R}^{d \times p}$ using the source data. It learns the columns in $\mathbf{\Pi}^*$ such that they are orthogonal to each other. The formalization in (Chen et al., 2023) is as follows:

$$\mathbf{\Pi}_i^* = \arg\min_{\mathbf{\Pi}_i} \hat{\mathbb{E}}_{(\mathbf{z},y) \in \mathcal{E}^s} l(\mathbf{\Pi}_i^\top \mathbf{z}, y) \qquad (3)$$

$$s.t. \ \mathbf{\Pi}_i^* \perp \mathbf{\Pi}_j^* \text{ for all } j < i,$$

where l is the loss function. Then it performs linear probing on the projected embeddings of the target data

$$\mathbf{w}_{\text{PRO}^2}^* = \arg\min_{\mathbf{w}} \hat{\mathbb{E}}_{(\mathbf{z},y) \in \mathcal{E}^t} l(\mathbf{z}^\top \mathbf{\Pi}^* \mathbf{w}, y). \qquad (4)$$

Intuitively, in the source distribution, $\mathbf{v}_3$ does not carry any label information, thus it is not learned by $\mathbf{\Pi}^*$ in Eq. (3). Thus, no information carried along $\mathbf{v}_3$ would exist after passing through $\mathbf{\Pi}$. Therefore, any linear model on top of the projected embeddings would fail on any test example $\mathbf{z}$ that contains $y\mathbf{v}_3$, which accounts for half of the target data.

In contrast, MixPro can succeed because we directly train a linear model on the mixture of source and target embeddings. The linear model, having been exposed to $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, will effectively learn all these features.

The following theorem summarizes the above discussion:

Theorem 5.1. *Assuming that the noise is sufficiently small, $\sigma = o(1)$, and $l(.,.)$ is the MSE loss, then w.h.p.:*
(1) The test loss on target data achieved by PRO^2 can always be lower bounded by a constant order:

$$\mathbb{E}_{(\mathbf{z},\mathbf{y}) \in \text{target dist.}} l(\mathbf{z}^\top \mathbf{\Pi}^* \mathbf{w}_{\text{PRO}^2}^*, y) \geq 0.5 - o(1).$$

(2) MixPro learns $\mathbf{w}_{MixPro}^* = \arg\min_{\mathbf{w}} \hat{\mathbb{E}}_{(\mathbf{z},y) \in \mathcal{E}_{mixed}} l(\mathbf{z}^\top \mathbf{w}, y)$, as described in Section 4. When $\exists \epsilon = \Theta(1)$ s.t. $\epsilon < s < 1 - \epsilon$, it achieves a test loss on target data that can be upper bounded by:

$$\mathbb{E}_{(\mathbf{z},\mathbf{y}) \in \text{target dist.}} l(\mathbf{z}^\top \mathbf{w}_{MixPro}^*, y)$$
$$\leq \left(\sigma + O(\sqrt{\frac{\log m}{m}}) \right)^2 = o(1).$$

The necessity of mixing with source data (advantage over DFR). Here, we further discuss why mixing target data with source data is important, compared to DFR, which solely trains on the target data. We provide a rough intuition by considering the extreme case where the noise is
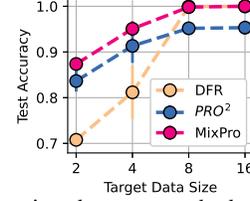

Figure 1. Comparison between methods on synthetic data.

very large, $\sigma = \omega(1)$. In this scenario, with probability of at least $1 - \delta$, the test loss on the target distribution $\mathbb{E}_{(\mathbf{z},\mathbf{y}) \in \text{target dist.}} l(\mathbf{z}^\top \mathbf{w}_{MixPro}^*, y)$, takes the form:

$$(\psi(s, \sigma) + O(s^2 \sigma^2 \sqrt{\frac{\log(1/\delta)}{m}}))^2,$$

where $\psi(s, \sigma)$ is a function of s and σ, independent of m and δ. See explanation in Appendix A.3. Note that using $s = 1$ in our method corresponds to DFR, i.e., using only the target data. We observe that reducing s, or placing more weight on the source data, diminishes the second term, which reflects the error due to the interaction of a small sample size m and noise σ. Next, we will consider another example that allows us to provide a more fine-grained analysis, illustrating the trade-off involved in selecting the value of s. In Fig. 1, we compare MixPro with DFR and PRO^2 using synthetic data similar to that in our analysis (see details in Appendix B.5).

5.2. A case study of subpopulation shift: trade-off between learning target and overfitting noise

Here, we adopt a variation of the subpopulation shift model used in prior work (Sagawa et al., 2020; Wald et al., 2021; Aubin et al., 2021; Yao et al., 2022; Xue et al., 2023) to provide a fine-grained analysis demonstrating the benefits of MixPro over DFR, which only uses target data. We will show that MixPro makes a trade-off between learning from the target data and overfitting noise, outperforming both DFR and only training on the source data.

We first define a family of data distributions $\mathscr{E}(\mu)$ parameterized by μ, where each embedding-label pair $(\mathbf{z}, y)$ is generated as follows. First, sample the label y uniformly from $\{1, -1\}$, then sample a from $\{1, -1\}$ with probabilities $\Pr(a = y) = \mu$ and $\Pr(a = -y) = 1 - \mu$. Finally, generate the embedding as $\mathbf{z} = [z_1, z_2, \boldsymbol{\xi}^\top]^T$ in $\mathbb{R}^d$, where $z_1 \sim \mathcal{N}(y, \sigma_1^2)$, $z_2 \sim \mathcal{N}(a, \sigma_2^2)$, and $\boldsymbol{\xi} \sim \mathcal{N}(0, \frac{\sigma_\xi^2}{d-2} \mathbf{I}_{d-2})$. This model and its variations, widely utilized in many prior works (Sagawa et al., 2020; Wald et al., 2021; Aubin et al., 2021; Yao et al., 2022; Xue et al., 2023), are designed to simulate a specific type of distribution shift known as subpopulation shift or spurious correlation. The first coordinate z_1, the *core* feature, carries the label information, while the second coordinate z_2, the *spurious* feature, carries information about an attribute a, with its correlation to the label y is dictated by μ. When $\mu > 1/2$, a and y are correlated. The remaining coordinates represent random noise.

(a) s=0(only source)   (b) $s = 0.4$ (c) $s = 1$ (DFR)
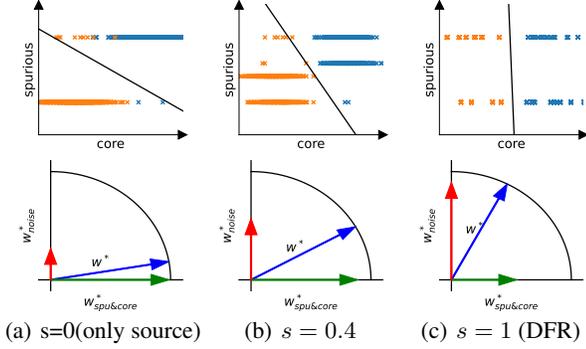
Figure 2. **A trade-off between learning target information and preventing overfitting noise. Top: incorporating more target information with increasing s.** The orange and blue dots represent examples from two classes in the spurious and core coordinates. The black line represents the decision boundary of the learned model along these first two coordinates. **Bottom: With increasing s, the model has a larger component in the coordinates for random noise.** The normalized weights vector is displayed in blue. Its component in the first two coordinates is shown in green, and the component in the directions of noise is shown in red.

Specifically, we consider the case where the source distribution is $\mathscr{E}(p_{\text{spu}})$ with $p_{\text{spu}} > 1/2$, and the target distribution is $\mathscr{E}(1/2)$ where a is completely independent of y.

The large source data is effectively less noisy than the small target data. Consider the very small sample size of the small target data. When the dimension is significantly larger than the number of examples, there is a high probability that the noise in each target example is asymptotically orthogonal to that in every other example, according to the concentration behavior of random Gaussian vectors. Due to this orthogonality, the noise appear unique to each example and thus the model can learn the noise to predic labels, especially when the scale of the noise is significantly large. Similarly, in real world data, noises are spread out across the entire embeddings with low correlation among themselves, as opposed to the useful features, which are usually concentrated in a lower-dimensional subspace (Papyan et al., 2017; Morcos et al., 2018; Oymak et al., 2019; Huh et al., 2021). In contrast, in the source data, which comprises many more examples, the noise cannot be used for prediction of the label. Therefore, the small target data is more affected by noise, while the source data is less so.

Visualizing the trade-off between learning target information and preventing overfitting noise. First, we visualize the decomposition of the weights $\boldsymbol{w}^*$ of a linear model trained using MixPro in Figure 2. Specifically, we break down the weights into two components: $\boldsymbol{w}^*_{s,c}$ and $\boldsymbol{w}^*_{noise}$. The former is the component of $\boldsymbol{w}^*$ in the span of the spurious and core features, while the latter is the component in the orthogonal complement, corresponding to noise. From Figure 2, we observe that as s increases, the data distribution

(a) vary severity of shift   (b) vary $\frac{\text{noise}}{\text{target data size}}$
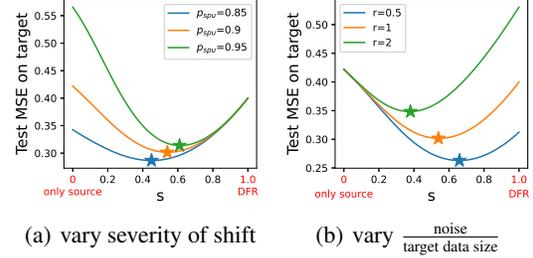
Figure 3. Left: A larger p_{spu}, indicating a more severe shift, necessitates a larger s; Right: A larger $r = \frac{\sigma_\xi^2}{m}$, signifying greater noise or a smaller target data size, necessitates a smaller s.

leans more towards the target distribution, leading the model to align more with the core coordinate (top row). However, simultaneously, the model also learns more noise (bottom row). Fig. 2(c) shows that at $s = 1$ (using only target data, i.e., DFR (Kirichenko et al., 2022)), the model learns excessive noise. In contrast, Fig. 2(a) shows that at $s = 0$ (using only source data), the model learns predominantly from the spurious feature. As shown in Fig. 2(b), by setting an intermediate s, we can strike a balance between learning target information and avoiding overfitting noise.

Formal theoretical results. Next, we present our formal analysis. We make the following assumptions:

Assumption 5.2. (1) Similar to (Sagawa et al., 2020), we assume that the second coordinate, carrying the spurious attribute a, has a smaller variance than the first coordinate, which carries the true label information. This makes the spurious attribute easier to learn. Specifically, we set their corresponding variances to be $\sigma_1 > 0$ and $\sigma_2 = 0$. (2) To obtain closed-form results, we examine the high-dimensional asymptotic limit where n, d, and m tend to infinity. To account for the small size of the target data, we assume that $\frac{n}{d} \to \infty$, while $\frac{m}{d} \to 0$, ensuring that m is much smaller than n. (3) Additionally, we assume that $\sigma_\xi \to \infty$, while ratio between the noise and the sample size of the target data, i.e., $\frac{\sigma_\xi^2}{m}$, remains constant at r. This allows us to observe the effects of the target dataset's size and noise level.

We perform an asymptotic analysis to derive the exact expression of the test performance on the target data, and plot it to observe the benefit of mixing source and target data, as well as how it interacts with the severity of distribution shift, noise level, and target dataset size.

