Out of the Ordinary: Spectrally Adapting Regression for Covariate Shift

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

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 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., 2018)—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., 2020; 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, 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. We provide further discussion of work related to this problem in Section 4.

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 for why regressors may fail under covariate shift, a specific form of distribution shift where the relationship between the sample and the label is the same both in and out of distribution (Sugiyama et al., 2007; Gretton et al., 2009; Ruan et al., 2021). We take on a transductive learning setting for domain adaptation, and by assuming access to unlabelled evaluation data (Sun et al., 2020; Bau et al., 2020; Shocher et al., 2018) we are able to improve the performance of regression models in a synthetic experiment and three real-world datasets.

2. 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)**.

[1]University of Toronto [2]Vector Institute [3]Google Research [4]Columbia University. Correspondence to: Benjamin Eyre <benjamin.eyre@mail.utoronto.ca>.

Published at the ICML 2023 Workshop on Spurious Correlations, Invariance, and Stability. Honolulu, Hawaii, USA.

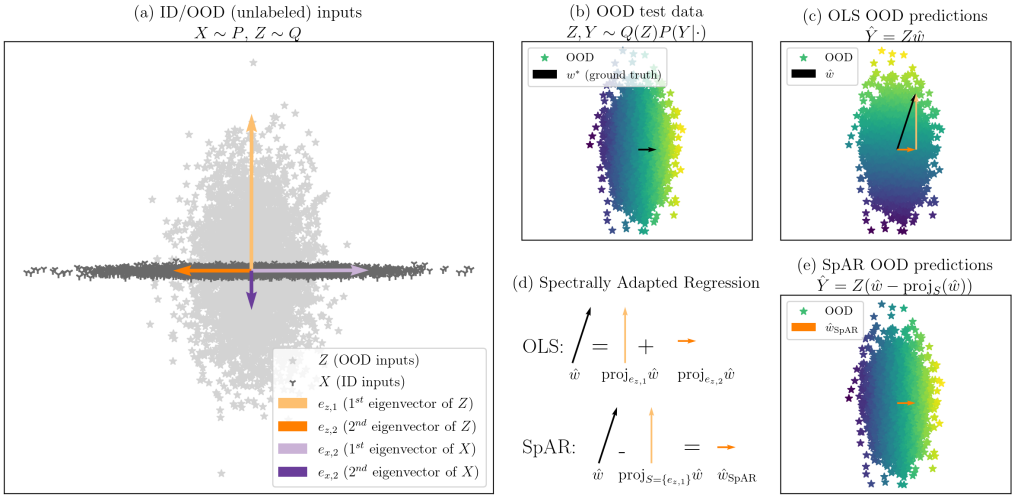

Figure 1. **Ordinary Least Squares Regression under Covariate Shift.** (a) Points are 2D input samples in the training set X and test set Z. The in-distribution (ID) training data demonstrates nearly zero vertical variance, while the out-of-distribution (OOD) test data varies significantly in this direction. (b) Samples in Z colored according to their true, noiseless labels Zw^*. (c) Samples in Z colored 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. (e) SpAR identifies a spectral subspace S where train/test variance differ the most, and projects it out. Thus, the regressor created by SpAR ignores the direction with high variance and nearly recovers w^*.

2.1. 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 labelling vector in $\mathbb{R}^D$, and a noise term ϵ. The targets associated with the test data Z use the same true labelling 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, $X^\dagger = V_X D_X^\dagger U_X^\top$:

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

We refer to $\hat{w}$ as the "OLS regressor" or "pseudoinverse solution". We will analyze 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:

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

In addition to using the singular value decomposition $X = U_X S_X V_X^\top$, we can also use the singular value decomposition 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 matrices $X^\top X$ and $Z^\top Z$. We represent the set containing the rows of a matrix with the operator $\text{Rows}()$. The OOD risk of $\hat{w}$ is presented in the following theorem in terms of interaction between the eigenspectra of X and Z:

