Out of the Ordinary: Spectrally Adapting Regression for Covariate Shift

Benjamin Eyre [1] **Elliot Creager** [2 3] **David Madras** [4] **Vardan Papyan** [3 5] **Richard Zemel** [1]

Abstract

Designing deep neural network classifiers that perform robustly on distributions differing from the available training data is an active area of machine learning research. However, out-of-distribution generalization for regression—the analogous problem for modeling continuous targets—remains relatively unexplored. To tackle this problem, we return to first principles and analyze how the closed-form solution for Ordinary Least Squares (OLS) regression is sensitive to covariate shift. We characterize the out-of-distribution risk of the OLS model in terms of the eigenspectrum decomposition of the source and target data. We then use this insight to propose a method called Spectral Adapted Regressor (SpAR) for adapting the weights of the last layer of a pre-trained neural regression model to perform better on input data originating from a different distribution. We demonstrate how this lightweight spectral adaptation procedure can improve out-of-distribution performance for synthetic and real-world datasets.

1. Introduction

Despite their groundbreaking benchmark performance on many tasks—from image recognition and natural language understanding to disease detection (Balagopalan et al., 2020; Krizhevsky et al., 2017; Devlin et al., 2019)—deep neural networks (DNNs) tend to underperform when confronted with data that is dissimilar to their training data (Geirhos et al., 2020; D'Amour et al., 2022; Arjovsky et al., 2019; Koh et al., 2021).

Understanding and addressing *distribution shift* is critical for the real-world deployment of machine learning (ML) systems. For instance, datasets from the WILDS benchmark (Koh et al., 2021) provide real-world case studies suggesting that poor performance at the subpopulation level can have dire consequences in crucial applications such as monitoring toxicity of online discussions, or tumor detection from medical images. Furthermore, DeGrave et al. (2021) demonstrated that models trained to detect COVID-19 from chest X-Rays performed worse when evaluated on data gathered from hospitals that were not represented in the training distribution. Unfortunately, poor out-of-distribution (OOD) generalization remains a key obstacle to broadly deploying ML models in a safe and reliable way.

While work towards remedying these OOD performance issues has been focused on classification, predicting continuous targets under distribution shift has received less attention. In this paper, we present a lightweight method for updating the weights of a pre-trained regression model (typically a neural network, in which case only the final layer is updated). This method is motivated by a theoretical analysis that yields a concrete reason, which we call *Spectral Inflation*, to explain why regressors may fail under covariate shift, a specific form of distribution shift. Through experiments on synthetic, tabular, and image data, we show that our novel post-processing method consistently improves the OOD performance of regression models. We release the code for these experiments at `https://github.com/btleyre/spar`.

2. Background

Distribution shift problems involve training on inputs X and target labels Y sampled from $P(X, Y)$, then evaluating the resulting model on a distinct distribution $Q(X, Y)$. Several learning frameworks consider different forms of distribution shift, depending on the structure of P and the degree of prior knowledge about Q that is available. Within this work, we assume access to a set of *unlabelled* examples from the target distribution Q.

We also assume the distribution shift is due to *covariate shift*, where the conditional distribution over the evaluation data $Q(Y|X)$ is equal to the conditional distribution over the training data $P(Y|X)$, but the input marginals $P(X)$ and $Q(X)$ differ. This broadly studied assumption (Sugiyama et al., 2007; Gretton et al., 2009; Ruan et al., 2022) states

[1]Columbia University, New York, USA [2]University of Waterloo, Waterloo, Canada [3]Vector Institute, Toronto, Canada [4]Google Research [5]University of Toronto, Toronto, Canada. Correspondence to: Benjamin Eyre <beneyre@cs.columbia.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

that the sample will have the same relationship to the label in both distributions.

3. Robust Regression by Spectral Adaptation

Least-squares regression has a known closed-form solution that minimizes the training loss, and yet this solution is not robust to covariate shift. In this section we show *why* this is the case by characterizing the OOD risk in terms of the eigenspectrum of the source and (distribution-shifted) target data. We then use insights from our theoretical analysis to derive a practical post-processing algorithm that uses unlabeled target data to adapt the weights of a regressor previously pre-trained on labeled source data. The adaptation is done in the spectral domain by first identifying subspaces of the target and source data that are misaligned, then projecting out the pre-trained regressor's components along these subspaces. We call our method **Spectral Adapted Regressor (SpAR)**.

3.1. Analyzing OLS Regression Under Covariate Shift

We begin with the standard Ordinary Least Squares (OLS) data generating process (Murphy, 2022). Rows of the input data matrix, $X \in \mathbb{R}^{N\times D}$, are i.i.d. samples from an unknown distribution P over $\mathbb{R}^D$; these can be any representation, including one learned by a DNN from training samples. The rows of the evaluation input data, $Z \in \mathbb{R}^{M\times D}$, are generated using a different distribution Q over $\mathbb{R}^D$. Analyzing final layer representations is useful as DNN architectures typically apply linear models to these to make predictions. Targets depend on X and w^*, a labeling vector in $\mathbb{R}^D$, and a noise term[1] ϵ. The targets associated with the test data Z use the same true labeling vector w^* but do not include a noise term as it introduces irreducible error:

$$X \sim P^N, \quad Y_X = Xw^* + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I), \qquad (1)$$

$$Z \sim Q^M, \quad Y_Z = Zw^*.$$

The estimated regressor $\hat{w}$ that minimizes the expected squared error loss has the following form (Murphy, 2022), using $X^\dagger$, the Moore-Penrose Pseudoinverse of X, and its Singular Value Decomposition (SVD), $X^\dagger = V_X D_X^\dagger U_X^\top$:

$$\arg\min_w \mathbb{E}[\|Y_X - Xw\|_2^2] = \hat{w} = X^\dagger Y_X = V_X D_X^\dagger U_X^\top Y_X. \qquad (2)$$

We refer to $\hat{w}$ as the "OLS regressor" or "pseudoinverse solution". Our primary expression of interest will be the expected loss of $\hat{w}$ under covariate shift, which is the squared error between the true labels Y_Z and the values predicted by our estimator $\hat{w}$. Specifically, we will analyze the expression:

$$\mathrm{Risk}_{\mathrm{OLS-OOD}}(\hat{w}) = \mathbb{E}[\|Y_Z - Z\hat{w}\|_2^2]. \qquad (3)$$

In addition to using the SVD of $X = U_X S_X V_X^\top$, we can also use the SVD of the target data $Z = U_Z S_Z V_Z^\top$. We define $\lambda_{x,i}, \lambda_{z,i}$ to be the i^{th} singular values of X and Z, respectively, and $e_{x,i}, e_{z,i}$ their corresponding unit-length right singular vectors. We will also refer to $\lambda_{x,i}^2, \lambda_{z,i}^2$ and $e_{x,i}, e_{z,i}$ as eigenvalues/eigenvectors, as they comprise the eigenspectrum of the uncentered covariance matrices $X^\top X$ and $Z^\top Z$. We use the operator $\mathrm{Rows}()$ to represent the set containing the rows of a matrix. The OOD risk of $\hat{w}$ is presented in the following theorem in terms of interaction between the eigenspectra of X and Z:

Theorem 3.1. *Assuming the data generative procedure defined in Equations 1, and that $w^* \in \mathrm{Span}(\mathrm{Rows}(X))$ and $\mathrm{Rows}(Z) \subset \mathrm{Span}(\mathrm{Rows}(X))$, the OOD squared error loss of the estimator $\hat{w} = X^\dagger Y$ is equal to:*

$$\mathbb{E}[\|Y_Z - Z\hat{w}\|_2^2] = \sigma^2 \sum_{i=1}^{D}\sum_{j=1}^{D} \frac{\lambda_{z,j}^2}{\lambda_{x,i}^2} \langle e_{x,i}, e_{z,j}\rangle^2 \mathbb{1}[\lambda_{x,i} > 0].$$

This decomposition of the OLS test loss reveals key components of the loss: if the samples in Z present a large amount of variance along the vector $e_{z,j}$, resulting in a large eigenvalue $\lambda_{z,j}^2$, but the training set X displays very little variance along vectors at very similar angles, $\hat{w}$ will incur high loss. We refer to this scenario, when an eigenvector demonstrates this spike in variance at test time, as **Spectral Inflation**. An illustration of Spectral Inflation and its consequences are depicted in Figure 1, and we present evidence of Spectral Inflation occurring in DNN representations in a real-world dataset in Figure 2. The analysis follows from the cyclic property of the trace operator, which allows us to isolate the noise term ϵ. This, in turn, enables a decomposition of the remaining expression in terms of the two eigenspectra of $Z^\top Z$ and $X^\top X$. A full derivation of this decomposition is available in Appendix B.

3.2. Spectral Adaptation Through Projection

We now focus on identifying the eigenvectors occupying the rows of $V_Z^\top$ that contribute significantly to the expected loss described in Theorem 3.1, and use them to construct a subset $S \subseteq \mathrm{Rows}(V_Z^\top)$. We then use S to construct a new regressor w_{proj}, by projecting $\hat{w}$ onto the subspace spanned by the eigenvectors in S^c, the complement of S:

$$w_{\mathrm{proj}} = \hat{w} - \sum_{e\in S} \langle \hat{w}, e\rangle e. \qquad (4)$$

This regressor is not influenced by the Spectral Inflation displayed along each eigenvector in S, as w_{proj} exists in a

[1] We assume after the design matrices X and Z are sampled that they are fixed, so the only random variables in our analysis are those in the label noise vector ϵ.

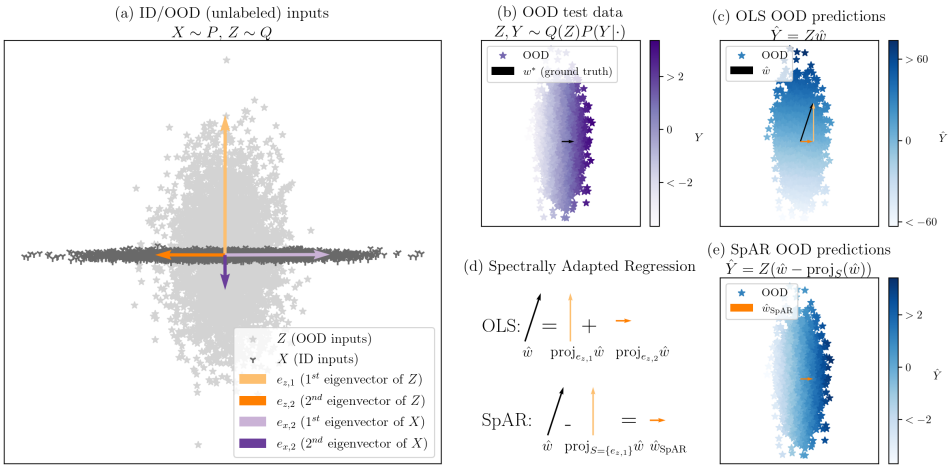

Figure 1. **Ordinary Least Squares Regression under Covariate Shift.** (a) Points in this illustration are 2D input samples in the training set X (in-distribution (ID)) and test set Z (out-of-distribution (OOD)). The training data demonstrates nearly zero vertical variance, while the test data varies significantly in this direction. (b) Samples in Z shaded according to their true, noiseless labels Zw^*. (c) Samples in Z shaded according to their OLS predictions $Z\hat{w}$. Crucially, to minimize training risk, *OLS learns to weigh the vertical component highly* causing erroneous predictions OOD. (d) Illustration of SpAR, which identifies a spectral subspace S where train/test variance differ the most, and projects it out. (e) Here the regressor created by SpAR ignores the direction with high variance and nearly recovers w^*. In (b, d, e), **Purple** indicates ground truth test labels, while **Blue** indicates model predictions on test data.