Theorem 5.3. *Consider the linear model $\boldsymbol{w}^*_{MixPro}$ learned by MixPro with $l(.,.)$ being MSE loss with ℓ_2 regularization. With assumption 5.2, the test loss on the target distribution achieved by $\boldsymbol{w}^*_{MixPro}$ can be expressed in closed form in terms of p_{spu}, σ_1, r and s, as shown in Appendix A.4.*

We plot the expression in Figure 3. Note that $s = 0$ corresponds to using only source data and $s = 1$ to using only target data (DFR). Across all plots, we observe that the test loss on the target distribution first decreases and then in-

creases as s increases. An intermediate value of s yields the best performance, confirming that a mix between the two performs better than using either only source or only target (DFR). In Fig 3(a), we plot the curves under different p_{spu} values, which indicate the severity of the distribution shift. We see that with a larger p_{spu}, implying a greater shift, the optimal s becomes larger, suggesting more emphasis on the target data. In Fig 3(b), we plot the curves under different r values, the ratio of noise to target sample size. We observe that a larger r, indicating either greater noise or a smaller number of target examples, results in the optimal s being smaller, aligning with the intuition that in such cases, we should rely more on the source data to counteract noise. This is also empirically confirmed on real datasets in Figure 9 in Appendix C.3.

6. Experiments

In this section, we will demonstrate the superior performance of MixPro over existing baselines across various datasets and using different backbone models. Additionally, we will explore realistic hyperparameter tuning in the few-shot scenario, which is not addressed in prior work.

Datasets. We consider 8 datasets with different types of distribution shifts. These include 3 subpopulation shift benchmarks: WaterBirds (Sagawa et al., 2019), bFFHQ (Kim et al., 2021), and UrbanCars (Li et al., 2023); along with 5 domain generalization benchmarks: PACS (Li et al., 2017), VLCS (Fang et al., 2013), Office-Home (Venkateswara et al., 2017), Terra Incognita (Beery et al., 2018), and Camelyon17 (Koh et al., 2021). Due to space limitations, we defer the details to Appendices B.1 and B.2. For the target dataset size, we consider the few-shot scenario where $\{2, 4, 8, 16\}$ data points per class are sampled from the target distribution (except for Terra Incognita, for which we use only $\{2, 4, 8\}$ samples, because the smallest class, i.e., class index 9, has only 12 examples). The test set is constructed using the remaining data in the target distribution.

Baselines. We consider the following four baselines: (1) DFR (Deep Feature Reweighting) (Kirichenko et al., 2022), which performs standard linear probing using the target data. (2) (Manifold) Mixup (Zhang et al., 2017; Verma et al., 2019) where we additionally apply Mixup to the embeddings while performing DFR. (3) Teney et al. (2022), which trains multiple models on the source data while minimizing the similarity between these models' gradients. We note that their method does not specify how to apply these models to target data. To maximize effectiveness, we train a linear model on the concatenation of the outputs of these models using the target data. The final prediction is thus a weighted combination of these models, with weights determined using the given target data. (4) PRO^2 (Chen et al., 2023), which first learns a linear projection to map the embeddings onto orthogonal directions using source data, and then performs linear probing using target data on the projected embeddings.

Backbone models. For the backbone model that produces embeddings, we consider two models: (1) The standard ImageNet-pretrained ResNet 50 and (2) the ViT-L/16 model pretrained with SWAG (Singh et al., 2022). These models are publicly available in TorchVision. For model (1), we additionally fine-tune the backbone model on the source data following (Kirichenko et al., 2022; Izmailov et al., 2022), as (Rosenfeld et al., 2022) suggests that this significantly improves the final performance. For model (2), we use the original weights directly, as (Mehta et al., 2022) has demonstrated the superior robustness provided by the original SWAG weights.

Hyperparameter tuning. In Table 1, we present the range of hyperparameters considered for each method, and the ranges are sourced from the corresponding original papers. We test our method and existing baselines using two different approaches to tune hyperparameters, for a comprehensive evaluation. The first setting (Section 6.1) follows the practice of prior works (Teney et al., 2022; Kirichenko et al., 2022; Izmailov et al., 2022; Chen et al., 2023), where hyperparameters are chosen based on performance on a held-out large validation set from the target distribution. We note that such a large validation set is essentially *hypothetical* and is not typically available in real-world scenarios, particularly in a few-shot setting where only a limited number of target data points are accessible. Prior work that uses this approach does admit that selecting hyperparameters in a realistic manner remains an open problem that they have not addressed yet. Therefore, we also test a second, more realistic setting for hyperparameter selection in few-shot learning, where we tune using cross-validation with only the few available target data. The details are provided in Section 6.2.

6.1. Results with hypothetical large validation data

The results for SWAG-pretrained ViT-L/16 are displayed in Figures 4, and results for ResNet50 are deferred to Fig 7 in Appendix C.2. We report the results over 5 runs. Overall, while some methods may perform comparably to MixPro on certain datasets, they falter on others. In contrast, MixPro consistently achieves the best performance across various datasets, data sizes, and backbone models, as shown from the *average* plot in Figures 4 and 7. The superiority of MixPro over DFR and Mixup is particularly noticeable in scenarios with small target dataset sizes (e.g., 2 per class). This is because DFR relies solely on the limited target data, leading to an insufficient sample size and consequently a higher risk of overfitting to noise, as elaborated in Section 5. The performance of PRO^2 largely depends on the severity of the distribution shift. Specifically, based on our fine-
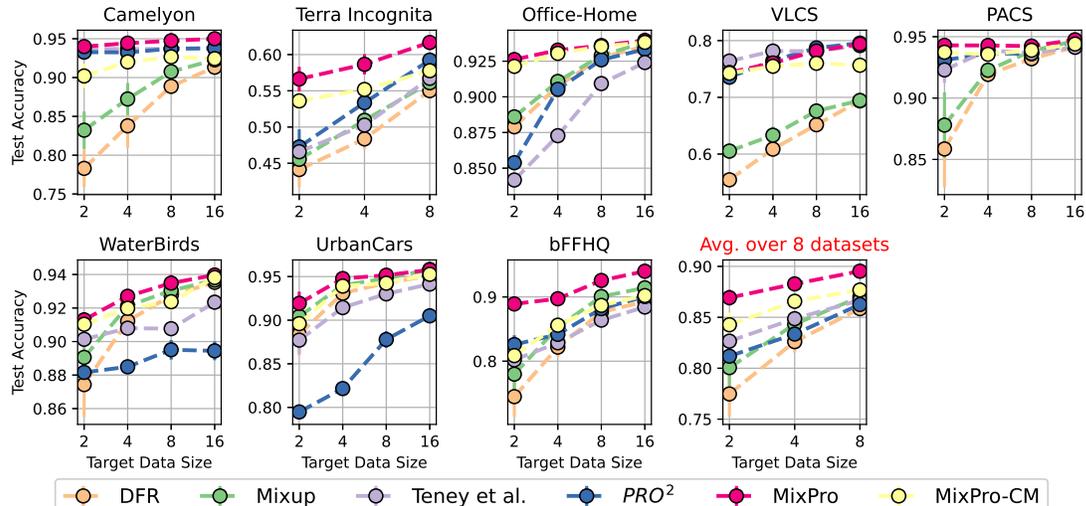
Figure 4. Test accuracy on the target distribution versus target data size for all baselines across the 8 datasets we consider. Here, we use the SWAG-pretrained ViT-L/16 model as the backbone . Overall, while some methods may perform comparably to MixPro on certain datasets, they falter on others. In contrast, MixPro consistently achieves the best performance across datasets and data sizes.
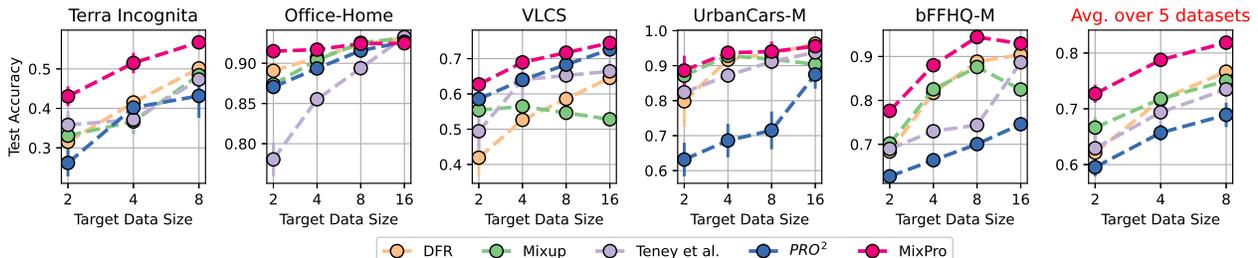

Figure 5. Results with hyperparameters tuned using cross-validation with only the few given target data. We use the SWAG-pretrained ViT-L/16 model as the backbone. In these scenarios, MixPro continues to outperform the others as the best method.

grained results in Appendix C.1 for the subpopulation shift datasets (WaterBirds, UrbanCars, bFFHQ), PRO^2 works better when the target distribution is *balanced* (Fig 6 top), i.e., all subpopulations are equally represented in the target distribution, despite still being outperformed by MixPro. However, in scenarios where the target distribution only consists of the *minority* subpopulations, indicating a more severe shift, PRO^2 achieves worse performance compared to others (Fig 6 middle). The corresponding three plots in Fig 4 present the average results over the two types of shifts. Furthermore, on domain generalization datasets (Fig 4, top row), which present a severe shift as the target and source come from completely disjoint domains, PRO^2 also achieves less satisfactory performance overall. In contrast, MixPro consistently performs well across datasets. This is supported by our theoretical analysis in Section 5.1, which explains that when the source and target differ significantly, such as when the target has examples not present in the source's support, PRO^2 may filter out crucial features of the embeddings during projection, while MixPro learns all vital aspects, thereby confirming MixPro's advantage over PRO^2. Besides, Teney et al. (2022) is also surpassed by MixPro, for a similar reason; they initially train multiple

models exclusively on source data before applying them to the target data, potentially missing relevant features.

6.2. Results with cross validation using few-shot data

Prior works, such as (Kirichenko et al., 2022; Izmailov et al., 2022; Chen et al., 2023), tuning hyperparameter—including learning rate, weight decay, and method-specific hyperparameters—by evaluating accuracy on a large held-out validation set. However, this may not be realistic for few-shot adaptation, as it contradicts the assumption of having only a few target data points available. Therefore, to evaluate if the methods can operate effectively in a true few-shot scenario without additional data, we employ standard k-fold cross-validation using the limited target data available for hyperparameter selection. Considering the smallest case in our experiments, where the target data size is only 4 (2 per class and 2 classes), we set $k = 2$ to ensure that each fold has at least one data point per class.

We conduct experiments on Terra Incognita, VLCS, Urban-Cars, bFFHQ and Offie-Home. Figs 5 and 8 show the results for different model architectures. We observe that although the overall performance slightly drops, it remains reason-

able, as the decrease is mostly within 10% compared to the case where a large validation set is used, considering that no extra information beyond the given few examples is used. This confirms that cross-validation strategy is promising and is a potential solution to the hyperparameter selection. Importantly, MixPro still stands out with the highest performance, showing that its advantage is retained even when hyperparameters are selected using very limited information.

6.3. Relaxing the requirement of source data availability

In some scenarios, the source and target data may not be available simultaneously. In other words, by the time we encounter the target data and wish to adapt to it, the source data may no longer be accessible. However, MixPro requires both data to be available at the same time to train a linear probe on the mixed source and target embeddings. This implies that one may always need to store the source embeddings to use them when a new target data is encountered. To alleviate this requirement, we propose a simple approach of storing only minimal information about the source data, thereby making the method more flexible.

Specifically, we store only the average embeddings (mean) of each class, resulting in only #classes embeddings. When we create the mixed embeddings, we combine each target data point with the corresponding class mean from the source. We call this modified method MixPro-CM and test it on all the datasets under the same settings as shown in Figure 4. From the figure, we observe that MixPro-CM performs slightly worse than the original MixPro, but still outperforms all other methods. We believe that further exploration into how to improve it, such as retaining more fine-grained information than just class means, would be a valuable direction for future research.