Theorem 2.1. *Assuming the data generative procedure defined in Equations 1, and that* $w^* \in \text{Span}(\text{Rows}(X))$ *and* $\text{Rows}(Z) \subset \text{Span}(\text{Rows}(X))$*, the OOD squared error loss of the estimator* $\hat{w} = X^\dagger Y$ *is equal to:*

$$\mathbb{E}[\|Y_Z - Z\hat{w}\|_2^2] = \sigma^2 \sum_{i=1}^{D}\sum_{j=1}^{D} \frac{\lambda_{z,j}^2}{\lambda_{x,i}^2} \langle e_{x,i}, e_{z,j}\rangle^2 \mathbb{1}[\lambda_{x,i} > 0]. \qquad (4)$$

This theorem indicates that if the samples in Z present a large amount of variance along the vector $e_{z,j}$, resulting in

a large eigenvalue $\lambda^2_{z,j}$, 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 Section J. 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 D.

2.2. 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 Equation 4, and use them to construct a subset $S \subseteq \text{Rows}(V_Z^\top)$. 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:

$$w_{\text{proj}} = \hat{w} - \sum_{e \in S} \langle \hat{w}, e \rangle e. \tag{5}$$

This regressor is not influenced by the Spectral Inflation displayed along each eigenvector in S, as w_{proj} exists in a subspace orthogonal to the subspace spanned by the vectors in S. Specifically, we can decompose the loss for this estimator w_{proj} into a sum over each of the eigenvectors 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 2.2. *Taking on the same assumptions as Theorem 2.1, the regressor w_{proj} constructed using a set $S \subseteq \text{Rows}(V_Z^\top)$ as defined in Equation 5, has the following expected OOD squared error loss:*

$$\mathbb{E}[\|Y_Z - Z w_{\text{proj}}\|_2^2] = \underbrace{\sum_{j, e_{z,j} \in S} \langle w^*, e_{z,j} \rangle^2 \lambda^2_{z,i}}_{\text{Bias}_{z,j}} + \underbrace{\sum_{j, e_{z,j} \in S^c} \sigma^2 \sum_{i=1}^{D} \frac{\lambda^2_{z,j}}{\lambda^2_{x,i}} \langle e_{x,i}, e_{z,j} \rangle^2 \mathbb{1}[\lambda_{x,i} > 0]}_{\text{Var}_{z,j}} \tag{6}$$

Similar to the proof of Theorem 2.1, this theorem's proof 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 (Appendix Section E). This 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 $\lambda^2_{z,j}$, 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:

$$\text{Bias}_{z,j} = \langle w^*, e_{z,j} \rangle^2 \lambda^2_{z,j}, \tag{7}$$
$$\text{Var}_{z,j} = \sigma^2 \sum_{i=1}^{D} \frac{\lambda^2_{z,j}}{\lambda^2_{x,i}} \langle e_{x,i}, e_{z,j} \rangle^2 \mathbb{1}[\lambda_{x,i} > 0]$$

$\text{Var}_{z,j}$ is closely tied with the Spectral Inflation of an eigenvector, as $\text{Var}_{z,j}$ 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. However, $\text{Bias}_{z,j}$ is determined by the cosine similarity between the true labelling regressor w^* and $e_{z,j}$. 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.

A regressor that is orthogonal to each eigenvector $e_{z,j}$ such that $\text{Var}_{z,j}$ is greater than $\text{Bias}_{z,j}$ could only perform better out of distribution than $\hat{w}$ (Theorem B.1). However, we can not calculate $\text{Bias}_{z,j}$ directly, and so we must estimate it using $\hat{w}$ and use a probabilistic procedure based on the distribution of these estimates to determine which eigenvectors are displaying Spectral Inflation (Sections B,C). Creating w_{proj} in this way yields SpAR, a regressor tailored for a specific covariate shift; see Algorithm 1 for pseudocode. In SpAR, we estimate the variance of the training label noise, σ^2, using a maximum likelihood estimate of this parameter (Murphy, 2022) from the training data.