subspace orthogonal to the subspace spanned by the vectors in S. We can decompose the loss for this estimator w_{proj} into a sum over each eigenvector in $\text{Rows}(V_Z^\top)$, where the contribution of the eigenvector $e_{z,j}$ to the loss is determined by whether that eigenvector is included in the set S. The following theorem expresses the expected OOD loss of w_{proj}:

Theorem 3.2. *Taking on the same assumptions as Theorem 3.1, the regressor w_{proj} constructed using a set $S \subseteq \text{Rows}(V_Z^\top)$ as defined in Equation 4, has the following expected OOD squared error loss:*

$$\mathbb{E}[\|Y_Z - Zw_{\text{proj}}\|_2^2] = \underbrace{\sum_{j, e_{z,j} \in S} \langle w^*, e_{z,j}\rangle^2 \lambda_{z,j}^2}_{\text{Bias}_{z,j}} + \underbrace{\sum_{j, e_{z,j} \in S^c} \sigma^2 \sum_{i=1}^{D} \frac{\lambda_{z,j}^2}{\lambda_{x,i}^2} \langle e_{x,i}, e_{z,j}\rangle^2 \mathbb{1}[\lambda_{x,i} > 0]}_{\text{Var}_{z,j}}. \quad (5)$$

The proof for this theorem is similar to the proof of Theorem 3.1 in that it uses the cyclic property of the trace to isolate the noise term. We then use the fact that each $e_{z,j} \in S$ is an eigenvector of $Z^\top Z$ to further decompose the expression. A full derivation is included in Appendix C. This case-like decomposition of the loss motivates our definition of the two different loss terms a single eigenvector $e_{z,j}$ can contribute to the overall expected loss. For a given eigenvector $e_{z,j}$ with associated eigenvalue $\lambda_{z,j}^2$, we will incur its **variance** loss if $e_{z,j} \notin S$, and its **bias** loss if $e_{z,j} \in S$, where the variance loss $\text{Var}_{z,j}$ and bias loss $\text{Bias}_{z,j}$ are defined as:

$$\text{Bias}_{z,j} = \langle w^*, e_{z,j}\rangle^2 \lambda_{z,j}^2, \quad (6)$$

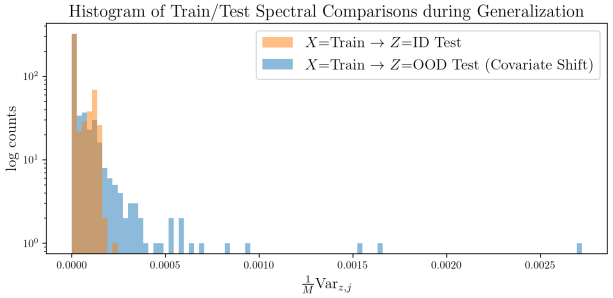

Figure 2. **Spectral Inflation.** We use the PovertyMap-WILDS dataset (Koh et al., 2021) to investigate how input spectra change when a regressor trained on real-world data generalizes to (perhaps shifted) test data. X and Z are composed of representations from a DNN. Z represents data either from an in-distribution or out-of-distribution test set. $\text{Var}_{z,j}$, as defined in Equation 6, measures the amount of *Spectral Inflation*—small amounts of training set variation becoming large at test time—occurring along a given test eigenvector. Because each test sample has a different number of examples M, we normalize for a fair comparison. We see that when Z is an out-of-distribution sample, much more spectral inflation occurs than when we generalize to an in-distribution sample.

$$\mathrm{Var}_{z,j} = \sigma^2 \sum_{i=1}^{D} \frac{\lambda_{z,j}^2}{\lambda_{x,i}^2} \langle e_{x,i}, e_{z,j} \rangle^2 \mathbb{1}[\lambda_{x,i} > 0].$$

$\mathrm{Var}_{z,j}$ is closely tied with the Spectral Inflation of an eigenvector, as $\mathrm{Var}_{z,j}$ will be large if $e_{z,j}$ demonstrates Spectral Inflation at test time. In this case if $e_{z,j} \notin S$, w_{proj} will have higher loss as a consequence of the label noise on the training examples distributed along this eigenvector. On the contrary, $\mathrm{Bias}_{z,j}$ is determined by the cosine similarity between the true labeling regressor w^* and the eigenvector $e_{z,j}$. High cosine similarity means that this eigenvector makes a large contribution to determining a sample's label. If $e_{z,j} \in S$ and $e_{z,j}$ has a large cosine similarity to w^*, w_{proj} will incur a high amount of loss as it is orthogonal to this important direction.

3.3. Projection Reduces Out-of-Distribution Loss

Thus far, we have presented a decomposition for the expected loss of an estimator that is equal to the pseudoinverse solution $\hat{w}$ projected into the ortho-complement of the span of the set $S \subseteq \mathrm{Rows}(V_Z^\top)$. In this subsection, we present a means for constructing the set S to minimize the expected loss by comparing $\mathrm{Var}_{z,j}$ and $\mathrm{Bias}_{z,j}$ for each test eigenvector $e_{z,j}$.

The ideal set $S^* \subseteq \mathrm{Rows}(V_Z^\top)$ would consist solely of the eigenvectors $e_{z,j}$ that have a greater variance loss than bias loss. Formally, this set would be constructed using the following expression:

$$S^* = \left\{ e_{z,j} : e_{z,j} \in \mathrm{Rows}(V_Z^\top), \mathrm{Var}_{z,j} \geq \mathrm{Bias}_{z,j} \right\}. \quad (7)$$

The following theorem demonstrates that using the set S^* would give us a regressor that achieves superior OOD performance than any other regressor produced using this projection procedure (including $\hat{w}$, which uses $S = \{\}$).

Theorem 3.3. *Under the same assumptions as Theorem 3.1, the regressor w^*_{proj} constructed as in Equation 4 using the set S^* (cf. Equation 7) can only improve on the OOD squared error loss of any other projected regressor w_{proj} constructed as in Equation 4 using a set $S \subseteq \mathrm{Rows}(V_Z^\top)$:*

$$\mathbb{E}[\|Y_Z - Z w_{\mathrm{proj}}\|_2^2] \geq \mathbb{E}[\|Y_Z - Z w^*_{\mathrm{proj}}\|_2^2]. \quad (8)$$

3.4. Eigenvector Selection Under Uncertainty

Theorem 3.3 shows that a regressor based on the set S^* has better OOD performance. Finding S^* would be easy if we knew both $\mathrm{Var}_{z,j}$ and $\mathrm{Bias}_{z,j}$ for each test eigenvector $e_{z,j}$. While we can calculate $\mathrm{Var}_{z,j}$ directly, $\mathrm{Bias}_{z,j}$ requires the true weight vector w^*, and so we can only *estimate* it using the pseudoinverse solution $\hat{w}$:

$$\widehat{\mathrm{Bias}}_{z,j} = \langle \hat{w}, e_{z,j} \rangle^2 \lambda_{z,j}^2 = (w^{*T} e_{z,j} + \epsilon^\top X^{\dagger\top} e_{z,j})^2 \lambda_{z,j}^2. \quad (9)$$

We fortunately have knowledge of some of the distributional properties of the dot product being squared: $\langle \hat{w}, e_{z,j} \rangle$. In particular, $w^{*\top} e_{z,j}$ is a fixed but unknown scalar and $\epsilon^\top X^{\dagger\top} e_{z,j}$ is the linear combination of several i.i.d. Gaussian variables with zero mean and variance σ^2.

$$\epsilon^\top X^{\dagger\top} e_{z,j} \lambda_{z,j} \sim \mathcal{N}(0, \mathrm{Var}_{z,j}) \quad (10)$$

$$\langle \hat{w}, e_{z,j} \rangle \lambda_{z,j} \sim \mathcal{N}(\sqrt{\mathrm{Bias}_{z,j}}, \mathrm{Var}_{z,j}).$$

The fact that $\widehat{\mathrm{Bias}}_{z,j}$ is a random variable makes it difficult to directly compare it with $\mathrm{Var}_{z,j}$. However, we can analyze the behavior of $\widehat{\mathrm{Bias}}_{z,j}$ when $\mathrm{Bias}_{z,j}$ is much larger than $\mathrm{Var}_{z,j}$, and vice versa, in order to devise a method for comparing these two quantities.

(Case 1): $\mathrm{Bias}_{z,j} \gg \mathrm{Var}_{z,j}$. In this case, $\mathrm{Bias}_{z,j} \approx \widehat{\mathrm{Bias}}_{z,j}$. This is because $w^{*\top} e_{z,j}$ will be much greater than $\epsilon^\top X^{\dagger\top} e_{z,j}$, which causes the former term to dominate in the RHS of Equation 9. Therefore $\widehat{\mathrm{Bias}}_{z,j} \gg \mathrm{Var}_{z,j}$.

(Case 2): $\mathrm{Var}_{z,j} \gg \mathrm{Bias}_{z,j}$. In this case, $\widehat{\mathrm{Bias}}_{z,j} \approx (\epsilon^\top X^{\dagger\top} e_{z,j})^2 \lambda_{z,j}^2$. This is because $w^{*\top} e_{z,j}$ will be much smaller than $\epsilon^\top X^{\dagger\top} e_{z,j}$, which causes the latter term to dominate in the RHS of Equation 9. Therefore, since Equation 10 indicates $(\epsilon^\top X^{\dagger\top} e_{z,j}) \lambda_{z,j}$ is a scalar Gaussian random variable, we know the distribution of its square:

$$\widehat{\mathrm{Bias}}_{z,j} \sim \mathrm{Var}_{z,j} \times \chi^2_{df=1}, \quad (11)$$

where $\chi^2_{df=1}$ is a chi-squared random variable with one degree of freedom. If $\mathrm{CDF}^{-1}_{\chi^2_{df=1}}$ is the inverse CDF of the chi-squared random variable, then we have:

$$\Pr(\widehat{\mathrm{Bias}}_{z,j} \leq \mathrm{CDF}^{-1}_{\chi^2_{df=1}}(\alpha) \times \mathrm{Var}_{z,j}) = \alpha. \quad (12)$$

By applying these two cases, we can construct our set S as follows:

$$S = \left\{ e_{z,j} : \widehat{\mathrm{Bias}}_{z,j} \leq \mathrm{CDF}^{-1}_{\chi^2_{df=1}}(\alpha) \times \mathrm{Var}_{z,j} \right\}. \quad (13)$$

The intuition behind this case-by-case analysis is formalized with the following proposition and lemma:

Proposition 3.4. *Making the same assumptions as Theorem 3.1, for a given choice of $\alpha \in [0, 1]$, the probability that test eigenvector $e_{z,j}$ is included in our set S as defined in 13:*

$$\Pr(e_{z,j} \in S) = 1 - Q_{\frac{1}{2}} \left(\sqrt{\frac{\mathrm{Bias}_{z,j}}{\mathrm{Var}_{z,j}}}, \sqrt{\mathrm{CDF}^{-1}_{\chi^2_{df=1}}(\alpha)} \right).$$

where $Q_{\frac{1}{2}}$ is the Marcum Q-function with $M = \frac{1}{2}$.

Lemma 3.5. *Using the same assumptions as Proposition 3.4:*

$$\Pr(e_{z,j} \in S) \xrightarrow{\frac{\mathrm{Bias}_{z,j}}{\mathrm{Var}_{z,j}} \to \infty} 0, \qquad \Pr(e_{z,j} \in S) \xrightarrow{\frac{\mathrm{Bias}_{z,j}}{\mathrm{Var}_{z,j}} \to 0} \alpha.$$

Lemma 3.5 tells us that if we would incur significantly higher OOD loss from including $e_{z,j}$ in S than excluding it, then $e_{z,j}$ **will not** be included in S. Similarly, if we would incur significantly higher OOD loss from excluding $e_{z,j}$ in our set S than including it, then $e_{z,j}$ **will** be included in S.

3.5. Spectral Adapted Regressor

Creating w_{proj} in this way yields SpAR, a regressor tailored for a specific covariate shift (see Algorithm 1). Finally, this procedure requires the the variance of the training label noise, σ^2. We use a maximum likelihood estimate of this parameter (Murphy, 2022) from the training data.