7. Conclusion

In this work, we proposed MixPro for few-shot adaptation to distribution shifts. MixPro performs linear probing on mixed embeddings of source and target data, avoiding overfitting to the small target data while effectively learning target information. Moreover, we proposed a variation of MixPro which requires minimal information about the source data, i.e., the class means. We provided theoretical analysis showing its advantages over previous methods and conduct extensive experiments demonstrating its superior performance across various scenarios for few-shot adaptation under distribution shift.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Aubin, B., Słowik, A., Arjovsky, M., Bottou, L., and Lopez-Paz, D. Linear unit-tests for invariance discovery. *arXiv preprint arXiv:2102.10867*, 2021.

Beery, S., Van Horn, G., and Perona, P. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 456–473, 2018.

Chen, A. S., Lee, Y., Setlur, A., Levine, S., and Finn, C. Project and probe: Sample-efficient domain adaptation by interpolating orthogonal features. *arXiv preprint arXiv:2302.05441*, 2023.

Creager, E., Jacobsen, J.-H., and Zemel, R. Environment inference for invariant learning. In *International Conference on Machine Learning*, pp. 2189–2200. PMLR, 2021.

Demmel, J. The componentwise distance to the nearest singular matrix. *SIAM Journal on Matrix Analysis and Applications*, 13(1):10–19, 1992.

Fang, C., Xu, Y., and Rockmore, D. N. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1657–1664, 2013.

Gandelsman, Y., Sun, Y., Chen, X., and Efros, A. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., and Lempitsky, V. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

Huh, M., Mobahi, H., Zhang, R., Cheung, B., Agrawal, P., and Isola, P. The low-rank simplicity bias in deep networks. *arXiv preprint arXiv:2103.10427*, 2021.

Iwasawa, Y. and Matsuo, Y. Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, 34: 2427–2440, 2021.

Izmailov, P., Kirichenko, P., Gruver, N., and Wilson, A. G. On feature learning in the presence of spurious correlations. *Advances in Neural Information Processing Systems*, 35:38516–38532, 2022.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Kim, E., Lee, J., and Choo, J. Biaswap: Removing dataset bias with bias-tailored swapping augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14992–15001, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kirichenko, P., Izmailov, P., and Wilson, A. G. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv preprint arXiv:2204.02937*, 2022.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pp. 5637–5664. PMLR, 2021.

Kornblith, S., Shlens, J., and Le, Q. V. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2661–2671, 2019.

Kumar, A., Raghunathan, A., Jones, R., Ma, T., and Liang, P. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

Lee, Y., Chen, A. S., Tajwar, F., Kumar, A., Yao, H., Liang, P., and Finn, C. Surgical fine-tuning improves adaptation to distribution shifts. *arXiv preprint arXiv:2210.11466*, 2022a.

Lee, Y., Yao, H., and Finn, C. Diversify and disambiguate: Out-of-distribution robustness via disagreement. In *The Eleventh International Conference on Learning Representations*, 2022b.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550, 2017.

Li, Z., Evtimov, I., Gordo, A., Hazirbas, C., Hassner, T., Ferrer, C. C., Xu, C., and Ibrahim, M. A whac-a-mole dilemma: Shortcuts come in multiples where mitigating one amplifies others. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20071–20082, 2023.

Liu, E. Z., Haghgoo, B., Chen, A. S., Raghunathan, A., Koh, P. W., Sagawa, S., Liang, P., and Finn, C. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*, pp. 6781–6792. PMLR, 2021.

Mehta, R., Albiero, V., Chen, L., Evtimov, I., Glaser, T., Li, Z., and Hassner, T. You only need a good embeddings extractor to fix spurious correlations. *arXiv preprint arXiv:2212.06254*, 2022.

Morcos, A. S., Barrett, D. G., Rabinowitz, N. C., and Botvinick, M. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

Nam, J., Cha, H., Ahn, S., Lee, J., and Shin, J. Learning from failure: De-biasing classifier from biased classifier. *Advances in Neural Information Processing Systems*, 33: 20673–20684, 2020.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724, 2014.

Oymak, S., Fabian, Z., Li, M., and Soltanolkotabi, M. Generalization guarantees for neural networks via harnessing the low-rank structure of the jacobian. *arXiv preprint arXiv:1906.05392*, 2019.

Pagliardini, M., Jaggi, M., Fleuret, F., and Karimireddy, S. P. Agree to disagree: Diversity through disagreement for better transferability. *arXiv preprint arXiv:2202.04414*, 2022.

Papyan, V., Romano, Y., and Elad, M. Convolutional neural networks analyzed via convolutional sparse coding. *The Journal of Machine Learning Research*, 18(1):2887–2938, 2017.

Qiu, S., Potapczynski, A., Izmailov, P., and Wilson, A. G. Simple and fast group robustness by automatic feature reweighting. *arXiv preprint arXiv:2306.11074*, 2023.

Quinonero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. *Dataset shift in machine learning*. Mit Press, 2008.

Rosenfeld, E., Ravikumar, P., and Risteski, A. Domain-adjusted regression or: Erm may already learn features

sufficient for out-of-distribution generalization. *arXiv preprint arXiv:2202.06856*, 2022.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

Sagawa, S., Raghunathan, A., Koh, P. W., and Liang, P. An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning*, pp. 8346–8356. PMLR, 2020.

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.

Singh, M., Gustafson, L., Adcock, A., de Freitas Reis, V., Gedik, B., Kosaraju, R. P., Mahajan, D., Girshick, R., Dollár, P., and Van Der Maaten, L. Revisiting weakly supervised pre-training of visual perception models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 804–814, 2022.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Teney, D., Abbasnejad, E., Lucey, S., and Van den Hengel, A. Evading the simplicity bias: Training a diverse set of models discovers solutions with superior ood generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16761–16772, 2022.

Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., and Darrell, T. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

Varsavsky, T., Orbes-Arteaga, M., Sudre, C. H., Graham, M. S., Nachev, P., and Cardoso, M. J. Test-time unsupervised domain adaptation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part I 23*, pp. 428–436. Springer, 2020.

Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5018–5027, 2017.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. Manifold mixup: Better

representations by interpolating hidden states. In *International conference on machine learning*, pp. 6438–6447. PMLR, 2019.

Wainwright, M. J. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge university press, 2019.

Wald, Y., Feder, A., Greenfeld, D., and Shalit, U. On calibration and out-of-domain generalization. *Advances in neural information processing systems*, 34:2215–2227, 2021.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Wortsman, M., Ilharco, G., Kim, J. W., Li, M., Kornblith, S., Roelofs, R., Lopes, R. G., Hajishirzi, H., Farhadi, A., Namkoong, H., et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7959–7971, 2022.

Xue, Y., Joshi, S., Nguyen, D., and Mirzasoleiman, B. Understanding the robustness of multi-modal contrastive learning to distribution shift. *arXiv preprint arXiv:2310.04971*, 2023.

Yang, Y., Zhang, H., Katabi, D., and Ghassemi, M. Change is hard: A closer look at subpopulation shift. *arXiv preprint arXiv:2302.12254*, 2023.

Yao, H., Wang, Y., Li, S., Zhang, L., Liang, W., Zou, J., and Finn, C. Improving out-of-distribution robustness via selective augmentation. In *International Conference on Machine Learning*, pp. 25407–25437. PMLR, 2022.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A. S., Neumann, M., Dosovitskiy, A., et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Zhang, M. and Ré, C. Contrastive adapters for foundation model group robustness. *Advances in Neural Information Processing Systems*, 35:21682–21697, 2022.

Zhang, M., Marklund, H., Dhawan, N., Gupta, A., Levine, S., and Finn, C. Adaptive risk minimization: Learning to

adapt to domain shift. *Advances in Neural Information Processing Systems*, 34:23664–23678, 2021.

Zhang, W., Shen, L., Zhang, W., and Foo, C.-S. Few-shot adaptation of pre-trained networks for domain shift. *arXiv preprint arXiv:2205.15234*, 2022.

Zhu, R., Yu, X., and Li, S. Progressive mix-up for few-shot supervised multi-source domain transfer. In *The eleventh international conference on learning representations*, 2022.

A. Theoretical Analysis

A.1. Discussion on Teney et al. (2022)

(Teney et al., 2022) proposes training multiple models $(h_1, \ldots, h_{n_{\text{models}}})$ on the embeddings of the source data while minimizing the similarity of these models, before adapting to the target data. We will see that this process closely resembles the projection step of PRO². The objective of (Teney et al., 2022) is defined as:

$$\min \left(\sum_{i=1}^{n_{\text{models}}} \hat{\mathbb{E}}_{(\boldsymbol{z},y) \in \mathcal{E}^s} l(h_i(\boldsymbol{z}), y) + \lambda \sum_{i \neq j} \sum_{(\boldsymbol{z},y) \in \mathcal{E}^s} \text{sim}_{\boldsymbol{z}}(h_i, h_j) \right),$$

where $\text{sim}_{\boldsymbol{z}}(h_i, h_j)$ is the inner product between the gradients of two models with respect to the input $\boldsymbol{z}$.

Considering that h's are linear models defined by $h_i(\boldsymbol{z}) = \boldsymbol{\theta}_i^\top \boldsymbol{z}$, the gradient with respect to any input $\boldsymbol{z}$ is just the model weights, $\nabla_{\boldsymbol{z}} h_i(\boldsymbol{z}) = \boldsymbol{\theta}_i$. Consequently, the objective becomes:

$$\min_{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{n_{\text{models}}}} \left(\sum_{i=1}^{n_{\text{models}}} \hat{\mathbb{E}}_{(\boldsymbol{z},y) \in \mathcal{E}^s} l(\boldsymbol{\theta}_i^\top \boldsymbol{z}, y) + \lambda' \sum_{i \neq j} \boldsymbol{\theta}_i^\top \boldsymbol{\theta}_j \right).$$

This goal bears a strong resemblance to that of PRO² where $\boldsymbol{\theta}_i$'s can be considered analogous to the projection vectors $\boldsymbol{\Pi}_i$'s in PRO² The objective above is to align each projection's output with the label y, while promoting orthogonality among the projections. The only difference is that (Teney et al., 2022) achieves this through a regularization term, whereas PRO² enforces exact orthogonality. We hypothesize that this additional flexibility is likely why (Teney et al., 2022) generally outperforms PRO² with hyperparameter tuning in our experiments in Section 6.

A.2. Proof of Theorem 5.1

A.2.1. ANALYSIS OF MIXPRO

Notations. Let $\{\tilde{z}_i\}_{i=1}^n$ denote the mixed embeddings where $\tilde{z}_i = (1-s)z_i^s + s z_{j_i}^t$.

Assumptions. As mentioned in Section 5.1, we assume $n \to \infty$ and $\sigma = o(1)$. We also assume that $\exists \epsilon = \Theta(1)$ s.t. $\epsilon < s < 1 - \epsilon$ as mentioned in the statement of Theorem 5.1.

Without loss of generality, we consider $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ to be $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$, respectively, i.e., the three standard basis vectors.

By the closed form expression of the minimizer of the MSE loss, we can write the weight of the linear model found in MixPro as

$$\boldsymbol{w}_{MixPro}^* = \hat{\boldsymbol{\Sigma}}_{\text{mixed}}^{-1} \hat{\boldsymbol{u}}_{\text{mixed}}, \tag{5}$$

where

$$\hat{\boldsymbol{\Sigma}}_{\text{mixed}} := \frac{1}{n} \sum \tilde{z}_i \tilde{z}_i^\top, \tag{6}$$

and

$$\hat{\boldsymbol{u}}_{\text{mixed}} := \frac{1}{n} \sum \tilde{z}_i \tilde{y}_i. \tag{7}$$

Next, we analyze $\hat{\boldsymbol{\Sigma}}_{\text{mixed}}^{-1}$ and $\hat{\boldsymbol{u}}_{\text{mixed}}$.