Algorithm 1 Spectral Adapted Regressor (SpAR)

Require: Training Data X, Y_X, Unlabeled Test 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**
  $\text{Var}_{z,j} \leftarrow \hat{\sigma^2} \sum_{i=1}^{D} \frac{\lambda^2_{z,j}}{\lambda^2_{x,i}} \langle e_{x,i}, e_{z,j} \rangle^2 \mathbb{1}[\lambda_{x,i} > 0]$
  $\text{Bias}_{z,j} \leftarrow \langle \hat{w}, e_{z,j} \rangle^2 \lambda^2_{z,j}$
  if $(\text{CDF}^{-1}_{\chi^2}(\alpha) \times \text{Var}_{z,j}) \geq \text{Bias}_{z,j}$ **then**
    $S \leftarrow S \cup \{e_{z_j}\}$ {Include this vector in S}
  end if
end for
$w_{\text{proj}} \leftarrow \hat{w} - \sum_{e \in S} \langle \hat{w}, e \rangle e$
return w_{proj}

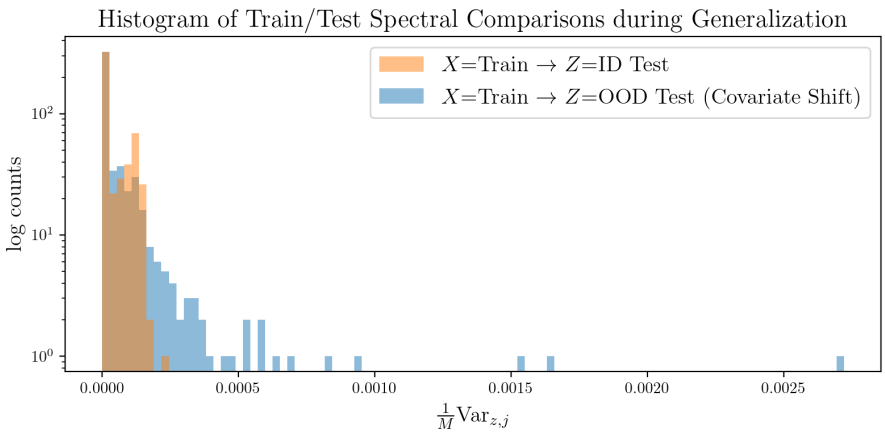

Figure 2. **Spectral Inflation.** We use the PovertyMap-WILDS dataset (Koh et al., 2021) to investigate how input spectra change when a regressor trained on real-world data generalizes to (perhaps shifted) test data. X and Z are composed of representations from a DNN. Z represents data either from an in-distribution or out-of-distribution test set. $\mathrm{Var}_{z,j}$, as defined in Equation 7, 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.

3. Experiments

In this section, we apply SpAR to a suite of real-world datasets to demonstrate its efficacy. Throughout this section, 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 on the training data mean that gradient descent will converge to the OLS solution. We also provide a proof-of-concept synthetic experiment in Appendix K.

3.1. Tabular Datasets

To test the efficacy of SpAR on real-world distribution shifts, we first experiment with two tabular datasets. CommunitiesAndCrime provides a task where crime rates per capita must be predicted for different communities across the United States, with some states held out of the training data and used to form an OOD test set (Redmond and 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 stratifying players based on skill.

For these experiments, we train neural networks with one hidden layer in the style of Yao et al. (2022). We benchmark two methods: the first is standard training (ERM), in which 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 and Ba, 2014). The other method we benchmark is C-Mixup (Yao et al., 2022), a data augmentation technique that generalizes the Mixup algorithm (Zhang et al., 2017) to a regression setting. For this method, the encoder and regressor are optimized using ADAM to minimize the error on both the original samples and the synthetic examples produced by C-Mixup.

Method	Average RMSE (↓)	Worst Group RMSE (↓)
ERM	6.273 ± 0.384	8.933 ± 1.338
ERM + OLS	6.884 ± 0.860	11.156 ± 3.892
ERM + SpAR (Ours)	**6.049** ± 0.379	**8.317** ± 1.327
CMixup	6.319 ± 0.450	8.713 ± 1.106
CMixup + OLS	7.070 ± 0.898	11.747 ± 3.450
CMixup + SpAR (Ours)	**6.038** ± 0.705	**8.343** ± 1.563

Table 1. **SkillCraft.** OOD RMSE averaged across 10 seeds.