SpAR takes as input a set of embedded train and test examples. Creating these representations is slightly less computationally expensive than simply performing inference on these two datasets, and far less computationally expensive than an additional training epoch and test set evaluation. SpAR also requires SVD to be performed on X and Z, which is polynomial in the number of samples. Importantly, these additional computations only have to be performed a single time for each unique evaluation set. This is a stark contrast from other methods which require a computationally taxing regularizer to be computed with every batch (Ganin et al., 2016; Sun & Saenko, 2016; Yao et al., 2022). Empirically, we find that using SpAR is much faster than other methods we compare with (Appendix O).

It is important to note that $\lambda_{z,j}^2$ can be cancelled out from both $\widehat{\text{Bias}}_{z,j}$ and $\text{Var}_{z,j}$ in the comparison made in Equation 13. While the amount of variance seen in the target distribution makes a great impact on the value of the loss, SpAR does not consider $\lambda_{z,j}^2$ when selecting whether $e_{z,j}$ should be projected out. SpAR can be thought of selecting eigenvectors whose signal-to-noise ratio is low. $\frac{\widehat{\text{Bias}}_{z,j}}{\lambda_{z,j}^2}$ captures the "signal," or correlation between $e_{z,j}$ and the true weight vector, while $\frac{\text{Var}_{z,j}}{\lambda_{z,j}^2}$ captures the expected amount of noise associated with this eigenvector. Given that the Spectral Inflation associated with $e_{z,j}$ is the variance of the noise scaled by the variance in the test distribution $\lambda_{z,j}^2$, SpAR still targets eigenvectors that are likely to be a source of Spectral Inflation. Future work should investigate how to better make use of $\lambda_{z,j}^2$, such as how Lei et al. (2021) use these eigenvalues to produce a minimax optimal estimator.

4. Experiments

We apply SpAR to a suite of real-world and synthetic datasets to demonstrate its efficacy and explain how this method overcomes some shortcomings of OLS as well other methods meant to improve OOD robustness.

Here we use models that are optimized using gradient-based procedures. This contrasts with the main target of our analysis, the OLS solution (Equation 2), as $\hat{w}$ is not found using an iterative procedure. Despite these differences, our analysis remains relevant as the optimality conditions of minimizing the squared error loss ensure that gradient descent will converge to the OLS solution.

Algorithm 1 Spectral Adapted Regressor (SpAR)

Require: Training Data X, Y_X, Unlabeled Test Distribution Data Z, Rejection Confidence α

$\hat{w} \leftarrow X^\dagger Y_X$
$U_X, D_X, V_X^\top \leftarrow \text{SVD}(X)$
$U_Z, D_Z, V_Z^\top \leftarrow \text{SVD}(Z)$
$\hat{\sigma}^2 \leftarrow \text{MLE}(X, Y_X)$
$S \leftarrow \{\}$ ▷ Initialize the set S as empty
for $e_{z,j} \in \text{Rows}(V_Z^\top), \lambda_{z,j} \in \text{Diagonal}(D_Z)$ **do**
  $\text{Var}_{z,j} \leftarrow \hat{\sigma^2} \sum_{i=1}^{D} \frac{\lambda_{z,j}^2}{\lambda_{x,i}^2} \langle e_{x,i}, e_{z,j} \rangle^2 \mathbb{1}[\lambda_{x,i} > 0]$
  $\text{Bias}_{z,j} \leftarrow \langle \hat{w}, e_{z,j} \rangle^2 \lambda_{z,j}^2$
  if $(\text{CDF}_{\chi^2}^{-1}(\alpha) \times \text{Var}_{z,j}) \geq \text{Bias}_{z,j}$ **then**
    $S \leftarrow S \cup \{e_{z_j}\}$ ▷ Include this vector in S .
  end if
end for
$w_{\text{proj}} \leftarrow \hat{w} - \sum_{e \in S} \langle \hat{w}, e \rangle e$ ▷ Projection
return w_{proj}

4.1. Synthetic Data

We establish a proof of concept by considering a synthetic data setting where we can carefully control the distribution shift under study. Specifically, we apply our approach to two-dimensional Gaussian data following the data generative process described in Section 3.1. Specifically, for experiments 1,2, and 3, we sample our train and test data X and Z from origin-centered Gaussians with diagonal covariance matrices, where the variances of X and Z are $(5, 10^{-5})$ and $(1, 40)$ respectively. For Experiment 4, there is no covariate shift and so the diagonal covariance matrices of X and Z are both $(1, 40)$.

We refer to the first and second indices of these vectors as the "horizontal" and "vertical" components and plot the vectors accordingly. In the first three experiments, the test distribution has much more variance along the vertical component in comparison to the training distribution. We experiment with three different true labeling vectors: $w_1^* = (.01, .99999995)^T$; $w_2^* = (0.9999995, 0.01)^T$; $w_3^* = (\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}})^T$. The first two true labeling vectors represent functions that almost entirely depend on the vertical/horizontal component of the samples, respectively. w_3^* depends on both directions, though it depends slightly more on the vertical component. For Experiment 4, we re-use w_3^* as the true labelling vector as the focus of this exper-

Table 1. **Synthetic Data**. Mean of the squared error of estimated regressors versus true labeling vectors. Experiments 1, 2, and 3 use different true weight vectors, while Experiment 4 does not experience covariate shift (see Section 4.1).

Method	Exp. 1 (w_1^*)	Exp. 2 (w_2^*)	Exp. 3 (w_3^*)	Exp. 4 (w_3^*)
OLS	2.5e6 ± 3.8e6	2.5e6 ± 3.8e6	2.5e6 ± 3.8e6	**1.7e0** ± 1.2e0
PCR	**1.6e5** ± 3.1e3	**1.1e0**±0.8e0	**1.3e5** ± 2.5e3	8.0e2 ± 1.3e1
SpAR	**1.6e5** ± 3.1e3	**2.8e0**±4.5e0	**1.3e5** ± 2.5e3	**1.7e0** ± 1.2e0

iment is the lack of covariate shift. For each labeling vector, we randomly sample Z, X, and ϵ 10 times and calculate the squared error for three regression methods: **OLS/Pseudoinverse Solution** (OLS): the minimizer for the training loss, $\hat{w} = X^{\dagger}Y_X$; **Principal Component Regression** (Bair et al., 2006) (PCR): we calculate the OLS solution after projecting the data onto the first principal component of the training data; and **SpAR**: the regressor produced by SpAR.

We present the results of these experiments in Table 1. For the first three experiments, $\hat{w}$ has the same error regardless of the true labeling vector. Notably, SpAR outperforms OLS in each of the first 3 experiments. Our method is most effective when w_2^* is being used to label the examples. This is because it relies mostly on the horizontal component of the examples, which has a similar amount of variance at both train and test time. As a result, SpAR is able to project out the vertical component while retaining the bulk of the true labeling vector's information. An example showing why this projection method is useful when w_2^* is being used to label the examples is depicted in Figure 1. Here, $\hat{w}$ significantly overestimates the influence of the vertical component on the samples' labels. SpAR is able to detect that it will not be able to effectively use the vertical component due to the large increase in variance as we move from train to test, and so it projects that component out of $\hat{w}$. Consequently, SpAR produces a labeling function nearly identical to the true labeling function.

PCR projects the second principal component of the training data out of the OLS solution, regardless of the spectral properties of X or Z. This is a reasonable decision in some scenarios; PCR is able to achieve performance similar to SpAR on Experiments 1, 2, and 3. These experiments have the second training principal component experiencing *Spectral Inflation*, and so both methods achieve superior performance by projecting out the second principal component. In Experiment 4, however, no such *Spectral Inflation* occurs, and so SpAR and OLS achieve performance far superior to PCR by leaving the OLS regressor intact. This demonstrates one of SpAR's most important properties: it flexibly adapts to the covariate shift specified by the test data, rather than relying on the assumption that a certain adaptation will best perform OOD, as is the case with PCR.

4.2. PovertyMap - WILDS

We next examine the robustness of deep regression models under realistic distribution shifts in a high-dimensional setting. This experiment uses the PovertyMap-WILDS dataset (Koh et al., 2021), where the task is to regress local satellite images onto a continuous target label representing an asset wealth index for the region. PovertyMap provides an excellent test-bed for our method since, as seen in Figure 2, DNNs attempting to generalize OOD on this dataset suffer from Spectral Inflation.

For this dataset, we experiment with an unsupervised domain adaptation setting (Ben-David et al., 2006) where we used unlabeled target domain data distinct from the test set to perform adaptation with SpAR (Sagawa et al., 2022). Both this setting and the setting where adaptation is performed directly using the test data are realistic and relevant to machine learning (Shocher et al., 2018; Sun et al., 2020; Bau et al., 2019) and so we experiment with both conditions.

We compare with many methods for robust ML, including some "in-processing" methods (Caron et al., 2020) which use the unlabelled data to define an additional objective that is optimized during training. Results are presented in Table 2. Two methods that we will benchmark in other experiments as well are:

- **Standard Training (ERM)**: both the encoder and the regressor are trained in tandem to minimize the training objective using a gradient-based optimizer, in this case ADAM (Kingma & Ba, 2015).
- **C-Mixup** (Yao et al., 2022): a data augmentation technique that generalizes the Mixup algorithm (Zhang et al., 2018) to a regression setting. For this method, the encoder and regressor are optimized to minimize the error on both the original samples and the synthetic examples produced by C-Mixup.

Data-augmentation techniques such as C-Mixup can be used in tandem with other techniques for domain adaptation, such as SpAR, to achieve greater results than either of the techniques on their own. Our results substantiate this.

We use the hyperparameters reported by Yao et al. (2022) when training Resnet-18 based models (He et al., 2016) on both ERM and C-Mixup. When training these baselines, we follow Yao et al. (2022) and select the model checkpoint which performed best on a hold-out validation set as a form of early stopping. These choices help to create strong baselines. After training, we apply SpAR to create a new regressor using the representations produced by the ERM model (ERM + SpAR) or C-Mixup model (C-Mixup + SpAR). We explored a few settings of SpAR's hyperparameter α (see Appendix N for a discussion), and use a fixed value of $\alpha = 0.999$ in all experiments presented here.

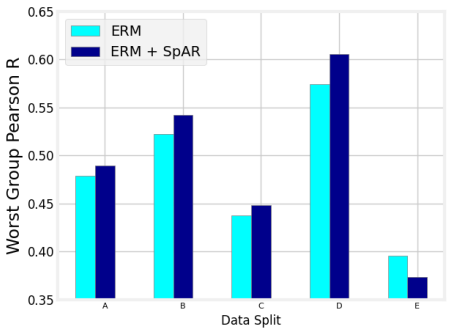

Figure 3. **PovertyMap**. Mean worst group Pearson r across 12 seeds for each of the 5 data splits of PovertyMap (higher is better).

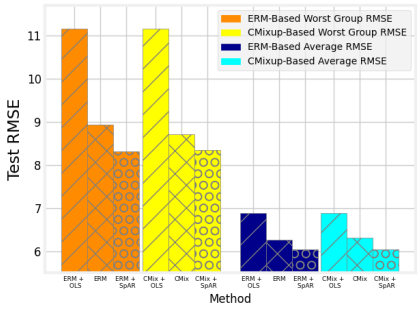

Figure 4. **SkillCraft.** OOD RMSE for several methods, each averaged across 10 seeds (lower is better).

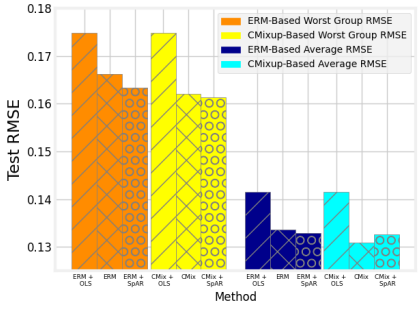

Figure 5. **CommunitiesAndCrime.** OOD RMSE for several methods, each averaged across 10 seeds (lower is better).