Firstly, we decompose $\hat{\boldsymbol{\Sigma}}_{\text{mixed}}$ as

$$\hat{\boldsymbol{\Sigma}}_{\text{mixed}} = (1-s)^2 \frac{1}{n} \sum z_i^s z_i^{s\top} + s^2 \frac{1}{n} \sum z_{j_i}^t z_{j_i}^{t\top} + (1-s)s \frac{1}{n} \sum z_i^s z_{j_i}^{t\top} + (1-s)s \frac{1}{n} \sum z_{j_i}^t z_i^{s\top} \tag{7}$$

$$= \boldsymbol{\Sigma}_{\text{mixed}} + \Delta \boldsymbol{\Sigma}_{\text{mixed}}, \tag{8}$$

where

$$\boldsymbol{\Sigma}_{\text{mixed}} := \mathbb{E}\hat{\boldsymbol{\Sigma}}_{\text{mixed}} = \boldsymbol{\Sigma}_0 + \boldsymbol{A}$$

with

$$\boldsymbol{\Sigma}_0 = \begin{bmatrix} \frac{1}{2}(1-s)^2 & \frac{1}{4}(1-s)s & \frac{1}{4}(1-s)s \\ \frac{1}{4}(1-s)s & \frac{1}{4}s^2 + \frac{1}{4}(1-s)^2 + \frac{1}{4} & \frac{1}{4}(1-s)s \\ \frac{1}{4}(1-s)s & \frac{1}{4}(1-s)s & \frac{1}{2}s^2 \end{bmatrix} \tag{9}$$

and

$$\boldsymbol{A} = \begin{bmatrix} s^2\sigma^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & (1-s)^2\sigma^2 \end{bmatrix}. \tag{10}$$

By applying concentration bounds for Gaussian vectors and matrices (Theorems 6.1 and 6.3 of (Wainwright, 2019)) we can obtain the following. We omit the exact derivation, but the process is similar to, for example, Section B.2.2 in (Xue et al., 2023).

$$\|\Delta\boldsymbol{\Sigma}_{\text{mixed}}\|_2 \leq O(s^2\sigma^2\sqrt{\frac{\log 1/\delta}{m}}) + O(s(1-s)\sigma\sqrt{\frac{\log 1/\delta}{m}}), \quad \text{with probability} \geq 1-\delta \tag{11}$$

Here, as we assume $\sigma = o(1)$, we further obtain $\|\Delta\boldsymbol{\Sigma}_{\text{mixed}}\|_2 = O(\sigma\sqrt{\frac{\log 1/\delta}{m}})$. Since $\exists\epsilon = \Theta(1)$ s.t. $\epsilon < s < 1-\epsilon$, we have

$$\|\boldsymbol{\Sigma}_0\|_2 \leq O(1), \|\boldsymbol{\Sigma}_0^{-1}\|_2 \leq O(1) \tag{12}$$

By applying the classical result for the inverse of perturbation (Demmel, 1992) and combining it with 12, we obtain

$$\|\hat{\boldsymbol{\Sigma}}_{\text{mixed}}^{-1} - \boldsymbol{\Sigma}_0^{-1}\|_2 \leq O(\|\boldsymbol{A} + \Delta\boldsymbol{\Sigma}_{\text{mixed}}\|_2) = O(\sigma\sqrt{\frac{\log 1/\delta}{m}}) \quad \text{with probability} \geq 1-\delta. \tag{13}$$

Next, we examine $\hat{\boldsymbol{u}}_{\text{mixed}}$, which can be further decomposed into

$$\hat{\boldsymbol{u}}_{\text{mixed}} := \frac{1}{n}\sum \tilde{\boldsymbol{z}}_i\tilde{y}_i$$
$$= \boldsymbol{u}_{\text{mixed}} + \Delta\boldsymbol{u}_{\text{mixed}},$$

where

$$\boldsymbol{u}_{\text{mixed}} := \mathbb{E}\boldsymbol{u}_{\text{mixed}} = \begin{bmatrix} \frac{1}{2}(1-s) \\ \frac{1}{2} \\ \frac{1}{2}s \end{bmatrix}.$$

Similarly, we can also bound $\|\Delta\boldsymbol{u}_{\text{mixed}}\|_2$ using Gaussian concentration bounds

$$\|\Delta\boldsymbol{u}_{\text{mixed}}\|_2 \leq O(\frac{\sigma}{\sqrt{m}}\sqrt{\log 1/\delta}) \quad \text{with probability} \geq 1-\delta.$$

$\boldsymbol{\Sigma}_0^{-1}\boldsymbol{u}_{\text{mixed}}$ can be easily calculated

$$\boldsymbol{\Sigma}_0^{-1}\boldsymbol{u}_{\text{mixed}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{14}$$

Now we bound the distance between $\boldsymbol{\Sigma}_0^{-1}\boldsymbol{u}_{\text{mixed}}$ and $\hat{\boldsymbol{\Sigma}}_{\text{mixed}}^{-1}\hat{\boldsymbol{u}}_{\text{mixed}}$. Observe that

$$\hat{\boldsymbol{\Sigma}}_{\text{mixed}}^{-1}\hat{\boldsymbol{u}}_{\text{mixed}} = \left(\boldsymbol{\Sigma}_0^{-1} - (\boldsymbol{\Sigma}_0^{-1} - \hat{\boldsymbol{\Sigma}}_{\text{mixed}}^{-1})\right)\left(\boldsymbol{u}_{\text{mixed}} - (\boldsymbol{u}_{\text{mixed}} - \hat{\boldsymbol{u}}_{\text{mixed}})\right).$$

By Cauchy–Schwarz inequality and realizing that $\|\boldsymbol{u}_{\text{mixed}}\|_2$ and $\|\boldsymbol{\Sigma}_0^{-1}\|_2$ are $\Theta(1)$, we have

$$\|\hat{\boldsymbol{\Sigma}}_{\text{mixed}}^{-1}\hat{\boldsymbol{u}}_{\text{mixed}} - \boldsymbol{\Sigma}_0^{-1}\boldsymbol{u}_{\text{mixed}}\|_2 \leq O(\|\hat{\boldsymbol{\Sigma}}_{\text{mixed}}^{-1} - \boldsymbol{\Sigma}_0^{-1}\|_2) + O(\|\hat{\boldsymbol{u}}_{\text{mixed}} - \boldsymbol{u}_{\text{mixed}}\|_2) \tag{15}$$

$$= O(\sigma\sqrt{\frac{\log(1/\delta)}{m}}) \quad \text{with probability} \geq 1 - \delta.$$

Now we are ready to bound the test loss. We first define the covariance matrix of the target embeddings

$$\boldsymbol{\Sigma_t} = \begin{bmatrix} \sigma^2 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}.$$

By deriving the test loss, leveraging the relation $y = \boldsymbol{\beta}_t^\top \boldsymbol{z}$ where $\boldsymbol{\beta}_t = [0, 1, 1]^\top$ in the target distribution, and incorporating Equation 5, we express the test loss as

$$\mathbb{E}_{(\boldsymbol{z},\boldsymbol{y})\in\text{target dist.}} l(\boldsymbol{z}^\top \boldsymbol{w}_{MixPro}^*, y) = \|\boldsymbol{\Sigma_t}^{1/2}(\boldsymbol{\beta}_t - \hat{\boldsymbol{\Sigma}}_{\text{mixed}}^{-1}\hat{\boldsymbol{u}}_{\text{mixed}})\|^2.$$

Applying Cauchy–Schwarz inequality and incoporating Equation 15 yield

$$\sqrt{\mathbb{E}_{(\boldsymbol{z},\boldsymbol{y})\in\text{target dist.}} l(\boldsymbol{z}^\top \boldsymbol{w}_{MixPro}^*, y)} \leq \|\boldsymbol{\Sigma_t}^{1/2}(\boldsymbol{\beta}_t - \boldsymbol{\Sigma}_0^{-1}\boldsymbol{u}_{\text{mixed}})\| + \|\boldsymbol{\Sigma_t}^{1/2}(\boldsymbol{\Sigma}_0^{-1}\boldsymbol{u}_{\text{mixed}} - \hat{\boldsymbol{\Sigma}}_{\text{mixed}}^{-1}\hat{\boldsymbol{u}}_{\text{mixed}})\|$$

$$\leq \sigma + O(\sqrt{\frac{\log(1/\delta)}{m}}).$$

Therefore

$$\mathbb{E}_{(\boldsymbol{z},\boldsymbol{y})\in\text{target dist.}} l(\boldsymbol{z}^\top \boldsymbol{w}_{MixPro}^*, y) \leq \left(\sigma + O(\sqrt{\frac{\log(1/\delta)}{m}})\right)^2.$$

Consequently, with a probability of at least $1 - O(\frac{1}{\text{poly}\,m})$

$$\mathbb{E}_{(\boldsymbol{z},\boldsymbol{y})\in\text{target dist.}} l(\boldsymbol{z}^\top \boldsymbol{w}_{MixPro}^*, y) \leq \left(\sigma + O(\sqrt{\frac{\log m}{m}})\right)^2 = o(1).$$

A.2.2. ANALYSIS OF PRO^2

We first solve $\boldsymbol{\Pi}_1^*$, which is simply given by $\arg\min_{\boldsymbol{\Pi}_i} \hat{\mathbb{E}}_{(\boldsymbol{z},y)\in\mathcal{E}^s} l(\boldsymbol{\Pi}_i^\top \boldsymbol{z}, y)$ without any restriction. Since we use MSE loss, the closed-form solution is given by

$$(\hat{\mathbb{E}}_{(\boldsymbol{z},y)\in\mathcal{E}^s} \boldsymbol{z}\boldsymbol{z}^\top)^{-1}\hat{\mathbb{E}}_{(\boldsymbol{z},y)\in\mathcal{E}^s} y\boldsymbol{z}$$

As we assume $n \to \infty$, the empirical expectation equals the true expectation, thus we can calculate that

$$\hat{\mathbb{E}}_{(\boldsymbol{z},y)\in\mathcal{E}^s} \boldsymbol{z}\boldsymbol{z}^\top = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & \sigma_\xi^2 \end{bmatrix},$$

and

$$\hat{\mathbb{E}}_{(\boldsymbol{z},y)\in\mathcal{E}^s} y\boldsymbol{z} = \begin{bmatrix} 1/2 \\ 1/2 \\ 0 \end{bmatrix}.$$

Therefore, we obtain that

$$\boldsymbol{\Pi}_1^* = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

For the remaining $\mathbf{\Pi}_i^*$'s ($i \geq 2$), we first write the loss function as follows

$$
\begin{aligned}
\hat{\mathbb{E}}_{(\boldsymbol{z},y)\in\mathcal{E}^s} l(\mathbf{\Pi}_i^\top \boldsymbol{z}, y) &= \mathbb{E}_{(\boldsymbol{z},y)\sim\text{source distribution}} l(\mathbf{\Pi}_i^\top \boldsymbol{z}, y) \\
&= \mathbf{\Pi}_i^\top \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix} \mathbf{\Pi}_i - [1, \ 1, \ 0] \ \mathbf{\Pi}_i + 1.
\end{aligned}
$$

Note that $\mathbf{\Pi}_i^*$'s (for $i \geq 2$) have to be orthogonal to $\mathbf{\Pi}_1^*$, which equals $[1, 1, 0]^\top$, while minimizing the above. Therefore, we substitute $[1, 1, 0]\mathbf{\Pi}_i = 0$ into the above, which reveals that $\mathbf{\Pi}_i^*$ minimizes

$$
\mathbf{\Pi}_i^* = \arg\min_{\mathbf{\Pi}_i} \left(\mathbf{\Pi}_i^\top \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix} \mathbf{\Pi}_i + 1 \right).
$$