We use the hyperparameters reported in the appendix of Yao et al. (2022) when training both ERM and C-Mixup. 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 use a fixed hyperparameter selection of $\alpha = 0.999$ when applying SpAR. These new regressors replace the learned regression weight in the last layer. We similarly benchmark the performance of the Pseudoinverse solution by replacing the last layer weight with $\hat{w}$ (ERM/C-Mixup + OLS). Results on each of the tabular datasets can be found in Tables 1 and 2.

As demonstrated in Tables 2 and 1, SpAR always produces a model with competitive or superior Average and Worst Group RMSE, regardless of the base model that it is applied to. This also demonstrates that SpAR can be used in tandem with other techniques such as C-Mixup to further improve performance. Furthermore, we experiment with

Method	Average RMSE (↓)	Worst Group RMSE (↓)
ERM	0.134 ± 0.006	0.166 ± 0.014
ERM + OLS	0.142 ± 0.004	0.175 ± 0.012
ERM + SpAR (Ours)	**0.133** ± 0.002	**0.163** ± 0.009
CMixup	**0.131** ± 0.005	0.162 ± 0.016
CMixup + OLS	0.140 ± 0.003	0.175 ± 0.010
CMixup + SpAR (Ours)	0.133 ± 0.002	**0.161** ± 0.004

Table 2. **CommunitiesAndCrime.** OOD RMSE (avg. 10 seeds).

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
CMixup	0.784 ± 0.045	0.489 ± 0.045
CMixup + SpAR (Ours)	**0.794** ± 0.043	**0.515** ± 0.091

Table 3. **PovertyMap-WILDS.** Average OOD all-group and worst-group Spearman r across 5 splits.

models obtained by tuning the hyperparameters for both ERM and C-Mixup in Appendix N. We find that with **no additional tuning for SpAR specifically**, SpAR yields a model with the best worst-group performance amongst any of the models presented in this work.

3.2. PovertyMap - WILDS

We next examine the robustness of DNNs under realistic distribution shifts in a high-dimensional setting. We use the PovertyMap-WILDS dataset (Koh et al., 2021), where the task is to regress local satellite images onto a continuous target representing an asset wealth index for the region. Figure 2 demonstrates that DNNs attempting to generalize OOD on this dataset suffer from Spectral Inflation.

Once again, for ERM and C-Mixup we use the hyperparameters suggested by Yao et al. (2022) and for SpAR we use $\alpha = 0.999$. Results are presented in Table 3.

We can observe from Table 3 that applying SpAR can significantly improve worst-group performance while maintaining competitive average performance. We further tune the hyperparameters for both the ERM and C-Mixup baselines in Appendix N. With **no tuning of SpAR specifically**, it is able to enhance the tuned baseline and yield worst-group performance superior to any model presented in this work.

4. Related Work

Improving OOD distribution is a critical and dynamic area of research. Our approach follows in the tradition of Transductive Learning (Gammerman et al., 1998) (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., 2016). When transductive reasoning (adaptation to a test distribution) is not possible, additional structure in P—such as auxiliary labels indicating the "domain" or "group" that each training example belongs to—may be exploited to promote OOD generalization. Noteworthy approaches include Domain Generalization (Arjovsky et al., 2019; Gulrajani and Lopez-Paz, 2020) and Distributionally Robust Optimization (Hu et al., 2018; Sagawa et al., 2019; Levy et al., 2020). Additionally, Kirichenko et al. (2022) leverage an additional dataset containing balanced groups in order to train a new, robust classification head, similar to how SpAR creates a new regressor while leaving the encoder unchanged.

Data augmentation is another promising avenue for improving OOD generalization (Hendrycks and Dietterich, 2019; Ovadia et al., 2019) that artificially increases the number and diversity of training set samples. The recently proposed C-Mixup method is particularly relevant (Yao et al., 2022) because of its focus on regression under covariate shift. The authors generalize the Mixup algorithm (Zhang et al., 2017) to a regression setting by upweighting the convex combination of training examples whose target values are similar. We emphasize that 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. We are not the first to study this problem, nor the first to use spectral properties in this investigation. 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) analyzes the sample efficiency of linear regression in terms of an eigendecomposition of the second moment matrix of individual data points drawn from 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.