We find that even when using a sample distinct from the evaluation data, the use of SpAR on either ERM or C-Mixup yields the best performance. The worst group performance of C-Mixup + SpAR is **state of the art** on PovertyMap-WILDS for methods using unlabeled target domain data[2] (Sagawa et al., 2022). SpAR is also more computationally efficient than other robustness methods (see Appendix O).We also conduct experiments on applying SpAR using the evaluation data directly, the results of which are presented in Appendix K and L. SpAR achieves a similar performance increase to that presented in table 2, demonstrating SpAR's effectiveness when using different OOD samples.

PovertyMap uses the average of five different data splits to benchmark performance. To further investigate how SpAR performs in comparison to the strongest baseline, ERM, we repeated the experiments on each of these five different distribution shifts 12 times using 12 different seeds and calculated the average worst group Pearson r. We find that SpAR is able to improve performance in 4 out of the 5 distribution shifts (see Figure 3).

4.3. Tabular Datasets

We next experiment with two tabular datasets. Tabular data is common in real-world machine learning applications and benchmarks, particularly in the area of algorithmic fairness (Barocas et al., 2019). Therefore, it is important for robust machine learning methods to function well in this setting.

CommunitiesAndCrime, a popular dataset in fairness studies, provides a task where crime rates per capita must be predicted for different American communities, with some states held out of the training data and used to form an OOD test set (Redmond & Baveja, 2009; Yao et al., 2022). Skillcraft defines a task where one predicts the latency, in milliseconds, between professional video game players perceiving an action and making their own action (Blair et al., 2013). An OOD test set is created by only including players from certain skill-based leagues in the train or test set.

We train neural networks with one hidden layer in the style of Yao et al. (2022). We again use the hyperparameters reported by Yao et al. (2022) when training both ERM and C-Mixup. In addition to benchmarking SpAR, we similarly benchmark the performance of the Pseudoinverse solution by replacing the last layer weight with $\hat{w}$ (ERM/C-Mixup + OLS). Results from these tabular data experiments can be found in Figures 4 and 5. Exact numbers are presented in Table 7 in the Appendix.

Figures 4 and 5 show that SpAR always produces a model with competitive or superior Average and Worst Group RMSE, regardless of the base model that it is applied to. We also experiment with tuning the hyperparameters for both the ERM and C-Mixup models in Appendix M. With no additional tuning SpAR yields a model with the strongest worst-group performance.

4.4. Image Datasets

We now turn our attention to SpAR's efficacy on high-dimensional image datasets where a distribution shift is induced. Specifically, we experiment with the RCF-MNIST

[2] https://wilds.stanford.edu/leaderboard/#with-unlabeled-data-7

Table 2. **PovertyMap-WILDS with unlabeled data.** In-processing methods and SpAR use unlabeled data that are distinct from the test set, but come from the same distribution (Sagawa et al., 2022).

Robustness approach	Method	$r_{all}(\uparrow)$	$r_{wg}(\uparrow)$
—	ERM	**0.79** $\pm$ 0.04	0.50 $\pm$ 0.10
Data augmentation (pre-processing)	C-Mixup (Yao et al., 2022)	0.78 $\pm$ 0.05	0.49 $\pm$ 0.05
	Noisy Student (Xie et al., 2020)	0.76 $\pm$ 0.08	0.42 $\pm$ 0.11
Self-supervised pre-training (pre-processing)	SwAV (Caron et al., 2020)	0.78 $\pm$ 0.06	0.45 $\pm$ 0.05
Distribution alignment (in-processing)	DANN (Ganin et al., 2016)	0.69 $\pm$ 0.04	0.33 $\pm$ 0.10
	DeepCORAL (Sun & Saenko, 2016)	0.74 $\pm$ 0.05	0.36 $\pm$ 0.08
	AFN (Xu et al., 2019)	0.75 $\pm$ 0.08	0.39 $\pm$ 0.08
Subspace alignment (in-processing)	RSD (Chen et al., 2021)	0.78 $\pm$ 0.03	0.44 $\pm$ 0.09
	DARE-GRAM (Nejjar et al., 2023)	0.76 $\pm$ 0.06	0.44 $\pm$ 0.05
Spectral adaptation (post-processing)	**ERM + SpAR** (Ours)	**0.79** $\pm$ 0.04	0.51 $\pm$ 0.10
	C-Mixup + SpAR (Ours)	**0.79** $\pm$ 0.04	**0.52** $\pm$ 0.08

dataset from Yao et al. (2022), as well as the ChairAngles-Tails dataset from Gustafsson et al. (2023). RCF-MNIST tasks the model with predicting the angle of rotation for a series of images of clothing (Xiao et al., 2017). However, a spurious correlation between color and rotation angle that is inverted at test time causes regressors focussing on this spurious feature to perform poorly when evaluated. The ChairAngles-Tails dataset requires the model to predict the angle of rotation for a synthetic image of a chair. A distribution shift is induced by only including certain rotation angles in the training set.

We benchmark ERM and C-Mixup, as well as two additional baseline methods:

- **DANN** (Ganin et al., 2016): in addition to minimizing the training loss, the encoder is trained to maximize the loss of an adversary trained to predict whether the representation comes from the training set or test set.
- **Deep CORAL** (Sun & Saenko, 2016): the encoder minimizes both the training loss and the difference between the first and second moments of the train and test data matrices.

After performing a hyperparameter sweep (additional details in Appendix L), we average results across 10 random seeds. We apply SpAR to each of these baseline models using $\alpha = 0.999$ and no additional hyperparameter tuning. In the style of Yao et al. (2022), we select the model checkpoint which best performed on a hold-out validation set as a form of early stopping. Following Gustafsson et al. (2023) and Yao et al. (2022), we use Resnet-34 and Resnet-18 based models for ChairAngles-Tails and RCF-MNIST, respectively. Results are presented in Tables 3 and 4.

SpAR is regularly able to improve performance across a wide variety of architectures, tasks, and training methods. The best performing baseline method varies across these two datasets, with ERM performing best on RCF-MNIST and Deep CORAL performing best on ChairAngles-Tails. Despite these inconsistencies in baseline performance, SpAR consistently improves the performance of each method. This demonstrates SpAR's utility as a lightweight, efficient post processing method with a strong theoretical foundation that can be applied to a wide array of learned representations.

Table 3. **RCF-MNIST.** OOD RMSE averaged across 10 seeds.

RCF-MNIST		
Method	Baseline (↓)	Baseline + SpAR (↓)
ERM	**0.155** $\pm$ 0.006	**0.154** $\pm$ 0.006
C-Mixup	0.158 $\pm$ 0.011	0.156 $\pm$ 0.009
Deep CORAL	0.167 $\pm$ 0.012	0.165 $\pm$ 0.010
DANN	0.177 $\pm$ 0.019	0.170 $\pm$ 0.015

Table 4. **ChairAngles-Tails.** OOD RMSE averaged across 10 seeds.

ChairAngles-Tails		
Method	Baseline (↓)	Baseline + SpAR (↓)
ERM	6.788 $\pm$ 0.634	6.753 $\pm$ 0.648
C-Mixup	6.504 $\pm$ 0.324	6.449 $\pm$ 0.325
Deep CORAL	**5.978** $\pm$ 0.243	**5.839** $\pm$ 0.259
DANN	6.440 $\pm$ 0.602	6.337 $\pm$ 0.603

5. Related Work

Improving OOD performance is a critical and dynamic area of research. Our approach follows in the tradition of Transductive Learning (Gammerman et al., 2013) (adapting a model using unlabelled test data) and unsupervised Domain Adaptation (Ben-David et al., 2006; Farahani et al., 2021) (using distributional assumptions to model train/test differences, then adapting using unlabeled test inputs). Regularizing statistical moments between P and Q during training is a popular approach in unsupervised DA (Gretton et al., 2009) that has also been realized using deep neural networks (Ganin et al., 2016; Sun et al., 2017). Other methods exploit additional structure in P—such as auxiliary labels indicating the "domain" or "group" that each training example belongs to— to promote OOD generalization. Noteworthy approaches include Domain Generalization (Arjovsky et al., 2019; Gulrajani & Lopez-Paz, 2021) and Distributionally Robust Optimization (Hu et al., 2018; Sagawa et al., 2019; Levy et al., 2020).

Data augmentation is another promising avenue for improv-

ing OOD generalization (Hendrycks & Dietterich, 2019; Ovadia et al., 2019). The recently proposed C-Mixup method focuses on regression under covariate shift; it adapts the Mixup algorithm (Zhang et al., 2018) to regression by upweighting the convex combination of training examples whose target values are similar. This pre-processing approach complements our post-processing adaptation approach; in our experiments we find that applying SpAR to a C-Mixup model often yields the best results.

In this work we investigate covariate shift in a regression setting by analyzing how the distribution shift affects eigenspectra of the source/target data. Others have studied spectral properties in this setting. Tripuraneni et al. (2021) used a similar decomposition of the OLS test loss as a motivating example when analyzing the expected loss of random feature models under covariate shift. Lei et al. (2021) also proposed a similar expression for the OLS test loss as a step towards characterizing the minimax risk of linear models under covariate shift. In contrast to these works, we also propose a bias-variance decomposition for each eigenvector of the second moment matrix of the test representations, as well as methods for estimating both of these quantities. Pathak et al. (2022) propose a new similarity measure between P and Q that can be used to bound the performance of non-parameteric regression methods under covariate shift. Wu et al. (2022) analyze the sample efficiency of linear regression in terms of an eigendecomposition of the covariance matrices of P and Q. Our work differs from these in that we go beyond an OOD theoretical analysis to propose a practical post-processing algorithm, which we find to be effective on real-world datasets.

6. Conclusion

This paper investigated the generalization properties of regression models when facing covariate shift. We show that the OLS solution can fail dramatically OOD due to *Spectral Inflation*, where spectral subspaces with small variation during training see increased variation upon evaluation. Our adaptation method, SpAR, uses unlabeled test data to estimate the subspaces with spectral inflation and project them away. We apply our method to the last layer of deep neural regressors and find that it improves OOD performance on several synthetic and real-world datasets. Our limitations include assumed access to unlabeled test data, and that the distribution shift in question is covariate shift.

Impact Statement

Our research seeks to improve OOD generalization with the hopes of ensuring ML benefits are distributed more equitably across social strata. However, it is worthwhile to be self-reflexive about the methodology we use when working towards this goal. For example, for the purposes of comparing against existing methods from the literature, we use the Communities and Crime dataset, where average crime rates are predicted based on statistics of neighborhoods, which could include demographic information. This raises a potential fairness concern: even if we have an OOD-robust model, it may not be fair if it uses demographic information in its predictions. While this is not the focus of our paper, we note that the research community is in the process of reevaluating tabular datasets used for benchmarking (Ding et al., 2021; Bao et al., 2021).

Acknowledgements

We are grateful to Marc-Etienne Brunet, Samuel Monnier, Sruthi Sudhakar, and Tom Zollo for their valuable feedback on this manuscript. This work is supported by the funds provided by the National Science Foundation and by DoD OUSD (RE) under Cooperative Agreement PHY-2229929 (The NSF AI Institute for Artificial and Natural Intelligence). Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

References

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

Bair, E., Hastie, T., Paul, D., and Tibshirani, R. Prediction by supervised principal components. *Journal of the American Statistical Association*, 2006.

Balagopalan, A., Novikova, J., Mcdermott, M. B., Nestor, B., Naumann, T., and Ghassemi, M. Cross-language aphasia detection using optimal transport domain adaptation. In *Machine Learning for Health Workshop*, 2020.