Note that the RHS is minimized when $\mathbf{\Pi}_i = \mathbf{0}$. Therefore we conclude that $\mathbf{\Pi}_i^* = \mathbf{0}$ for $i \geq 2$. Given that, and considering $\boldsymbol{v}_3^\top \mathbf{\Pi}_1 = 0$ we obtain

$$
\boldsymbol{v}_3^\top \mathbf{\Pi}^* = \mathbf{0}.
$$

The linear model trained on top of the projected embeddings using the target data has the following expression

$$
\boldsymbol{w}^* = \hat{\mathbb{E}}_{\boldsymbol{z}^t\in\mathcal{E}^t}(\mathbf{\Pi}^{*\top}\boldsymbol{z}^t\boldsymbol{z}^{t\top}\mathbf{\Pi}^*)^\dagger \hat{\mathbb{E}}_{\boldsymbol{z}^t\in\mathcal{E}^t}(\mathbf{\Pi}^{*\top}\boldsymbol{z}^t y^t) \tag{16}
$$

We first examine the projected covariance

$$
\hat{\mathbb{E}}_{(\boldsymbol{z}^t,y)\in\mathcal{E}^t}(\mathbf{\Pi}^{*\top}\boldsymbol{z}^t\boldsymbol{z}^{t\top}\mathbf{\Pi}^*) = \hat{\mathbb{E}}_{(\boldsymbol{z}^t,y)\in\mathcal{E}^t} \begin{bmatrix} (\boldsymbol{z}^{t\top}\mathbf{\Pi}_1^*)^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}
$$

Noticing that $\boldsymbol{z}^{t\top}\mathbf{\Pi}_1^* = by^t + \xi$ where $b \sim \text{Bernoulli}(1/2)$, we can apply concentration bounds of random variables to deduce that $|\hat{\mathbb{E}}(\boldsymbol{z}^{t\top}\mathbf{\Pi}_1^*)^2 - (1/2 + \sigma^2)| \leq o(1)$ with high probability, when m is sufficiently large and $\sigma = o(1)$. Additionally, $\hat{\mathbb{E}}_{\boldsymbol{z}^t\in\mathcal{E}^t}(\mathbf{\Pi}^{*\top}\boldsymbol{z}^t y^t)$ concentrates around $[1/2, 0, \dots]^\top$ with error bounded by $o(1)$ with high probability, similar to what we show in the previous section. Finally, we can further calculate that $\boldsymbol{w}^*$ concentrates around $\begin{bmatrix} \frac{1}{2\sigma^2+1} \\ 0 \\ \vdots \end{bmatrix}$ by

Equation 16. Thus, $\mathbf{\Pi}^* \boldsymbol{w}^*$ concentrates around $\frac{1}{2\sigma^2+1}\mathbf{\Pi}_1^* = \frac{1}{2\sigma^2+1}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$.

Now we are ready to bound the test loss. Combining the above conclusions on concentration and the fact that $\sigma = o(1)$, we obtain

$$
\begin{aligned}
\mathbb{E}_{(\boldsymbol{z},\boldsymbol{y})\in\text{target dist.}} l(\boldsymbol{z}^\top\mathbf{\Pi}^*\boldsymbol{w}^*, y) &= \frac{1}{2}\mathbb{E}_{\xi,y}([0, y, \xi]\mathbf{\Pi}^*\boldsymbol{w}^* - y)^2 + \frac{1}{2}\mathbb{E}_{\xi,y}([0, \xi, y]\mathbf{\Pi}^*\boldsymbol{w}^* - y)^2 \\
&= \frac{1}{2}(\frac{1}{2\sigma^2+1} - 1)^2 + \frac{1}{2}\mathbb{E}_{\xi,y}(\frac{1}{2\sigma^2+1}\xi - y)^2 \pm o(1) \\
&= \frac{1}{2}(\frac{1}{2\sigma^2+1} - 1)^2 + \frac{1}{2}(\frac{\sigma^2}{(2\sigma^2+1)^2} + 1) \pm o(1) \\
&= 0.5 \pm o(1).
\end{aligned}
$$

A.3. Effect of s when $\sigma = \omega(1)$

Here, to understand the benefit of choosing an intermediate s instead of solely using the target data by setting $s = 1$, we consider the case of large noise where $\sigma = \omega(1)$. The analysis is similar to that in Section A.2.1, except that the dominant

term in Equation 11 is now $O(s^2\sigma^2\sqrt{\frac{\log 1/\delta}{m}})$. Additionally, in Equation 15, we compare $\hat{\Sigma}_{\text{mixed}}^{-1}\hat{u}_{\text{mixed}}$ to $\Sigma_{\text{mixed}}^{-1}u_{\text{mixed}}$ instead of $\Sigma_0^{-1}u_{\text{mixed}}$, i.e.,

$$\|\hat{\Sigma}_{\text{mixed}}^{-1}\hat{u}_{\text{mixed}} - \Sigma_{\text{mixed}}^{-1}u_{\text{mixed}}\|_2 \leq O(\|\Delta\Sigma_{\text{mixed}}^{-1}\|_2) + O(\|\Delta u_{\text{mixed}}\|_2) = O(s^2\sigma^2\sqrt{\frac{\log 1/\delta}{m}}).$$

Then, the final step becomes:

$$\sqrt{\mathbb{E}_{(z,y)\in\text{target dist.}}l(z^\top w_{MixPro}^*, y)} \leq \|\Sigma_t^{1/2}(\beta_t - \Sigma_{\text{mixed}}^{-1}u_{\text{mixed}})\| + \|\Sigma_t^{1/2}(\Sigma_{\text{mixed}}^{-1}u_{\text{mixed}} - \hat{\Sigma}_{\text{mixed}}^{-1}\hat{u}_{\text{mixed}})\|.$$

Note that $\Sigma_{\text{mixed}}^{-1}$ and therefore $\Sigma_{\text{mixed}}^{-1}u_{\text{mixed}}$ can be calculated exactly and are independent of m, thus the above takes the form $\psi(s, \sigma) + O(s^2\sigma^2\sqrt{\frac{\log 1/\delta}{m}})$, where $\psi(s, \sigma)$ is a function of only s and σ.

A.4. Proof of Theorem 5.3

Expression of the test loss. Consider the linear model w_{MixPro}^* learned by MixPro with $l()$ being MSE loss with ℓ_2 regularization, i.e., we minimize $\min_w \hat{\mathbb{E}}_{(z,y)\in\mathcal{E}_{\text{mixed}}}(w^\top z - y)^2 + \lambda\|w\|^2$. With assumption 5.2, the test loss on the target distribution achieved by w_{MixPro}^* has the following closed form expression:

$$\mathbb{E}_{(z,y)\in\mathcal{E}^t}(w_{MixPro}^{*\top}z - y)^2 = \left(\frac{q}{\psi_1\psi_3 - \psi_2^2}(\psi_3 - \psi_2(1-s)(2p-1)) - 1\right)^2$$
$$+ \left(\frac{q}{\psi_1\psi_3 - \psi_2^2}(-\psi_2 + \psi_1(1-s)(2p-1))\right)^2$$
$$+ \sigma_1^2\left(\frac{q}{\psi_1\psi_3 - \psi_2^2}(\psi_3 - \psi_2(1-s)(2p-1))\right)^2,$$

where

$$\psi_1 := (1-s)^2(1+\sigma_1^2) + 2s(1-s)q + s^2q(1+\sigma_1^2) - (1-s)^2(1-q) + \lambda$$
$$\psi_2 := (1-s)^2(2p-1) + s(1-s)q(2p-1) - (1-s)^2(1-q)(2p-1)$$
$$\psi_3 := (1-s)^2 + s^2q - (1-s)^2(1-q)(2p-1)^2 + \lambda,$$

with $q := \frac{\lambda}{s^2r+\lambda}$.

We prove the above conclusion below.

Notations. Let $Z \in \mathbb{R}^{d\times n}$ collect inputs in $\mathcal{E}^s$. Let $Z' \in \mathbb{R}^{d\times m}$ collect inputs in $\mathcal{E}^t$. Remember that for each example in $\mathcal{E}^s$, we sample an example from $\mathcal{E}^t$ to construct $\mathcal{E}_{\text{mixed}}$. Let $Z'' \in \mathbb{R}^{d\times n}$ collect the n sampled inputs (i.e., with duplicates) from $\mathcal{E}^t$. Let $\tilde{Z}$ denote the inputs in $\mathcal{E}_{\text{mixed}}$. Then $\tilde{Z} = (1-s)Z + sZ''$. Let Y, Y', Y'' be the corresponding labels. Note that $Y'' = Y$. For convenience, in this section, we use $z, z', z'', \tilde{z}$ for columns in $Z, Z', Z'', \tilde{Z}$, respectively (instead of the notations with t and s used in the main paper). We write $z' = \begin{bmatrix} f' \\ \zeta' \end{bmatrix}$ where $f' \in \mathbb{R}^2$ and $\zeta' \in \mathbb{R}^{d-2}$. Let $F' = [f'_1, f'_2, \ldots, f'_m]$. Let $\mathcal{C}_y$ denote the set of indices of examples in $\mathcal{E}^t$ with label y. Let $\mathcal{S}_j$ denote the set of indices of examples from $\mathcal{E}^s$ that are going to be mixed with z'_j.

Proposition A.1. *The following holds asymptotically almost surely (a.a.s.) in the asymptotic regime we consider*

$$\frac{1}{m}\boldsymbol{F}'\boldsymbol{F}' = \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{f}_i'\boldsymbol{f}_i'^\top = \begin{bmatrix} 1+\sigma_1^2 & 0 \\ 0 & 1 \end{bmatrix} := \boldsymbol{G}_{core} \tag{17}$$

$$\forall i \neq j \in [m], \ \ \boldsymbol{\zeta}_i'^\top \boldsymbol{\zeta}_j' = 0 \tag{18}$$

$$\forall i \in [m], \ \ \|\boldsymbol{\zeta}_i'\| = \sigma_\xi \tag{19}$$

$$\frac{1}{n}\boldsymbol{Z}\boldsymbol{Z}^\top = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{z}\boldsymbol{z}^\top = \begin{bmatrix} \boldsymbol{G}_{spu} & \boldsymbol{0} \\ \boldsymbol{0} & \frac{\sigma_\xi^2}{d-2}\boldsymbol{I}_{d-2} \end{bmatrix}, \ where \ \ \boldsymbol{G}_{spu} := \begin{bmatrix} 1+\sigma_1^2 & 2p-1 \\ 2p-1 & 1 \end{bmatrix}$$

$$\frac{m}{n}\sum_{i \in \mathcal{S}_j}\boldsymbol{z}_i = \begin{bmatrix} y_j' \\ y_j'(2p-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} := \bar{\boldsymbol{z}}_{y_j'} \tag{20}$$

$$\frac{2}{m}\sum_{j \in \mathcal{C}_y}\boldsymbol{f}_j' = \begin{bmatrix} y \\ 0 \end{bmatrix} \tag{21}$$

$$\frac{1}{m}\boldsymbol{F}'\boldsymbol{Y}' = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{22}$$

$$\frac{1}{n}\boldsymbol{Z}\boldsymbol{Y} = \begin{bmatrix} 1 \\ 2p-1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{23}$$

The following derivations are performed based on the equations in Proposition A.1. We index the examples such that the first $m/2$ examples in $\mathcal{E}^t$ have label 1 and the rest have label -1. Since equations 18 and 19 are true, without loss of generality, we assume $\boldsymbol{\zeta}_i' = \sigma_\xi \boldsymbol{e}_i$ where $\boldsymbol{e}_i$ is the i-th standard basis in $\mathbb{R}^{d-2}$.