5. Conclusion

Our analysis shows that the Ordinary Least Squares solution—which minimizes the training risk—can fail dramatically when subject to covariate-shift due a phenomenon we call *Spectral Inflation*. Our new method, SpAR, combats Spectral Inflation and leads to improved OOD performance on several synthetic and real-world datasets.

REFERENCES

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Aparna Balagopalan, Jekaterina Novikova, Matthew BA Mcdermott, Bret Nestor, Tristan Naumann, and Marzyeh Ghassemi. Cross-language aphasia detection using optimal transport domain adaptation. In *Machine Learning for Health Workshop*, pages 202–219. PMLR, 2020.

Michelle Bao, Angela Zhou, Samantha Zottola, Brian Brubach, Sarah Desmarais, Aaron Horowitz, Kristian Lum, and Suresh Venkatasubramanian. It's compaslicated: The messy relationship between rai datasets and algorithmic fairness benchmarks. *arXiv preprint arXiv:2106.05498*, 2021.

David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *arXiv preprint arXiv:2005.07727*, 2020.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

Mark Blair, Joe Thompson, Andrew Henrey, and Bill Chen. Skillcraft1 master table dataset. *UCI Machine Learning Repository*, 2013.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

Alex J DeGrave, Joseph D Janizek, and Su-In Lee. Ai for radiographic covid-19 detection selects shortcuts over signal. *Nature Machine Intelligence*, 3(7):610–619, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. *Advances in neural information processing systems*, 34:6478–6490, 2021.

Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia. A brief review of domain adaptation. *Advances in Data Science and Information Engineering: Proceedings from ICDATA 2020 and IKE 2020*, pages 877–894, 2021.

A Gammerman, V Vapnik, and VI Vovk. Learning by transduction. 1998.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

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

Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.

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

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning*, pages 2029–2037. PMLR, 2018.

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

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv preprint arXiv:2204.02937*, 2022.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Daniel Levy, Yair Carmon, John C Duchi, and Aaron Sidford. Large-scale methods for distributionally robust optimization. *Advances in Neural Information Processing Systems*, 33:8847–8860, 2020.

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

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

Reese Pathak, Cong Ma, and Martin Wainwright. A new similarity measure for covariate shift with applications to nonparametric regression. In *International Conference on Machine Learning*, pages 17517–17530. PMLR, 2022.

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

Yangjun Ruan, Yann Dubois, and Chris J Maddison. Optimal representations for covariate shift. *arXiv preprint arXiv:2201.00057*, 2021.

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3118–3126, 2018.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. *arXiv preprint arXiv:1612.01939*, 2016.

Yin Sun and Árpád Baricz. Inequalities for the generalized marcum q-function. *Applied Mathematics and Computation*, 203(1):134–141, 2008.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020.

Jingfeng Wu, Difan Zou, Vladimir Braverman, Quanquan Gu, and Sham Kakade. The power and limitation of pretraining-finetuning for linear regression under covariate shift. *Advances in Neural Information Processing Systems*, 35:33041–33053, 2022.

Huaxiu Yao, Yiping Wang, Linjun Zhang, James Y Zou, and Chelsea Finn. C-mixup: Improving generalization in regression. *Advances in Neural Information Processing Systems*, 35:3361–3376, 2022.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

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:

$$\hat{w}_{OLS} = (X^\top X)^{-1} X^\top Y_X \tag{8}$$

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:

$$\hat{w} = X^\dagger Y_X = V_X D_X^\dagger U_X^\top Y_X \tag{9}$$

B. 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 $S \subseteq \text{Rows}(V_Z^\top)$. In this subsection, we present a means for constructing the set S to minimize the expected loss by comparing $\text{Var}_{z,j}$ and $\text{Bias}_{z,j}$ for each test eigenvector $e_{z,j}$.