Bao, M., Zhou, A., Zottola, S. A., Brubach, B., Desmarais, S., Horowitz, A. S., Lum, K., and Venkatasubramanian, S. It's compaslicated: The messy relationship between rai datasets and algorithmic fairness benchmarks. In *NeurIPS Datasets and Benchmarks Track (Round 1)*, 2021.

Barocas, S., Hardt, M., and Narayanan, A. *Fairness and Machine Learning: Limitations and Opportunities*. fairmlbook.org, 2019.

Bau, D., Strobelt, H., Peebles, W. S., Wulff, J., Zhou, B., Zhu, J., and Torralba, A. Semantic photo manipulation with a generative image prior. *ACM Transactions on Graphics (TOG)*, 2019.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. Analysis of representations for domain adaptation. *NeurIPS*, 2006.

Blair, M., Thompson, J., Henrey, A., and Chen, B. Skillcraft1 master table dataset. *UCI Machine Learning Repository*, 2013.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 2020.

Chen, X., Wang, S., Wang, J., and Long, M. Representation subspace distance for domain adaptation regression. In *ICML*, 2021.

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., et al. Underspecification presents challenges for credibility in modern machine learning. *Journal Machine Learning Research*, 2022.

DeGrave, A. J., Janizek, J. D., and Lee, S.-I. Ai for radiographic covid-19 detection selects shortcuts over signal. *Nature Machine Intelligence*, 2021.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

Ding, F., Hardt, M., Miller, J., and Schmidt, L. Retiring adult: New datasets for fair machine learning. *NeurIPS*, 2021.

Farahani, A., Voghoei, S., Rasheed, K., and Arabnia, H. R. A brief review of domain adaptation. *Advances in Data Science and Information Engineering: Proceedings from ICDATA 2020 and IKE 2020*, 2021.

Gammerman, A., Vovk, V., and Vapnik, V. Learning by transduction. *arXiv preprint arXiv:1301.7375*, 2013.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 2016.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2020.

Ginsberg, T., Liang, Z., and Krishnan, R. G. A learning based hypothesis test for harmful covariate shift. In *NeurIPS 2022 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2022.

Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., and Schölkopf, B. Covariate shift by kernel mean matching. *Dataset Shift in Machine Learning*, 2009.

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization. In *ICLR*, 2021.

Gustafsson, F. K., Danelljan, M., and Schön, T. B. How reliable is your regression model's uncertainty under real-world distribution shifts? *arXiv preprint arXiv:2302.03679*, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Hu, W., Niu, G., Sato, I., and Sugiyama, M. Does distributionally robust supervised learning give robust classifiers? In *ICML*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

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 *ICML*, 2021.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 2017.

Lei, Q., Hu, W., and Lee, J. Near-optimal linear regression under distribution shift. In *ICML*, 2021.

Levy, D., Carmon, Y., Duchi, J. C., and Sidford, A. Large-scale methods for distributionally robust optimization. *NeurIPS*, 2020.

Murphy, K. P. *Probabilistic machine learning: an introduction*. MIT press, 2022.

Nejjar, I., Wang, Q., and Fink, O. DARE-GRAM : Unsupervised domain adaptation regression by aligning inverse gram matrices. In *CVPR*, 2023.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *NeurIPS*, 2019.

Pathak, R., Ma, C., and Wainwright, M. A new similarity measure for covariate shift with applications to nonparametric regression. In *ICML*, 2022.

Redmond, M. and Baveja, A. Communities and crime data set. *UCI Machine Learning Repository*, 2009.

Ruan, Y., Dubois, Y., and Maddison, C. J. Optimal representations for covariate shift. In *ICLR*, 2022.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks. In *ICLR*, 2019.

Sagawa, S., Koh, P. W., Lee, T., Gao, I., Xie, S. M., Shen, K., Kumar, A., Hu, W., Yasunaga, M., Marklund, H., Beery, S., David, E., Stavness, I., Guo, W., Leskovec, J., Saenko, K., Hashimoto, T., Levine, S., Finn, C., and Liang, P. Extending the WILDS benchmark for unsupervised adaptation. In *ICLR*, 2022.

Shocher, A., Cohen, N., and Irani, M. "zero-shot" super-resolution using deep internal learning. In *CVPR*, 2018.

Sugiyama, M., Krauledat, M., and Müller, K.-R. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 2007.

Sun, B. and Saenko, K. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops*, 2016.

Sun, B., Feng, J., and Saenko, K. Correlation alignment for unsupervised domain adaptation. *Domain Adaptation in Computer Vision Applications*, 2017.

Sun, Y. and Baricz, Á. Inequalities for the generalized marcum q-function. *Applied Mathematics and Computation*, 2008.

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 *ICML*, 2020.

Tripuraneni, N., Adlam, B., and Pennington, J. Overparameterization improves robustness to covariate shift in high dimensions. *NeurIPS*, 2021.

Wu, J., Zou, D., Braverman, V., Gu, Q., and Kakade, S. The power and limitation of pretraining-finetuning for linear regression under covariate shift. *NeurIPS*, 2022.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xie, Q., Luong, M., Hovy, E. H., and Le, Q. V. Self-training with noisy student improves imagenet classification. In *CVPR*, 2020.

Xu, R., Li, G., Yang, J., and Lin, L. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *ICCV*, 2019.

Yao, H., Wang, Y., Zhang, L., Zou, J. Y., and Finn, C. C-mixup: Improving generalization in regression. *NeurIPS*, 2022.

Zhang, H., Cissé, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

A. OLS and the Pseudoinverse

Classical statistics (Murphy, 2022) tells us that when X is full rank, the $\hat{w}$ minimizing this expression—known as the OLS regressor—has the following form:

$$\hat{w}_{OLS} = (X^\top X)^{-1} X^\top Y_X \tag{14}$$

Of course, if X is not full rank, the product $X^\top X$ cannot be inverted. In this case, the minimum norm solution can be constructed using the singular value decomposition of X. Specifically, X can be decomposed as $X = U_X D_X V_X^\top$. We can then construct the pseudoinverse of X by using U_X, V_X, and the matrix $D^\dagger$ which is given by taking the transpose of D, and replacing the diagonal singular value elements with their reciprocal. In the case that the singular value is zero, the value of zero is used instead. The pseudoinverse is then constructed as $X^\dagger = V_X D_X^\dagger U_X^\top$. Using these components, the minimum norm solution in the case of a degenerate X matrix is given by the following expression:

$$\hat{w} = X^\dagger Y_X = V_X D_X^\dagger U_X^\top Y_X \tag{15}$$

B. Derivation of Loss of OLS Under Covariate Shift

We are interested in the following expression for the OOD risk of the OLS regressor:

$$\begin{aligned} \text{Risk}_{\text{OOD}}(\hat{w}) &= \mathbb{E}[\|Y_Z - Z\hat{w}\|_2^2] = \mathbb{E}[\|Zw^* - ZX^\dagger Y_X\|_2^2] \\ &= \mathbb{E}[\|Zw^* - ZX^\dagger(Xw^* + \epsilon)\|_2^2] \end{aligned} \tag{16}$$

If we assume that w^* exists within the span of the rows of X, then $X^\dagger X$ acts as an identity on w^*, giving us:

$$= \mathbb{E}[\|ZX^\dagger \epsilon\|_2^2] \tag{17}$$

The Euclidean norm is $\|x\|_2 = \sqrt{x^\top x}$, so we can rephrase this expression as a scalar dot product. Scalars can be seen as 1×1 matrices, and are therefore equal to their trace. Therefore we can express this dot product as a trace in order to later use the cyclic property of the trace operator:

$$= \mathbb{E}[\epsilon^\top X^{\dagger\top} Z^\top Z X^\dagger \epsilon] = \mathbb{E}[tr(\epsilon^\top X^{\dagger\top} Z^\top Z X^\dagger \epsilon)] \tag{18}$$

We can cycle the trace and apply the properties of the trace of the product of two $N \times N$ matrices:

$$= \mathbb{E}[tr(\epsilon\epsilon^\top X^{\dagger\top} Z^\top Z X^\dagger)] = \mathbb{E}[\sum_{i=1}^{N}\sum_{j=1}^{N} (\epsilon\epsilon^\top)_{i,j} (X^{\dagger\top} Z^\top Z X^\dagger)_{i,j}] \tag{19}$$

Since each entry of ϵ is independent from the other entries, and these entries follow the normal distribution $\mathcal{N}(0, \sigma^2)$, by applying the linearity of expectation we know that every term in this sum such that $i \neq j$ will be equal to zero, giving us:

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} \mathbb{E}[(\epsilon\epsilon^\top)_{i,j}](X^{\dagger\top} Z^\top Z X^\dagger)_{i,j} = \sum_{i=1}^{N} \sigma^2 (X^{\dagger\top} Z^\top Z X^\dagger)_{i,i} = \sigma^2 tr(X^{\dagger\top} Z^\top Z X^\dagger) \tag{20}$$

We will use the singular value decompositions of these two matrices to simplify the expression further after cycling the trace:

$$= \sigma^2 tr(Z^\top Z X^\dagger X^{\dagger\top}) = \sigma^2 tr(V_Z D_Z^\top U_Z^\top U_Z D_Z V_Z^\top V_X D_X^\dagger U_X^\top U_X D_X^{\dagger\top} V_X^\top) \tag{21}$$

$$= \sigma^2 tr(V_Z D_Z^2 V_Z^\top V_X D_X^{\dagger 2} V_X^\top) = \sigma^2 tr(D_X^{\dagger 2} V_X^\top V_Z D_Z^2 V_Z^\top V_X) \tag{22}$$

where $D_Z^2, D_X^{\dagger 2}$ are $D \times D$ diagonal matrices with diagonal values equal to the diagonal values of D_Z and $D_X^\dagger$ squared, respectively. The i^{th} diagonal entry of the matrix $V_X^\top V_Z D_Z^2 V_Z^\top V_X$ is:

$$\left[diag(V_X^\top V_Z D_Z^2 V_Z^\top V_X)\right]_i = \sum_{j=1}^{D} \lambda_{z,j}^2 \langle e_{x,i}, e_{z,j}\rangle^2 \tag{23}$$

Meaning that the entire expression will be equal to the value described:

$$\text{Risk}_{\text{OOD}}(\hat{w}) = \sigma^2 \sum_{i=1}^{D}\sum_{j=1}^{D} \frac{\lambda_{z,j}^2}{\lambda_{x,i}^2} \langle e_{x,i}, e_{z,j}\rangle^2 \mathbb{1}[\lambda_{x,i} > 0]. \tag{24}$$

C. Derivation of Bias-Variance Decomposition

SpAR produces a regressor of the following form:

$$w_{\text{proj}} = \hat{w} - \sum_{e \in S} \langle \hat{w}, e\rangle e \tag{25}$$

Where we are projecting out a set of eignevectors S from the pseudoinverse solution $\hat{w}$. We can substitute this into our expression for the OOD risk of a regressor to arrive at a bias-variance decomposition.

$$\text{Risk}_{\text{OOD}}(w_{\text{proj}}) = \mathbb{E}[\|Zw^* - Z(\hat{w} - \sum_{e_{z,j} \in S} \langle \hat{w}, e_{z,j}\rangle e_{z,j})\|_2^2] \tag{26}$$

$$= \mathbb{E}[\| - ZV_X D_X^\dagger U_X^\top \epsilon + Z \sum_{e_{z,j} \in S} (\epsilon^\top X^{\dagger\top} e_{z,j} + w^{*\top} e_{z,j}) e_{z,j}\|_2^2] \tag{27}$$

$$= \mathbb{E}[\| - ZV_X D_X^\dagger U_X^\top \epsilon + Z \sum_{e_{z,j} \in S} \langle w^*, e_{z,j}\rangle e_{z,j} + Z \sum_{e_{z,j} \in S} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j}\rangle e_{z,j}\|_2^2] \tag{28}$$

We can further simplify this expression by using the fact that the eigenvectors in $Rows(V_Z^\top)$ form an orthonormal basis, and so the sum of their outer products forms an identity matrix. Formally, $\sum_{j=1}^{D} e_{z,j} e_{z,j}^\top = I$. Using this on the leftmost term in the sum, we have:

$$= \mathbb{E}[\| - Z \sum_{j=1}^{D} e_{z,j} e_{z,j}^\top V_X D_X^\dagger U_X^\top \epsilon + Z \sum_{e_{z,j} \in S} \langle w^*, e_{z,j}\rangle e_{z,j} + Z \sum_{e_{z,j} \in S} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j}\rangle e_{z,j}\|_2^2] \tag{29}$$

$$= \mathbb{E}[\| - Z \sum_{j=1}^{D} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j}\rangle e_{z,j} + Z \sum_{e_{z,j} \in S} \langle w^*, e_{z,j}\rangle e_{z,j} + Z \sum_{e_{z,j} \in S} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j}\rangle e_{z,j}\|_2^2] \tag{30}$$

We can use the fact that $S \cup S^c$ form an orthogonal basis, where S^c is the complement set of eigenvectors. We are also assuming that we are only projecting out vectors from the Z right singular vector basis. This gives us:

$$\mathbb{E}[\| - Z \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j}\rangle e_{z,j} + Z \sum_{e_{z,j} \in S} \langle w^*, e_{z,j}\rangle e_{z,j}\|_2^2] = \mathbb{E}[\|V - B\|_2^2] \tag{31}$$

The Euclidean norm $\|x\|_2 = \sqrt{x^\top x}$, and so we can consider the sum of products $V^\top V - 2V^\top B + B^\top B$. If we take the expectation over the error term ϵ, which has mean 0, we are left with only $V^\top V + B^\top B$.