Corollary A.2. *The following is true*

$$\frac{1}{n}\tilde{\boldsymbol{Z}}\boldsymbol{Y} = \begin{bmatrix} 1 \\ (1-s)(2p-1) \\ s\frac{\sigma_\xi}{m}\boldsymbol{Y}' \\ \boldsymbol{0} \end{bmatrix}.$$

Proof. $\frac{1}{n}\boldsymbol{Z}''\boldsymbol{Y} = \frac{1}{m}\boldsymbol{Z}'\boldsymbol{Y}'$ holds almost surely in the limit we are considering. Then

$$\frac{1}{n}\tilde{\boldsymbol{Z}}\boldsymbol{Y} = (1-s)\frac{1}{n}\boldsymbol{Z}\boldsymbol{Y} + s\frac{1}{n}\boldsymbol{Z}''\boldsymbol{Y}$$

$$= (1-s)\frac{1}{n}\boldsymbol{Z}\boldsymbol{Z} + s\frac{1}{m}\boldsymbol{Z}'\boldsymbol{Y}'$$

$$= (1-s)\begin{bmatrix} 1 \\ 2p-1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + s\begin{bmatrix} 1 \\ 0 \\ \frac{\sigma_\xi}{m}\boldsymbol{Y}' \\ \boldsymbol{0} \end{bmatrix} \quad \text{by the assumption on } \boldsymbol{\zeta}_i\text{'s and Equations 23 and 22,}$$

which completes the proof. □

Lemma A.3. *The following is true*

$$\frac{1}{n}(\boldsymbol{Z}\boldsymbol{Z}''^{\top} + \boldsymbol{Z}''\boldsymbol{Z}^{\top}) = \begin{bmatrix} \boldsymbol{H} & \boldsymbol{N} & \boldsymbol{0} \\ \boldsymbol{N}^{\top} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix},$$

where

$$\boldsymbol{H} := \begin{bmatrix} 2 & 2p-1 \\ 2p-1 & 0 \end{bmatrix} \tag{24}$$

$$\boldsymbol{N} := \frac{\sigma_\xi}{m} \begin{bmatrix} 1 \\ 2p-1 \end{bmatrix} [\underbrace{1 \ 1 \dots \ 1}_{\text{the first } m/2 \text{ elements are } 1} \quad \underbrace{-1 \ -1 \dots \ -1}_{\text{the last } m/2 \text{ elements are } -1} \ ] = \frac{\sigma_\xi}{m} \begin{bmatrix} 1 \\ 2p-1 \end{bmatrix} \boldsymbol{Y}'^{\top} \tag{25}$$

Proof. We first derive the expression of $\frac{1}{n}\boldsymbol{Z}\boldsymbol{Z}''^{\top}$

$$\begin{aligned}
\frac{1}{n}\boldsymbol{Z}\boldsymbol{Z}''^{\top} &= \frac{1}{2} \sum_{y \in \{1,-1\}} \frac{2}{m} \sum_{j \in \mathcal{C}_y} (\frac{m}{n} \sum_{i \in \mathcal{S}_j} \boldsymbol{z}_i) \boldsymbol{z}_j'^{\top} \\
&= \frac{1}{2} \sum_{y \in \{1,-1\}} \bar{\boldsymbol{z}}_y \frac{2}{m} \sum_{j \in \mathcal{C}_y} \boldsymbol{z}_j'^{\top} \quad \text{by equation 20} \\
&= \frac{1}{2} \sum_{y \in \{1,-1\}} \bar{\boldsymbol{z}}_y \frac{2}{m} \sum_{j \in \mathcal{C}_y} [\boldsymbol{f}_j'^{\top} \ \boldsymbol{\zeta}_j'^{\top}] \\
&= \frac{1}{2} \sum_{y \in \{1,-1\}} \bar{\boldsymbol{z}}_y [y \ 0 \ \frac{2}{m} \sum_{j \in \mathcal{C}_y} \sigma_\xi \boldsymbol{e}_j^{\top}] \quad \text{by equation 21 and the assumption about } \boldsymbol{\zeta}_j' \\
&= \begin{bmatrix} 1 & 0 & \frac{\sigma_\xi}{2} & \frac{\sigma_\xi}{2} & \cdots & -\frac{\sigma_\xi}{2} & -\frac{\sigma_\xi}{2} & \cdots & 0 & \cdots \\ 2p-1 & 0 & (2p-1)\frac{\sigma_\xi}{2} & (2p-1)\frac{\sigma_\xi}{2} & \cdots & -(2p-1)\frac{\sigma_\xi}{2} & -(2p-1)\frac{\sigma_\xi}{2} & \cdots & 0 & \cdots \\ 0 & \cdots & & & & & & & 0 & \cdots \\ \vdots & & & & & & & & \vdots & \ddots \\ 0 & \cdots & & & & & & & 0 & \cdots \end{bmatrix}.
\end{aligned}$$

The expression of $\frac{1}{n}\boldsymbol{Z}''\boldsymbol{Z}^{\top}$ is just the transpose of the above. Adding the two expressions together completes the proof. □

Lemma A.4. *The following is true*

$$\frac{1}{n}\boldsymbol{Z}''\boldsymbol{Z}''^{\top} = \begin{bmatrix} \boldsymbol{G}_{core} & \frac{\sigma_\xi}{m}\boldsymbol{F}' & \boldsymbol{0} \\ \frac{\sigma_\xi}{m}\boldsymbol{F}'^{\top} & \frac{\sigma_\xi^2}{m}\boldsymbol{I}_m & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}.$$

Proof. $\frac{1}{n}\boldsymbol{Z}''\boldsymbol{Z}''^{\top} = \frac{1}{m}\boldsymbol{Z}'\boldsymbol{Z}'^{\top}$ almost surely in the limit we are considering. Then

$$\begin{aligned}
\frac{1}{n}\boldsymbol{Z}''\boldsymbol{Z}''^{\top} &= \frac{1}{m}\boldsymbol{Z}'\boldsymbol{Z}'^{\top} \\
&= \frac{1}{m} \begin{bmatrix} \boldsymbol{F}' \\ \sigma_\xi \boldsymbol{I}_m \\ \boldsymbol{0} \end{bmatrix} [\boldsymbol{F}'^{\top} \ \ \sigma_\xi \boldsymbol{I}_m \ \ \boldsymbol{0}^{\top}] \\
&= \frac{1}{m} \begin{bmatrix} \boldsymbol{F}'\boldsymbol{F}'^{\top} & \sigma_\xi\boldsymbol{F}' & \boldsymbol{0} \\ \sigma_\xi\boldsymbol{F}'^{\top} & \sigma_\xi^2\boldsymbol{I}_m & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{G}_{core} & \frac{\sigma_\xi}{m}\boldsymbol{F}' & \boldsymbol{0} \\ \frac{\sigma_\xi}{m}\boldsymbol{F}'^{\top} & \frac{\sigma_\xi^2}{m}\boldsymbol{I}_m & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}
\end{aligned}$$

□

Define $M := \frac{1}{n}\tilde{Z}\tilde{Z}^\top + \lambda I_d$. Then

$$M = \frac{(1-s)^2}{n}ZZ^\top + \frac{s(1-s)}{n}ZZ''^\top + \frac{s(1-s)}{n}Z''Z^\top + \frac{s^2}{n}Z''Z''^\top + \lambda I_d$$

$$= (1-s)^2 \begin{bmatrix} G_{spu} & 0 \\ 0 & \frac{\sigma_\xi^2}{d-2}I_{d-2} \end{bmatrix} + s(1-s)\begin{bmatrix} H & N & 0 \\ N^\top & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + s^2 \begin{bmatrix} G_{core} & \frac{\sigma_\xi}{m}F' & 0 \\ \frac{\sigma_\xi}{m}F'^\top & \frac{\sigma_\xi^2}{m}I_m & 0 \\ 0 & 0 & 0 \end{bmatrix} + \lambda I_d$$

$$= \begin{bmatrix} A & B & 0 \\ B^\top & D & 0 \\ 0 & 0 & \frac{\sigma_\xi^2}{d-2}I_{d-2-m} + \lambda I_{d-2-m} \end{bmatrix},$$

where

$$A := (1-s)^2 G_{spu} + s(1-s)H + s^2 G_{core} + \lambda I_2$$

$$B := s(1-s)N + s^2\frac{\sigma_\xi}{m}F'$$

$$D := \Big((1-s)^2\frac{\sigma_\xi^2}{d-2} + s^2\frac{\sigma_\xi^2}{m} + \lambda\Big)I_m = (s^2 r + \lambda)I_m$$

The inverse of M is

$$M^{-1} = \begin{bmatrix} (P/D)^{-1} & -(P/D)^{-1}BD^{-1} & 0 \\ -D^{-1}B^\top(P/D)^{-1} & D^{-1} + D^{-1}B^\top(P/D)^{-1}BD^{-1} & 0 \\ 0 & 0 & \frac{1}{\frac{\sigma_\xi^2}{d-2}+\lambda}I_{d-2-m} \end{bmatrix}, \quad (26)$$

where $P := \begin{bmatrix} A & B \\ B^\top & D \end{bmatrix}$ and $P/D := A - BD^{-1}B^\top$ is the Schur complement of D in P.

We first derive the expressions of $F'N^\top$.

$$F'N^\top = \frac{\sigma_\xi}{m}F'Y'[1 \ \ 2p-1]$$

$$= \sigma_\xi\begin{bmatrix} 1 & 2p-1 \\ 0 & 0 \end{bmatrix} \quad \text{by Equation 22} \quad (27)$$

Now we derive $BB^\top$.

$$BB^\top = s^2(1-s)^2 NN^\top + s^4\frac{\sigma_\xi^2}{m^2}F'F'^\top + s^3(1-s)\frac{\sigma_\xi}{m}(NF'^\top + F'N^\top)$$

$$= s^2(1-s)^2 rJ + s^4 rG_{core} + s^3(1-s)rH \quad \text{by Equations 25, 17 and 27 ,} \quad (28)$$

where $J := \begin{bmatrix} 1 & 2p-1 \\ 2p-1 & (2p-1)^2 \end{bmatrix}$.