The ideal set $S^* \subseteq \text{Rows}(V_Z^\top)$ 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:

$$S^* = \left\{ e_{z,j} : e_{z,j} \in \text{Rows}(V_Z^\top), \text{Var}_{z,j} \geq \text{Bias}_{z,j} \right\}. \tag{10}$$

The following theorem demonstrates that using the set S^* would give us a regressor that achieves superior OOD performance than the pseudoinverse solution.

Theorem B.1. *Under the same assumptions as Theorem 2.1, the regressor w_{proj} constructed as in Equation 5 using the set S^* (cf. Equation 10) can only improve on the OOD squared error loss of the pseudoinverse solution $\hat{w}$:*

$$\mathbb{E}[\|Y_Z - Z\hat{w}\|_2^2] \geq \mathbb{E}[\|Y_Z - Z w_{\text{proj}}\|_2^2]. \tag{11}$$

C. Eigenvector Selection Under Uncertainty

Theorem B.1 shows that a regressor based on the set S^* works better OOD. Finding S^* would be easy if we knew both $\text{Var}_{z,j}$ and $\text{Bias}_{z,j}$ for each test eigenvector $e_{z,j}$. While we can calculate $\text{Var}_{z,j}$ directly, $\text{Bias}_{z,j}$ requires the true weight vector w^*, and so we can only *estimate* it using the pseudoinverse solution $\hat{w}$:

$$\widehat{\text{Bias}}_{z,j} = \langle \hat{w}, e_{z,j} \rangle^2 \lambda_{z,j}^2 = (w^{*T} e_{z,j} + \epsilon^\top X^{\dagger\top} e_{z,j})^2 \lambda_{z,j}^2. \tag{12}$$

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.

$$\epsilon^\top X^{\dagger\top} e_{z,j} \lambda_{z,j} \sim \mathcal{N}(0, \text{Var}_{z,j}), \tag{13}$$

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

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

(Case 1): $\text{Bias}_{z,j} \gg \text{Var}_{z,j}$

In this case, $\text{Bias}_{z,j} \approx \widehat{\text{Bias}}_{z,j}$. 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 12. Therefore $\widehat{\text{Bias}}_{z,j} \gg \text{Var}_{z,j}$.

(Case 2): $\text{Var}_{z,j} \gg \text{Bias}_{z,j}$

In this case, $\widehat{\text{Bias}}_{z,j} \approx (\epsilon^\top X^{\dagger\top} e_{z,j})^2 \lambda_{z,j}^2$. 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 12.

Therefore, since Equation 13 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:

$$\widehat{\text{Bias}}_{z,j} \sim \text{Var}_{z,j} \times \chi^2_{df=1}, \tag{14}$$

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

$$\Pr(\widehat{\text{Bias}}_{z,j} \leq \text{CDF}^{-1}_{\chi^2_{df=1}}(\alpha) \times \text{Var}_{z,j}) = \alpha. \tag{15}$$

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

$$S = \left\{ e_{z,j} : \widehat{\text{Bias}}_{z,j} \leq \text{CDF}^{-1}_{\chi^2_{df=1}}(\alpha) \times \text{Var}_{z,j} \right\}. \tag{16}$$

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

Proposition C.1. *Making the same assumptions as Theorem 2.1, 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 16:*

$$\Pr(e_{z,j} \in S) = 1 - Q_{\frac{1}{2}}\left(\sqrt{\frac{\text{Bias}_{z,j}}{\text{Var}_{z,j}}}, \sqrt{\text{CDF}^{-1}_{\chi^2_{df=1}}(\alpha)} \right), \tag{17}$$

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

Lemma C.2. *Using the same assumptions as Proposition C.1:*

$$\Pr(e_{z,j} \in S) \xrightarrow{\frac{\text{Bias}_{z,j}}{\text{Var}_{z,j}} \to \infty} 0, \qquad \Pr(e_{z,j} \in S) \xrightarrow{\frac{\text{Bias}_{z,j}}{\text{Var}_{z,j}} \to 0} \alpha. \tag{18}$$

Lemma C.2 tells us that if we would incur significantly higher OOD loss from including $e_{z,j}$ in our set S than excluding it, then this vector **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 this vector **will** be included in S.