$V^\top V$ is the error term we are already familiar with (Theorem 3.1), restricted to the eigenvectors that weren't projected out:

$$V^\top V = (Z \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j})^\top Z \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j} \tag{32}$$

$$= (\sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j})^\top Z^\top Z \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j} \tag{33}$$

We note that each vector $e_{z,j} \in S^c$ is an eigenvector of $Z^\top Z$ with eigenvalue $\lambda_{z,j}^2$.

$$= \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j}^\top \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle \lambda_{z,j}^2 e_{z,j} \tag{34}$$

$$= \sum_{e'_{z,j} \in S^c} \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e'_{z,j} \rangle \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle \lambda_{z,j}^2 e'^\top_{z,j} e_{z,j} \tag{35}$$

Since S^c is a subset of an orthonormal basis, we know that $e'^\top_{z,j} e_{z,j} = 1$ iff $e'_{z,j} = e_{z,j}$. Otherwise, $e'^\top_{z,j} e_{z,j} = 0$.

$$= \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle^2 \lambda_{z,j}^2 = \sum_{e_{z,j} \in S^c} \epsilon^\top X^{\dagger\top} e_{z,j} e_{z,j}^T X^\dagger \epsilon \lambda_{z,j}^2 \tag{36}$$

In the expected loss, the expectation operator is applied to this expression, giving:

$$\mathbb{E}[V^\top V] = \mathbb{E}[\sum_{e_{z,j} \in S^c} \epsilon^\top X^{\dagger\top} e_{z,j} e_{z,j}^T X^\dagger \epsilon \lambda_{z,j}^2] \tag{37}$$

We can use the properties of the trace to isolate the label noise, as in Appendix B:

$$= \sum_{e_{z,j} \in S^c} \sigma^2 tr(e_{z,j}^\top X^\dagger X^{\dagger\top} e_{z,j}) \lambda_{z,j}^2 \tag{38}$$

We can analyze the inner product of the vector $X^{\dagger\top} e_{z,j} = U_X D_X^{\dagger\top} V_X^\top e_{z,j}$ with itself:

$$e_{z,j}^\top X^\dagger X^{\dagger\top} e_{z,j} = \sum_{i=1}^{d} \sum_{k=1}^{d} \frac{1}{\lambda_{x,i}} \langle e_{z,j}, e_{x,i} \rangle \frac{1}{\lambda_{x,k}} \langle e_{z,j}, e_{x,k} \rangle u_{x,i}^\top u_{x,k} \mathbb{1}[\lambda_{x,i} > 0] \mathbb{1}[\lambda_{x,k} > 0] \tag{39}$$

Where $u_{x,i}$ is the i^{th} column of U_X, i.e. the i^{th} left singular vector of X. These left singular vectors also create an orthonormal basis, and so $u_{x,i}^\top u_{x,k} = 1$ iff $u_{x,i} = u_{x,k}$. Otherwise, $u_{x,i}^\top u_{x,k} = 0$. This ultimately gives us:

$$\mathbb{E}[V^\top V] = \sigma^2 \sum_{i=1}^{D} \sum_{j, e_{z,j} \in S^c} \frac{\lambda_{z,j}^2}{\lambda_{x,i}^2} \langle e_{x,i}, e_{z,j} \rangle^2 \mathbb{1}[\lambda_{x,i} > 0] \tag{40}$$

We can use similar reasoning to show that bias term $B^\top B$ is a simple expression relying on the true weight vector:

$$\mathbb{E}[B^T B] = B^T B = \sum_{e_{z,j} \in S} \langle w^*, e_{z,j} \rangle e_{z,j}^\top Z^\top Z \sum_{e_{z,j} \in S} \langle w^*, e_{z,j} \rangle e_{z,j} \tag{41}$$

$$= \sum_{j, e_{z,j} \in S} \langle w^*, e_{z,j} \rangle^2 \lambda_{z,i}^2 \tag{42}$$

Therefore, we have the following expression for the expected loss:

$$\mathbb{E}[\|Zw^* - Z(\hat{w} - \sum_{e_{z,j}\in S}\langle \hat{w}, e_{z,j}\rangle e_{z,j})\|_2^2] = \mathbb{E}[V^\top V] + \mathbb{E}[B^\top B] \tag{43}$$

$$= \sigma^2 \sum_{i=1}^{D} \sum_{j, e_{z,j}\in S^c} \frac{\lambda_{z,j}^2}{\lambda_{x,i}^2} \langle e_{x,i}, e_{z,j}\rangle^2 \mathbb{1}[\lambda_{x,i} > 0] + \sum_{j, e_{z,j}\in S} \langle w^*, e_{z,j}\rangle^2 \lambda_{z,j}^2 \tag{44}$$

D. Proof of Theorem 3.3

In this section, we provide the proof of Theorem 3.3.

This theorem compares the OOD squared error loss of two regressors, w_{proj} and w^*_{proj}, which are constructed in the following way:

$$w_{\text{proj}} = \hat{w} - \sum_{e\in S}\langle \hat{w}, e\rangle e, \quad w^*_{\text{proj}} = \hat{w} - \sum_{e\in S^*}\langle \hat{w}, e\rangle e \tag{45}$$

We can invoke Theorem 3.2 to decompose the OOD squared error loss of the regressors:

$$\mathbb{E}[\|Y_Z - Zw_{\text{proj}}\|_2^2] = \sum_{e_{z,j}\in S^c} \text{Var}_{z,j} + \sum_{e_{z,j}\in S} \text{Bias}_{z,j} \tag{46}$$

$$\mathbb{E}[\|Y_Z - Zw^*_{\text{proj}}\|_2^2] = \sum_{e_{z,j}\in S^{*c}} \text{Var}_{z,j} + \sum_{e_{z,j}\in S^*} \text{Bias}_{z,j} \tag{47}$$

Since $\text{Rows}(V_Z^\top) = S \cup S^c = S^* \cup S^{*c}$, we can decompose the losses into four sums:

$$\mathbb{E}[\|Y_Z - Zw_{\text{proj}}\|_2^2] = \sum_{e_{z,j}\in S^c\cap S^{*c}} \text{Var}_{z,j} + \sum_{e_{z,j}\in S^c\cap S^*} \text{Var}_{z,j} + \sum_{e_{z,j}\in S\cap S^*} \text{Bias}_{z,j} + \sum_{e_{z,j}\in S\cap S^{*c}} \text{Bias}_{z,j}. \tag{48}$$

$$\mathbb{E}[\|Y_Z - Zw^*_{\text{proj}}\|_2^2] = \sum_{e_{z,j}\in S^c\cap S^{*c}} \text{Var}_{z,j} + \sum_{e_{z,j}\in S^c\cap S^*} \text{Bias}_{z,j} + \sum_{e_{z,j}\in S\cap S^*} \text{Bias}_{z,j} + \sum_{e_{z,j}\in S\cap S^{*c}} \text{Var}_{z,j}. \tag{49}$$

This gives us:

$$\mathbb{E}[\|Y_Z - Zw_{\text{proj}}\|_2^2] - \mathbb{E}[\|Y_Z - Zw^*_{\text{proj}}\|_2^2] = \sum_{e_{z,j}\in S^c\cap S^*} (\text{Var}_{z,j} - \text{Bias}_{z,j}) + \sum_{e_{z,j}\in S\cap S^{*c}} (\text{Bias}_{z,j} - \text{Var}_{z,j}). \tag{50}$$

By the definition of S^*, we know that $e_{z,j} \in S^*$ implies that $\text{Var}_{z,j} \geq \text{Bias}_{z,j}$. Therefore:

$$\sum_{e_{z,j}\in S^c\cap S^*} (\text{Var}_{z,j} - \text{Bias}_{z,j}) \geq 0. \tag{51}$$

Furthermore, for $e_{z,j} \notin S^*$ and therefore in S^{*c}, it must be the case that $\text{Var}_{z,j} < \text{Bias}_{z,j}$. Therefore:

$$\sum_{e_{z,j}\in S\cap S^{*c}} (\text{Bias}_{z,j} - \text{Var}_{z,j}) \geq 0. \tag{52}$$

This implies that the difference of OOD squared error losses is also greater or equal to zero, and therefore that w^*_{proj} achieves superior loss.

$$\mathbb{E}[\|Y_Z - Zw_{\text{proj}}\|_2^2] - \mathbb{E}[\|Y_Z - Zw^*_{\text{proj}}\|_2^2] = \sum_{e_{z,j}\in S^c\cap S^*} (\text{Var}_{z,j} - \text{Bias}_{z,j}) + \sum_{e_{z,j}\in S\cap S^{*c}} (\text{Bias}_{z,j} - \text{Var}_{z,j}) \geq 0 \tag{53}$$

$$\implies \mathbb{E}[\|Y_Z - Zw_{\text{proj}}\|_2^2] \geq \mathbb{E}[\|Y_Z - Zw^*_{\text{proj}}\|_2^2]. \tag{54}$$

E. Distribution of $\widehat{\text{Bias}}$

In Section 3.4 we make statements about the distribution of $\widehat{\text{Bias}}$. In this section, we further explain our reasoning for these claims.

$$\widehat{\text{Bias}}_{z,j} = \langle \hat{w}, e_{z,j} \rangle^2 \lambda_{z,j}^2 = (w^{*T} e_{z,j} + \epsilon^\top X^{\dagger\top} e_{z,j})^2 \lambda_{z,j}^2. \tag{55}$$

We know that ϵ is a Gaussian vector with zero mean and spherical covariance. Therefore, $\epsilon^\top X^{\dagger\top} e_{z,j} \lambda_{z,j}$ would also have zero mean. For its covariance, we need only to multiply this expression by itself to recognize the expression from previous derivations:

$$\mathbb{E}[e_{z,j}^\top X^\dagger \epsilon \epsilon^\top X^{\dagger\top} e_{z,j} \lambda_{z,j}^2] \tag{56}$$

This expression is seen in the derivation of Theorem 3.2, where we show it is equal to $\text{Var}_{z,j}$. Therefore, the variance of $\epsilon^\top X^{\dagger\top} e_{z,j} \lambda_{z,j}$ is $\text{Var}_{z,j}$. With this in mind, we can rewrite this expression as a scaling of a standard normal random variable:

$$\epsilon^\top X^{\dagger\top} e_{z,j} \lambda_{z,j} = \sqrt{\text{Var}_{z,j}}\beta, \quad \beta \sim \mathcal{N}(0,1) \tag{57}$$