Let $q := \frac{\lambda}{s^2 r + \lambda}$. Then

$$P/D = (1-s)^2 G_{spu} + s(1-s)\frac{\lambda}{s^2 r + \lambda}H + s^2\frac{\lambda}{s^2 r + \lambda}G_{core} - (1-s)^2\frac{s^2 r}{s^2 r + \lambda}J + \lambda I_2$$

$$= (1-s)^2 G_{spu} + s(1-s)qH + s^2 q G_{core} - (1-s)^2(1-q)J + \lambda I_2$$

$$= \begin{bmatrix} \psi_1 & \psi_2 \\ \psi_2 & \psi_3 \end{bmatrix},$$

where

$$\psi_1 := (1-s)^2(1+\sigma_1^2) + 2s(1-s)q + s^2 q(1+\sigma_1^2) - (1-s)^2(1-q) + \lambda$$

$$\psi_2 := (1-s)^2(2p-1) + s(1-s)q(2p-1) - (1-s)^2(1-q)(2p-1)$$

$$\psi_3 := (1-s)^2 + s^2 q - (1-s)^2(1-q)(2p-1)^2 + \lambda.$$

Then

$$(\boldsymbol{P}/\boldsymbol{D})^{-1} = \frac{1}{\psi_1\psi_3 - \psi_2^2} \begin{bmatrix} \psi_3 & -\psi_2 \\ -\psi_2 & \psi_1 \end{bmatrix}.$$

Let $\boldsymbol{w}$ be the minimizer of the objective in 2, which has the closed form expression $\boldsymbol{M}^{-1}\frac{1}{n}\tilde{\boldsymbol{Z}}\boldsymbol{Y}$. We are ready to derive the elements in $\boldsymbol{w}$ using Corollary A.2 and 26. But before that, we first derive the following

$$\begin{aligned}
\boldsymbol{B}\boldsymbol{Y}' &= s(1-s)\boldsymbol{N}\boldsymbol{Y}' + s^2\frac{\sigma_\xi}{m}\boldsymbol{F}'\boldsymbol{Y}' \\
&= s(1-s)\sigma_\xi \begin{bmatrix} 1 \\ 2p-1 \end{bmatrix} + s^2\sigma_\xi \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
&= s\sigma_\xi \begin{bmatrix} 1 \\ (1-s)(2p-1) \end{bmatrix}.
\end{aligned}$$

Now we can drive the first two elements in $\boldsymbol{w}$ with the above and Corollary A.2 and 26

$$\begin{aligned}
\boldsymbol{w}_{1:2} &= (\boldsymbol{P}/\boldsymbol{D})^{-1} \begin{bmatrix} 1 \\ (1-s)(2p-1) \end{bmatrix} - (\boldsymbol{P}/\boldsymbol{D})^{-1}\boldsymbol{B}\boldsymbol{D}^{-1}s\frac{\sigma_\xi}{m}\boldsymbol{Y}' \\
&= q(\boldsymbol{P}/\boldsymbol{D})^{-1} \begin{bmatrix} 1 \\ (1-s)(2p-1) \end{bmatrix} \\
&= \frac{q}{\psi_1\psi_3 - \psi_2^2} \begin{bmatrix} \psi_3 - \psi_2(1-s)(2p-1) \\ -\psi_2 + \psi_1(1-s)(2p-1) \end{bmatrix},
\end{aligned} \tag{29}$$

as well as the next m entries

$$\begin{aligned}
\boldsymbol{w}_{3:m+2} &= -\boldsymbol{D}^{-1}\boldsymbol{B}^\top(\boldsymbol{P}/\boldsymbol{D})^{-1} \begin{bmatrix} 1 \\ (1-s)(2p-1) \end{bmatrix} + \boldsymbol{D}^{-1}s\frac{\sigma_\xi}{m}\boldsymbol{Y}' + \boldsymbol{D}^{-1}\boldsymbol{B}^\top(\boldsymbol{P}/\boldsymbol{D})^{-1}\boldsymbol{B}\boldsymbol{D}^{-1}s\frac{\sigma_\xi}{m}\boldsymbol{Y}' \\
&= \frac{\sigma_\xi}{m} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{bmatrix},
\end{aligned}$$

where

$$\begin{aligned}
c_1 &:= \psi_3 - \psi_2(1-s)(2p-1) \\
c_2 &:= -\psi_2 + \psi_1(1-s)(2p-1) \\
h_i &:= \left(\frac{s}{s^2 r + \lambda} - \frac{q}{s^2 + \lambda}\frac{1}{\psi_1\psi_3 - \psi_2^2}s(1-s)(c_1 + (2p-1)c_2)\right)y_i - s^2(f_{i,1}c_1 + f_{i,2}c_2).
\end{aligned}$$

Note that the remaining entries in $\boldsymbol{w}$ are all zero.

With simple calculation, we can obtain the MSE test loss on the target distribution

$$\mathbb{E}_{(\boldsymbol{z},y)\in\mathcal{E}^t}(\boldsymbol{w}^\top\boldsymbol{z} - y)^2 = (w_1 - 1)^2 + w_2^2 + \sigma_1^2 w_1^2 + \frac{\sigma_\xi^2}{d-2}\sum_{i=3}^{m+2} w_i^2.$$

Let's look at the last term, which can be written as

$$\frac{\sigma_\xi^2}{d-2}\sum_{i=3}^{m+2} w_i^2 = \frac{\sigma_\xi^2}{d-2}r\frac{1}{m}\sum_{i=i}^{m} h_i^2.$$

Since $\mathbb{E}h$ is a constant and by the law of large number $\frac{1}{m}\sum_{i=i}^{m} h_i^2$ converges almost surely to the expected value, $\frac{1}{m}\sum_{i=i}^{m} h_i^2$ is also a constant. Note that r is a constant, too. Then the RHS converges to 0 because $\frac{\sigma_\xi^2}{d-2} = \frac{rm}{d-2} \to 0$ by our assumptions.

Combining the above and equations 29, we obtain the following expression of

$$\mathbb{E}_{(\boldsymbol{z},y)\in\mathcal{E}^t}(\boldsymbol{w}^\top \boldsymbol{z} - y)^2 = \left(\frac{q}{\psi_1\psi_3 - \psi_2^2}(\psi_3 - \psi_2(1-s)(2p-1)) - 1\right)^2 + \left(\frac{q}{\psi_1\psi_3 - \psi_2^2}(-\psi_2 + \psi_1(1-s)(2p-1))\right)^2$$
$$+ \sigma_1^2 \left(\frac{q}{\psi_1\psi_3 - \psi_2^2}(\psi_3 - \psi_2(1-s)(2p-1))\right)^2.$$

which completes the proof.

B. Experimental Details

B.1. Datasets

Waterbirds. In the WaterBirds dataset (Sagawa et al., 2019), the task is to classify images of birds as either landbirds or waterbirds, where each bird image has a background of either $land$ or $water$. The dataset introduces a challenge as the label is spuriously correlated with the image background. Among the 4,795 training samples, about 95% exhibit this spurious correlation, with the numbers of the four combinations of background and label—(land background, landbird), (water background, landbird), (land background, waterbird), and (water background, waterbird)—being 3498, 184, 56, and 1057, respectively.

UrbanCars. This dataset, constructed by (Li et al., 2023), features multiple spurious correlations. The task is classifying images as either urban cars or country cars. Each image has one background (BG) and one co-occurring object (CoObj). The BG is selected from either urban or country backgrounds, and the CoObj is selected from either urban or country objects. For each class, images with common BG and CoObj constitute 90.25%, images with uncommon BG and common CoObj, or with common BG and uncommon CoObj, constitute 4.75%, and images with both uncommon BG and CoObj constitute 0.25%. This dataset presents a challenge due to multiple spurious correlations/shortcuts.

bFFHQ. This dataset, constructed by (Kim et al., 2021), is the gender-biased version of the Flickr-Faces-HQ (FFHQ) dataset (Karras et al., 2019). The task is to classify whether a given facial image is 'young' (aged 10–29) or 'old' (aged 40–59). The label 'young' is highly correlated with the attribute 'women', and 'old' is highly correlated with the attribute 'man'.

Camelyon17. This dataset, included in the WILDS benchmark by (Koh et al., 2021), comprises medical images collected from various hospitals, leading to naturally occurring distribution shifts due to the differences in data collection processes. In the training set, the images are patches taken from 30 WSIs (whole-slide images), with 10 WSIs from each of the 3 hospitals in the training set. In its out-of-distribution test set, the images are patches taken from 10 WSIs from a different hospital, which was chosen because its patches were the most visually distinctive. These WSIs are also distinct from those in the training set.

PACS. This dataset was constructed by (Li et al., 2017) and is included in the DOMAINBED benchmark by (Gulrajani & Lopez-Paz, 2020). The task is image classification across seven classes. It features images from four domains: Art (A), Cartoons (C), Photos (P), and Sketches (S).

VLCS. This dataset was constructed by (Fang et al., 2013) and is included in the DOMAINBED benchmark by (Gulrajani & Lopez-Paz, 2020). The task is image classification across five classes. It features images from four photographic domains: Caltech101 (C), LabelMe (L), SUN09 (S), and VOC2007 (V).

Terra Incognita. This dataset was constructed by (Beery et al., 2018) and is included in the DOMAINBED benchmark by (Gulrajani & Lopez-Paz, 2020). The task is image classification across ten classes. It contains photographs of wild animals taken by camera traps at four different locations: L100, L38, L43, and L46.

Office-Home. This dataset was constructed by (Venkateswara et al., 2017) and is included in the DOMAINBED benchmark by (Gulrajani & Lopez-Paz, 2020). The task is image classification across 65 classes. It features images from four domains: Art (A), Clipart (C), Product (P) and Real (R).

B.2. Details of data distribution

WaterBirds, bFFHQ and UrbanCars exhibit subpopulation shifts. In these datasets, certain subpopulations are significantly underrepresented in the source distribution. Following (Chen et al., 2023), we examine two types of target distribution that

Table 1. Hyperparameter range for each method. m.s. represents 'method-specific'.

	PRO^2	DFR	Mixup	Teney et al. (2022)	MixPro
lr	$\{0.1, 0.01, 0.001\}$				
wd	$\{0.1, 0.01, 0.001\}$				
m.s.	$d \in \{1, 2^2, 2^4, 2^6, 2^8, 2^{10}\}$	None	$\alpha \in \{0.2, 0.4, 2^2, 2^3, 2^5\}$	$\lambda \in \{5e^{-3}, 1e^{-2}, 0.1, 1, 5\}$	$s \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$

differ from the source distribution: (1) Minority (M), which only contains the minority subpopulations, and (2) Balanced (B), which contains an equal number of examples from each subpopulation.

PACS, VLCS, Terra Incognita, and Office-Home are domain generalization datasets, each consisting of data collected from multiple distinct environments/domains. Following prior work (Gulrajani & Lopez-Paz, 2020; Li et al., 2017; Fang et al., 2013; Beery et al., 2018; Venkateswara et al., 2017), we select one domain as the target distribution, with the remaining domains forming the source distribution. Below are the exact setups, following the common settings used in the aforementioned works. For PACS, we let P, A, and C constitute the source domain, with S as the target domain; for VLCS, we let V, L, and C constitute the source domain, with S as the target domain; for Terra Incognita, we let L100, L38, and L43 constitute the source domain, with L46 as the target domain; for Office-Home, we let A, C, and P constitute the source domain, with R as the target domain.

Camelyon17 is also a domain generalization dataset, where the source and target distributions consist of data collected from different hospitals.

B.3. Training details

Finetuning of ResNet50 on source data. On WaterBirds, bFFHQ, and UrbanCars, we train the model on the source data using the SGD optimizer with a batch size of 128, a learning rate of 0.001 and weight decay of 0.0001 for 100 epochs. On Camelyon17, we use the same setting except we train for 20 epochs. On PACS, VLCS, Office-Home, and Terra Incognita, we use the same setting but train for 50 epochs.

Training details of all methods. For all methods, following (Chen et al., 2023), we employ the Adam optimizer (Kingma & Ba, 2014) with a batch size of 64 and train for 100 epochs. On Camelyon17, due to the large size of the source data, we randomly subsample 20,000 examples from the source data to mix with the target data when applying our method, MixPro. This turns out to be sufficient to achieve very high performance.

B.4. Details of hyperparameter tuning

Table 1 displays the hyperparameter ranges we use for tuning each method. For every method, we tune the learning rate and weight decay, along with method-specific hyperparameters. It's important to note that the ranges we selected here are all based on the original paper, covering the ranges or the exact values considered in the original study. For (Teney et al., 2022), in addition to the regularization strength λ, there is another hyperparameter, which is the number of diverse models trained on top of the embeddings. We set this number to 96, as (Teney et al., 2022) demonstrates that a higher number of models leads to better performance, and 96 is the maximum value they used.

B.5. Details of the toy experiments in Figure 1

We generate synthetic data that is a higher-dimensional generalization of that analyzed in Section 5.1, to demonstrate the generality of our results. Formally, each example $z \in \mathbb{R}^d$ is given by:

$$z = yv + \xi,$$

where v is selected from $\{v_1, v_2\}$ in the source distribution and from $\{v_2, v_3\}$ in the target distribution. The noise ξ is a random Gaussian vector sampled from the space orthogonal to the features, formally $\xi \sim \mathcal{N}(0, \frac{\sigma^2}{d-2}(I - VV^\top))$, where $V = [v_1 \ v_2]$ in the source domain and $V = [v_2 \ v_3]$ in the target domain.