We can also easily describe the distribution of $\langle \hat{w}, e_{z,j} \rangle \lambda_{z,j}$:

$$\langle \hat{w}, e_{z,j} \rangle \lambda_{z,j} = w^{*T} e_{z,j} \lambda_{z,j} + \epsilon^\top X^{\dagger\top} e_{z,j} \lambda_{z,j} \tag{58}$$

Which is a Gaussian random variable plus a constant, which shifts the mean of the Gaussian. This gives us the two distributions we list in Section 3.4:

$$\epsilon^\top X^{\dagger\top} e_{z,j} \lambda_{z,j} \sim \mathcal{N}(0, \text{Var}_{z,j}), \quad \langle \hat{w}, e_{z,j} \rangle \lambda_{z,j} \sim \mathcal{N}(\sqrt{\text{Bias}_{z,j}}, \text{Var}_{z,j}). \tag{59}$$

We would next like to explain the claims made in Case 2 of Section 3.4. Specifically, we make claims about the distribution of $\widehat{\text{Bias}}_{z,j}$ when $\widehat{\text{Bias}}_{z,j} \approx (\epsilon^\top X^{\dagger\top} e_{z,j})^2 \lambda_{z,j}^2$:

$$\widehat{\text{Bias}}_{z,j} \approx (\epsilon^\top X^{\dagger\top} e_{z,j})^2 \lambda_{z,j}^2 = (\sqrt{\text{Var}_{z,j}}\beta)^2 = \text{Var}_{z,j}\beta^2 \tag{60}$$

$$\beta \sim \mathcal{N}(0,1), \quad \beta^2 \sim \chi^2(df = 1) \tag{61}$$

We therefore know in this case that $\widehat{\text{Bias}}_{z,j}$ is the scaling of a chi-squared random variable. By properties of CDFs, we know that $\Pr(\text{Var}_{z,j}\beta^2 \leq \alpha) = \Pr(\beta^2 \leq \frac{\alpha}{\text{Var}_{z,j}})$, and therefore we know that the inverse CDF of $\text{Var}_{z,j}\beta^2$ will be $\text{CDF}^{-1}_{\chi^2_{df=1}}(\alpha) \times \text{Var}_{z,j}$.

F. Proof of Proposition 1

First, we will restructure $\widehat{\text{Bias}}_{z,j}$ as the scaling of a non-central chi-squared random variable. From Equation 59, we know the distribution of $\sqrt{\widehat{\text{Bias}}_{z,j}}$, which we can write in terms of a Gaussian random variable with non-zero mean:

$$\sqrt{\widehat{\text{Bias}}_{z,j}} = \langle \hat{w}, e_{z,j} \rangle \lambda_{z,j} \sim \mathcal{N}(\sqrt{\text{Bias}_{z,j}}, \text{Var}_{z,j}) \tag{62}$$

$$\implies \sqrt{\widehat{\text{Bias}}_{z,j}} = \sqrt{\text{Var}_{z,j}}\delta, \quad \delta \sim \mathcal{N}(\frac{\sqrt{\text{Bias}}}{\sqrt{\text{Var}_{z,j}}}, 1) \tag{63}$$

We therefore know that δ^2 is distributed according to a non-central chi-squared distribution:

$$\widehat{\text{Bias}}_{z,j} = (\sqrt{\text{Var}_{z,j}}\delta)^2 = \text{Var}_{z,j}\delta^2, \quad \delta^2 \sim \chi^2_\lambda(df = 1, \lambda = \frac{\text{Bias}_{z,j}}{\text{Var}_{z,j}}) \tag{64}$$

Furthermore, we know the CDF of this variable as $\Pr(\text{Var}_{z,j}\delta^2 \leq \alpha) = \Pr(\delta^2 \leq \frac{\alpha}{\text{Var}_{z,j}})$.

We include an eigenvector $e_{z,j}$ in our set S if $\widehat{\text{Bias}}_{z,j} \leq \text{CDF}^{-1}_{\chi^2_{df=1}}(\alpha) \times \text{Var}_{z,j}$. The probability of this event occurring is given by the CDF of $\widehat{\text{Bias}}_{z,j}$, which is the following:

$$\Pr(\widehat{\text{Bias}}_{z,j} \leq \text{CDF}^{-1}_{\chi^2_{df=1}}(\alpha) \times \text{Var}_{z,j}) = 1 - Q_{\frac{1}{2}}(\sqrt{\frac{\text{Bias}_{z,j}}{\text{Var}_{z,j}}}, \frac{\sqrt{\text{Var}_{z,j}}\sqrt{\text{CDF}^{-1}_{\chi^2_{df=1}}(\alpha)}}{\sqrt{\text{Var}_{z,j}}}) \tag{65}$$

$$= 1 - Q_{\frac{1}{2}}(\sqrt{\frac{\text{Bias}_{z,j}}{\text{Var}_{z,j}}}, \sqrt{\text{CDF}^{-1}_{\chi^2_{df=1}}(\alpha)}) \tag{66}$$

G. Proof of Lemma 3.5

Proposition 3.4 gives us an expression for the probability that a given eigenvector is included in the set S. Lemma 3.5 will use this proposition to demonstrate the tail behaviour of this expression. We will first note that since the expression in Proposition 3.4 is a CDF, it is continuous. Therefore, in order to find its limits at 0 and ∞, we need only be able to evaluate the expression at these values.

We will first show that:

$$\Pr(e_{z,j} \in S) \xrightarrow{\frac{\text{Bias}_{z,j}}{\text{Var}_{z,j}} \to \infty} 0 \tag{67}$$

This is a special value of the Marcum Q function (Sun & Baricz, 2008). Specifically, $Q_{\frac{1}{2}}(\infty, b) = 1$ for any b. Therefore:

$$\Pr(e_{z,j} \in S) = 1 - Q_{\frac{1}{2}}(\infty, \sqrt{\text{CDF}^{-1}_{\chi^2_{df=1}}(\alpha)}) = 1 - 1 = 0 \tag{68}$$

We will next show that:

$$\Pr(e_{z,j} \in S) \xrightarrow{\frac{\text{Bias}_{z,j}}{\text{Var}_{z,j}} \to 0} \alpha \tag{69}$$

This is another special value of the Marcum Q function (Sun & Baricz, 2008). Specifically:

$$Q_{\frac{1}{2}}(0, b) = \frac{\Gamma(\frac{1}{2}, \frac{b^2}{2})}{\Gamma(\frac{1}{2})} \tag{70}$$

For any b. Here, Γ with one argument is the gamma function and Γ with two arguments is the upper incomplete gamma function. By properties of gamma functions, we know that if γ is the lower incomplete gamma function, then $\Gamma(\frac{1}{2}, \frac{b^2}{2}) + \gamma(\frac{1}{2}, \frac{b^2}{2}) = \Gamma(\frac{1}{2})$. Using this property, and by letting $b = \sqrt{\text{CDF}^{-1}_{\chi^2_{df=1}}(\alpha)}$, we have the following:

$$\Pr(e_{z,j} \in S) = 1 - Q_{\frac{1}{2}}(0, b) = \frac{\Gamma(\frac{1}{2}, \frac{b^2}{2}) + \gamma(\frac{1}{2}, \frac{b^2}{2})}{\Gamma(\frac{1}{2})} - \frac{\Gamma(\frac{1}{2}, \frac{b^2}{2})}{\Gamma(\frac{1}{2})} \tag{71}$$

$$= \frac{\gamma(\frac{1}{2}, \frac{b^2}{2})}{\Gamma(\frac{1}{2})} = \text{CDF}_{\chi^2_{df=1}}(\text{CDF}^{-1}_{\chi^2_{df=1}}(\alpha)) = \alpha \tag{72}$$

Where we have used the observation that the leftmost expression in Equation 72 is the CDF for a chi-squared distribution with one degree of freedom.

H. High Dimensional Synthetic Data

To supplement the two-dimensional experiment presented in Section 4.1, we present an additional synthetic experiment that is more complex and occurs in a higher-dimensional space. It uses the following data generative process, which occurs in 100-dimensional space:

1. Generate a random target vector w^* from $\mathcal{N}(0, \mathcal{I})$, the multivariate Normal distribution with zero mean and identity covariance, and normalize it to be a unit vector.
2. Generate two random means from $\mathcal{N}(0, \mathcal{I})$, which will be the means of our train and test distributions.
3. Generate another random vector from $\mathcal{N}(0, \mathcal{I})$ and take its absolute value. These will be the entries of our diagonal test covariance matrix.
4. Generate a final random vector from $\mathcal{N}(0, \mathcal{I})$ and take its absolute value for the ID covariance matrix. For block j of 10 entries in the vector, scale the entries by $10^{(7-j)}$.
   - For instance, entries 90 through 99 will be scaled by $10^{(7-9)}$.
   - As we descend through the entries, the scale of the variance along the associated eigenvectors seen at training time shrinks exponentially. Meanwhile, the test distribution has no such scaling.
5. Generate ID and OOD Gaussian data X and Z, respectively, according to the respective means and covariance matrices.
6. Generate labels using the true target vector, with noise added to the training labels: $Y_X = Xw^* + \epsilon, \epsilon \sim \mathcal{N}(0, \mathcal{I}), Y_Z = Zw^*$.

We repeat this random procedure 100 times and benchmark the same methods presented in Table 1. We find that SpAR is able to vastly outperform PCR and make strong gains on OLS. The results are presented in Table 5:

Table 5. High-dimensional synthetic experiment results.

	ERM/OLS	PCR	U-PCR	SpAR
OOD Squared Error	8.7e3±1.4e4	1.5e5±3.2e5	6.4e3±4.2e3	**5.9e3±7.0e3**

We note that since the mean differs between the train and test distributions, traditional PCR will suffer from the centering operation performed using the training data. We include PCR as a baseline in spite of this as PCR is a commonly used method.

In addition, we compare with a baseline that projects the OLS regressor onto the first right singular vector of X. This is equivalent to PCR without the centering operation (U-PCR). We find that SpAR has a slight advantage, despite U-PCR being a heavily biased method meant to overcome the shortcomings of PCR.

I. Additional Training Details

For our experiments in Section 4, we adapt the code provided by Yao et al. (2022) in this Github repo: `https://github.com/huaxiuyao/C-Mixup`. While training, we perform early stopping on a validation set evaluation metric. For PovertyMap, this procedure is seen in the original work of Koh et al. (2021). We also use the hyperparameters provided in the appendix of Yao et al. (2022)'s work, including the learning rates and bandwidth parameters for C-Mixup provided in Table 6.

Table 6. Hyperparameters used for training models responsible for the results in Section 4.

Hyperparameter	CommunitiesAndCrime	SkillCraft	PovertyMap
Learning Rate	1e-3	1e-2	1e-3
Bandwidth	1.0	5e-4	0.5

Table 7. **Tabular data.** OOD RMSE averaged across 10 seeds for models using the hyperparameters described in Table 6.

SkillCraft			**CommunitiesAndCrime**		
Method	Average RMSE (↓)	Worst Group RMSE (↓)	Method	Average RMSE (↓)	Worst Group RMSE (↓)
ERM	6.273 ± 0.384	8.933 ± 1.338	ERM	0.134 ± 0.006	0.166 ± 0.014
ERM + OLS	6.884 ± 0.860	11.156 ± 3.892	ERM + OLS	0.142 ± 0.004	0.175 ± 0.012
ERM + SpAR (Ours)	**6.049** ± 0.379	**8.317** ± 1.327	ERM + SpAR (Ours)	**0.133** ± 0.002	**0.163** ± 0.009
C-Mixup	6.319 ± 0.450	8.713 ± 1.106	C-Mixup	**0.131** ± 0.005	0.162 ± 0.016
C-Mixup + OLS	7.070 ± 0.898	11.747 ± 3.450	C-Mixup + OLS	0.140 ± 0.003	0.175 ± 0.010
C-Mixup + SpAR (Ours)	**6.038** ± 0.705	**8.343** ± 1.563	C-Mixup + SpAR (Ours)	0.133 ± 0.002	**0.161** ± 0.004

We additionally make the modification to train models without a bias term in the final linear layer. This is due to the fact that SpAR assumes a regressor that does not use a bias.

Models are trained using Tesla T4 GPUs from NVIDIA. Tabular and synthetic experiments take less than 10 minutes to run for a single seed and hyperparameter setting. PovertyMap experiments take roughly 3 hours to run when training ERM and roughly 15 hours to run when training C-Mixup.

J. Tabular Data Results with Base Hyperparameters

In this section, we provide the table of results that Figures 5 and 4 are based upon. This is the performance of the models using the hyperparameters described in Table 6. The results are included in Table 7.

K. Transductive Learning for Povertymap

In this section, we investigate SpAR's effectiveness on the PovertyMap dataset when using the evaluation data directly. We can observe from Table 8 that applying SpAR can significantly improve worst-group performance while maintaining competitive average performance. This is similar to the results presented in Table 2, suggesting that SpAR can achieve strong performance with different sets of samples from the target distribution.

L. Hyperparameter Search

For hyperparameter tuning, we perform random search over the learning rate and the bandwidth used in C-Mixup. Specifically, we search over learning rates using the following formula for the learning rate lr and bandwidth bw:

$$lr = base_{lr} * 10^u, u \sim Unif(-1, 1) \tag{73}$$

$$bw = base_{bw} * 10^u, u \sim Unif(-1, 1) \tag{74}$$

where $base_{lr}$ and $base_{bw}$ are the values described in Table 6 for each dataset. We test out 10 randomly selected hyperparameter settings for both ERM and C-Mixup, and select the settings that yield the best validation performance. Those

Table 8. **PovertyMap-WILDS.** Average OOD all-group and worst-group Spearman r across 5 splits.

Method	$r_{all}(\uparrow)$	$r_{wg}(\uparrow)$
ERM	0.793 ± 0.040	0.497 ± 0.099
ERM + SpAR (Ours)	**0.794** ± 0.046	**0.512** ± 0.092
C-Mixup	0.784 ± 0.045	0.489 ± 0.045
C-Mixup + SpAR (Ours)	**0.794** ± 0.043	**0.515** ± 0.091

hyperparameter settings selected for C-Mixup are presented in Table 9 and hyperparameter settings selected for ERM are presented in Table 10.

Table 9. Tuned hyperparameters used for training C-Mixup models. 4.

Hyperparameter	CommunitiesAndCrime	SkillCraft	PovertyMap
Learning Rate	0.003630376073213171	0.023276939100527687	0.003630376073213171
Bandwidth	0.35090148857968506	0.0013316008334250096	0.17545074428984253

Table 10. Tuned hyperparameters used for training ERM models.

Hyperparameter	CommunitiesAndCrime	SkillCraft	PovertyMap
Learning Rate	0.008246671732726021	0.023276939100527687	0.003630376073213171

For RCF-MNIST and ChairAngles-Tails, we use a similar procedure, only instead using a base learning rate of $7e-5$ for RCF-MNIST and 0.001 for ChairAngles-Tails. For the bandwidth, we use a base bandwidth of 0.2 for RCF-MNIST and $5e-4$ for ChairAngles-Tails. Additionally, Deep CORAL and DANN require a penalty weight pw which we generate using the same procedure and a base pw of 1.0. These hyperparameters are presented in Tables 11 and 12.

$$pw = base_{pw} * 10^u, u \sim Unif(-1, 1) \tag{75}$$

Table 11. Tuned hyperparameters used for training models on RCF-MNIST.

Method/Hyperparameter	Learning Rate	Bandwidth	Penalty Weight
ERM	3.851230830192189e-05	-	-
C-Mixup	3.636910217027964e-05	0.20091864782463165	-
Deep CORAL	5.2547676552479794e-05	-	3.721492665736836
DANN	7.100098449013041e-05	-	0.10654593849857387

Table 12. Tuned hyperparameters used for training models on ChairAngles-Tails.

Method/Hyperparameter	Learning Rate	Bandwidth	Penalty Weight
ERM	0.0003572140318996373	-	-
C-Mixup	0.0007506810936068544	0.0018607463328684177	-
Deep CORAL	0.008246671732726021	-	5.647617424572879
DANN	0.0010142997784304342	-	0.10654593849857387

M. Tuned Baselines

Using the hyperparameters presented in Tables 9 and 10 which were selected hyperparameter search process described in Section L, we benchmark the performance of ERM and C-Mixup models across 10 seeds for the tabular datasets and the 5 data folds for PovertyMap. We report results for PovertyMap and the tabular datasets in Tables 14 and 13, respectively.

We find that SpAR can achieve superior worst group performance than any other method presented in either Tables 14 or 13, or in Section 4. For C-Mixup on CommunitiesAndCrime, we see that tuning hyperparameters on the validation set yields poorer performance (Table 13) than using the hyperparameters presented in Yao et al. (2022)'s work (Table 7). However, we can see that a SpAR model is able to achieve the best worst-group RMSE of any model on this dataset, 0.161.

N. Sensitivity of Alpha Hyperparameter

Throughout this work, we use a single setting of α for each of our experiments. Our specific setting of α=0.999 was selected using a minimal amount of tuning on a single seed of a single experiment. This value was then used on every seed of every dataset, regardless of potential improvements. To achieve a more complete understanding of SpAR's sensitivity to α, we conduct an experiment measuring OOD performance as a function of α when SpAR is applied to an ERM base model on the

Table 13. **Tabular data.** OOD RMSE averaged across 10 seeds for models using tuned hyperparameters.

SkillCraft			**CommunitiesAndCrime**		
Method	Average RMSE (↓)	Worst Group RMSE (↓)	Method	Average RMSE (↓)	Worst Group RMSE (↓)
ERM	**5.917** ± 0.620	8.308 ± 1.915	ERM	**0.133** ± 0.004	**0.161** ± 0.010
ERM + OLS	6.548 ± 0.915	10.219 ± 3.123	ERM + OLS	0.149 ± 0.018	0.184 ± 0.032
ERM + SpAR (Ours)	6.083 ± 0.681	**8.193** ± 1.212	ERM + SpAR (Ours)	0.134 ± 0.007	0.164 ± 0.013
C-Mixup	**5.816** ± 0.558	8.371 ± 1.611	C-Mixup	0.133 ± 0.003	0.171 ± 0.012
C-Mixup + OLS	6.535 ± 0.822	10.297 ± 2.362	C-Mixup + OLS	0.144 ± 0.011	0.177 ± 0.019
C-Mixup + SpAR (Ours)	5.833 ± 0.580	**7.922** ± 1.043	C-Mixup + SpAR (Ours)	**0.132** ± 0.004	**0.164** ± 0.008

Table 14. **PovertyMap-WILDS.** Average OOD all-group and worst-group Spearman r across 5 splits for models using tuned hyperparameters.

Method	$r_{all}(\uparrow)$	$r_{wg}(\uparrow)$
ERM	0.798 ± 0.052	0.518 ± 0.076
ERM + SpAR (Ours)	**0.799** ± 0.045	**0.522** ± 0.080
C-Mixup	**0.806** ± 0.031	0.523 ± 0.083
C-Mixup + SpAR (Ours)	0.803 ± 0.038	**0.528** ± 0.087

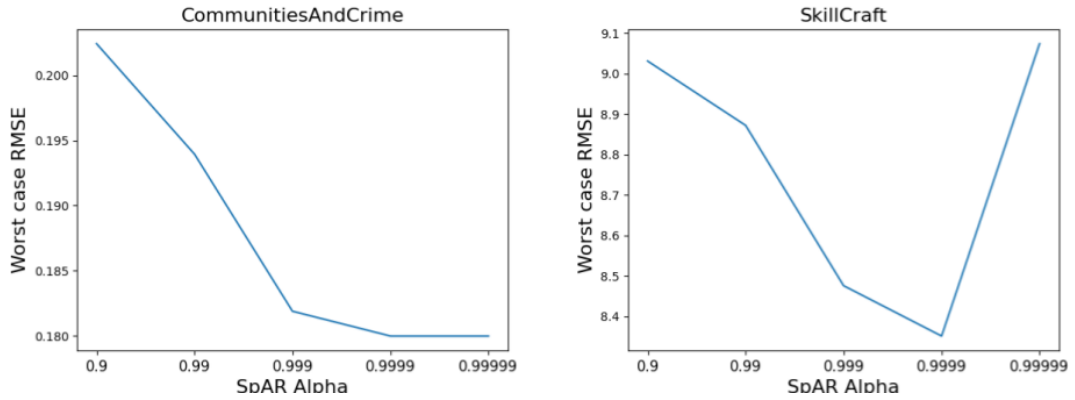

Figure 6. **Hyperparameter sensitivity** SpAR performance as a function of α on tabular datasets.

SkillCraft and CommunitiesAndCrime datasets. See Figure 6 for results. We see that on the CommunitiesAndCrime dataset, a higher α than 0.999 could have resulted in superior worst case performance. Meanwhile, on SkillCraft, we clearly see that setting α too close to 1 can result in very poor worst group performance. Expression 13 in the paper indicates that as α tends towards zero, the regressor produced by SpAR will more closely resemble the solution produced by OLS. Specifically, fewer eigenvectors will be projected out from the OLS solution. Conversely, as α tends towards one, the regressor produced by SpAR will tend towards the zero vector. This can be seen as a tradeoff between the cases where no Spectral Inflation is expected and where Spectral Inflation is expected to occur along every right singular vector.

In general, selecting α using validation set performance can have mixed results, as SpAR is intended to produce a regressor for a specific evaluation set (namely, the OOD test set, not the ID validation set). Future work could investigate the interesting question of how α could be selected based on the amount of spectral inflation presented in the train/evaluation data.

O. Computational Cost

The computational cost of SpAR comes from collecting the representations (running forward passes for every train and test example) and performing SVD, with the former step dominating the cost. Notably, it is much less cumbersome than other adaptation techniques. Computing the SVD of the matrix can be done in polynomial time, and we find in practice that performing this one-time post-hoc adaptation is quite efficient relative to other methods that must compute a regularizer or augment data on each training iteration (see Table 15).

Table 15. **Measured train time** on PovertyMap. Each model is trained on a NVIDIA Tesla T4 GPU. In-processing methods and SpAR use a large pool of unlabeled data that are distinct from the test set, but come from the same distribution (Sagawa et al., 2022).

Method	Average RMSE
ERM	3h22m $\pm$ 0h22m
C-Mixup	14h58 $\pm$ 1h01m
DARE-GRAM	5h58m $\pm$ 0h26m
ERM + SpAR (Ours)	4h11m $\pm$ 0h33m
SpAR only	0h40m $\pm$ 0h18m

P. Limitations

SpAR is designed for covariate shift, and its ability to handle other types of distribution shift (such as concept shift) is not known analytically. To be more specific, we assume that the targets have a the same linear relationship (via the ground truth weight w^*) with inputs X and Z, and that X and Z are covariate-shifted. A subtle issue here is that when X and Z are internal representations of some neural net, we require that the difference P and Q is captured in terms of a covariate shift *in the representation space*, which may or may not correspond to a covariate shift in the original input space (which could be some high-dimensional vector, e.g. pixels).

Empirically, however, we successfully apply SpAR to several real-world datasets without assurance that they exhibit only covariate shift, and find promising results. The spectral inflation property that we observe in real data (Figure 2) may be relevant to other distribution shifts as well, although this remains to be seen in future studies. Identifying covariate shift within a datasets is an active area of work (Ginsberg et al., 2022) that complements our efforts in this paper.