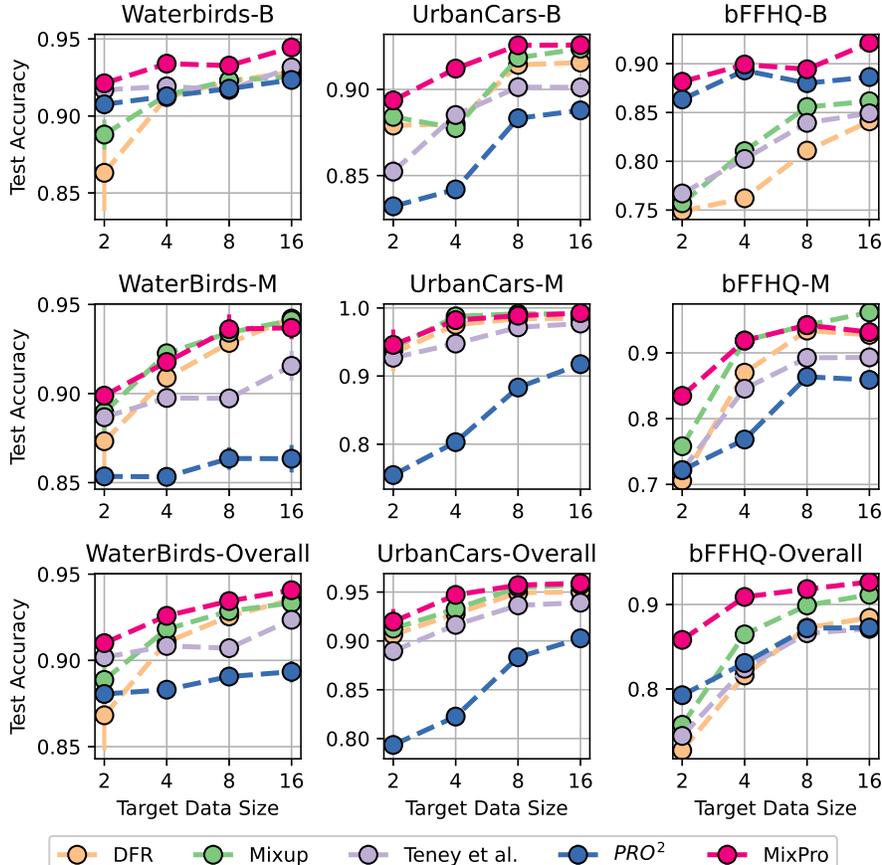In our experiments, we set $\sigma = 4$, $d = 8$ and $n = 4000$.

Figure 6. Detailed results on the three subpopulation shift datasets where we use the SWAG-pretrained VIT-L/16 as the backbone. The top section shows the results when the target distribution is balanced. The middle section displays the results when the target distribution contains only subpopulations that are minorities in the source data. The bottom section shows the average results over these two cases. We note that these bottom plots are the same as the corresponding plots in Figure 4.

C. Additional Experimental Results

C.1. Detailed results on subpopulation shift datasets

As mentioned in Section B.2, for each subpopulation shift dataset, we examined two types of target distributions: Balanced (B) and Minority (M). The complete results are presented in Figure 6. We observe that MixPro consistently performs the best on each dataset, while other methods may perform well in some cases but fail in others. Overall, as indicated by the bottom row of plots, MixPro demonstrates the best performance.

C.2. Results for ResNet50

We also present results for cases where we use the ImageNet-pretrained ResNet50, fine-tuned on the source data, as the backbone. Fine-tuning on the source data aligns with the practices in (Rosenfeld et al., 2022; Kirichenko et al., 2022) and is essential for achieving good performance; otherwise, the performance would be significantly lower, unlike with the SWAG-pretrained ViT.

Figure 7 presents the results obtained by using an additional validation set for fine-tuning, while Figure 8 shows the results from using cross-validation with the few given target data. We observe that, similar to the conclusions drawn in Figures 4 and 5, some methods may perform comparably to MixPro on certain datasets but are surpassed on others.
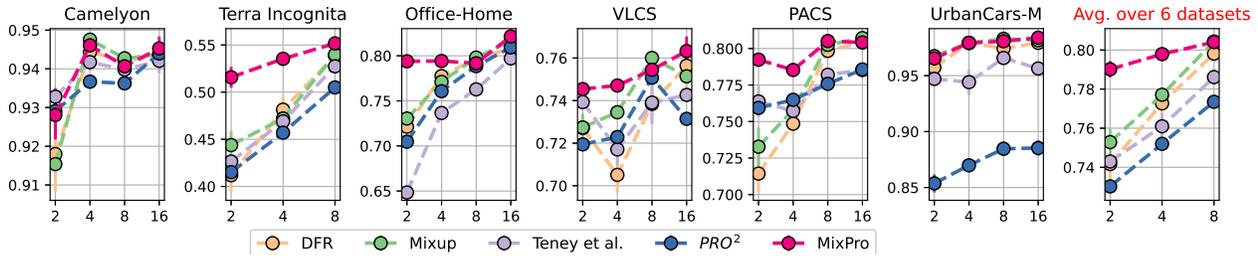
Figure 7. Results are presented for cases where we use ResNet50, pretrained on ImageNet and fine-tuned on the source data, as the backbone model. Here we use the additional validation set for hyperparameter tuning. We evaluate all methods across 6 datasets. We see that MixPro is the only one that consistently delivers the best performance.
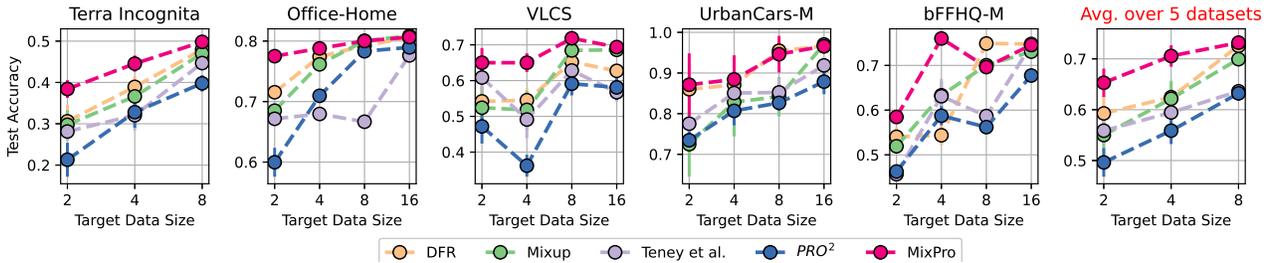

Figure 8. Results are presented for cases where we use ResNet50, pretrained on ImageNet and fine-tuned on the source data, as the backbone model. Here we use cross-validation with the few given target data for hyperparameter tuning.
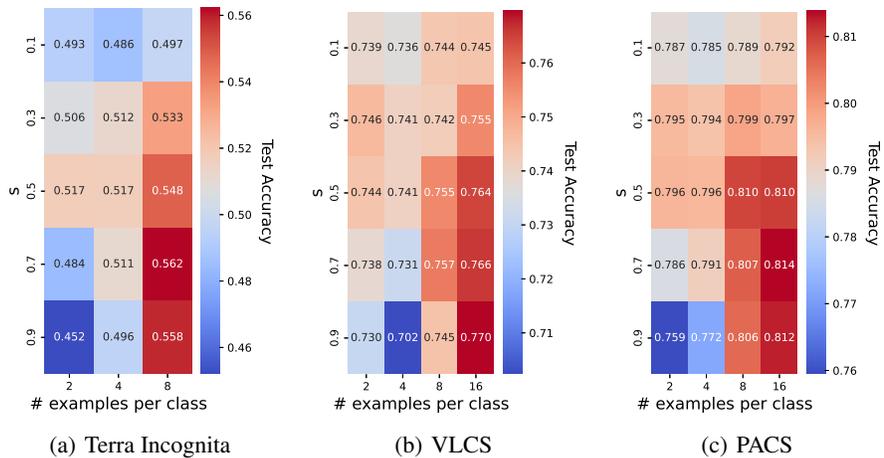

Figure 9. Target data Size and the selection of the mixing weight s. Recall that s represents the weight assigned to the target data. Consistent with our intuition and the analysis presented in Sections 5.2 and 3, we observe that with a smaller target data size, the optimal value of s decreases.

C.3. Mixing weight and target dataset size

Here we demonstrate how the size of the target data influences the choice of s. As depicted in Fig 3(b), our theoretical analysis in Section 5.2 has already indicated that with fewer target data points, a smaller s is preferable. This means placing more emphasis on the source data to counteract the noise. We confirm this intuition on real datasets. In Figure 9, we show the test accuracy under different values of s and target data sizes, for the results with ResNet50 as the backbone on three datasets.

dataset	P-Mixup	MixPro
PACS	81.97	**84.50**
Office-Home	85.05	**90.84**

Table 2. Comparison between MixPro and (Zhu et al., 2022) on Office-Home and PACS.

	LCCS (1-shot)	MixPro (1-shot)	LCCS (5-shot)	MixPro (5-shot)	LCCS (10-shot)	MixPro (10-shot)
Sketch	72.5	**75.4**	76.7	**77.1**	79.4	**79.5**

Table 3. Comparison between MixPro and (Zhang et al., 2022) on PACS.

C.4. Comparisons with (Zhu et al., 2022) and (Zhang et al., 2022)

(Zhu et al., 2022) employs both image-level and representation-level mixup, with the mixing ratio progressively adjusted, and optimization performed using the MAML meta learning framework. Below, we present a comparison between their method and ours.

- **MixPro is data-efficient and performs better.** Our method is more data-efficient than (Zhu et al., 2022), achieving better performance when given very few target data. We conducted new experiments on PACS (with the target domain being S) and Office Home (with the target domain being R) using ResNet18, employing the same setup as in (Zhu et al., 2022) (which uses only a subset of the classes), and reported our method's mean Average Precision (mAP) — the metric used in their paper — comparing it with theirs in Table 2. We observed that our method indeed outperforms theirs on both datasets. This could be attributed to their approach of updating the encoder rather than freezing it, which can potentially be harmful as the entire network is more prone to overfitting the very few target examples, compared to just the linear layer.

- **MixPro is lightweight.** A few factors contribute to their method being significantly more costly than ours: (1) They update the encoder and employ image-level mixup, while the cost of our method is equivalent to just linear probing. (2) Their use of MAML, a meta learning framework, involves computing the gradient of a gradient, which is very expensive even with approximations. Despite being considerably more cost-effective, our method still achieves superior performance.

LCCS (Zhang et al., 2022) adapts the normalization layers leveraging both source data and the information from the few target data. Below, we present a comparison between their method and ours.

- **MixPro is data-efficient and performs better.** Our method is more data-efficient than (Zhang et al., 2022) and achieves better performance given few target data. We conducted new experiments on the PACS dataset using ResNet18, following the same setup as in (Zhang et al., 2022). The performances are compared in Table 3. MixPro outperforms theirs with the margin being especially significant when the amount of target data is small. The potential reason is that batch normalization, known to speed up training, also increases the risk of overfitting given few target data, thereby potentially degrading representation quality. In contrast, our method only trains the last linear layer, posing smaller risk of overfitting.

- **MixPro is lightweight.** Our method is also more efficient compared to (Zhang et al., 2022). Their method involves performing SVD on the matrix of channel-wise features, which is extremely expensive for very large models, whereas our method solely trains a linear layer on representations.