Creating w_{proj} in this way yields SpAR, a regressor tailored for a specific covariate shift; see Algorithm 1 for pseudocode. The only quantity still needed to perform this procedure is the variance of the training label noise, σ^2. We use a maximum likelihood estimate of this parameter (Murphy, 2022) from the training data.

D. Derivation of Loss of OLS Under Covariate Shift

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

$$\text{Risk}_{\text{OOD}}(\hat{w}) = \mathbb{E}[\|Y_Z - Z\hat{w}\|_2^2] = \mathbb{E}[\|Zw^* - ZX^\dagger Y\|_2^2] \tag{19}$$

$$= \mathbb{E}[\|Zw^* - ZX^\dagger(Xw^* + \epsilon)\|_2^2].$$

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:

$$= \mathbb{E}[\|ZX^\dagger\epsilon\|_2^2] \tag{20}$$

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:

$$= \mathbb{E}[\epsilon^\top X^{\dagger\top} Z^\top ZX^\dagger\epsilon] = \mathbb{E}[tr(\epsilon^\top X^{\dagger\top} Z^\top ZX^\dagger\epsilon)] \tag{21}$$

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

$$= \mathbb{E}[tr(\epsilon\epsilon^\top X^{+T} Z^\top ZX^\dagger)] = \mathbb{E}[\sum_{i=1}^{N}\sum_{j=1}^{N}(\epsilon\epsilon^\top)_{i,j}(X^{+T}Z^\top ZX^\dagger)_{i,j}] \tag{22}$$

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:

$$= \sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{E}[(\epsilon\epsilon^\top)_{i,j}](X^{+T}Z^\top ZX^\dagger)_{i,j} \tag{23}$$

$$= \sum_{i=1}^{N}\sigma^2(X^{+T}Z^\top ZX^\dagger)_{i,i} = \sigma^2 tr(X^{+T}Z^\top ZX^\dagger)$$

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

$$= \sigma^2 tr(Z^\top ZX^\dagger X^{+T}) \tag{24}$$

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

$$= \sigma^2 tr(V_Z D_Z^2 V_Z^\top V_X D_X^{\dagger 2} V_X^\top) = \sigma^2 tr(D_X^{\dagger 2} V_X^\top V_Z D_Z^2 V_Z^\top V_X) \tag{25}$$

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:

$$\left[diag(V_X^\top V_Z D_Z^2 V_Z^\top V_X)\right]_i = \sum_{j=1}^{D}\lambda_{z,j}^2\langle e_{x,i}, e_{z,j}\rangle^2 \tag{26}$$

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

$$\text{Risk}_{\text{OOD}}(\hat{w}) = \sigma^2\sum_{i=1}^{D}\sum_{j=1}^{D}\frac{\lambda_{z,j}^2}{\lambda_{x,i}^2}\langle e_{x,i}, e_{z,j}\rangle^2\mathbb{1}[\lambda_{x,i} > 0]. \tag{27}$$

E. Derivation of Bias-Variance Decomposition

SpAR produces a regressor of the following form:

$$w_{\text{proj}} = \hat{w} - \sum_{e \in S} \langle \hat{w}, e \rangle e \tag{28}$$

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.

$$\text{Risk}_{\text{OOD}}(w_{\text{proj}}) = \mathbb{E}[\|Zw^* - Z(\hat{w} - \sum_{e_{z,j} \in S} \langle \hat{w}, e_{z,j} \rangle e_{z,j})\|_2^2] \tag{29}$$

$$= \mathbb{E}[\| - ZV_X D_X^\dagger U_X^\top \epsilon + Z \sum_{e_{z,j} \in S} (\epsilon^\top X^{\dagger\top} e_{z,j} + w^{*\top} e_{z,j}) e_{z,j}\|_2^2] \tag{30}$$

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

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:

$$= \mathbb{E}[\| - Z \sum_{j=1}^{D} e_{z,j} e_{z,j}^\top V_X D_X^\dagger U_X^\top \epsilon + Z \sum_{e_{z,j} \in S} \langle w^*, e_{z,j} \rangle e_{z,j} + Z \sum_{e_{z,j} \in S} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j}\|_2^2] \tag{32}$$

$$= \mathbb{E}[\| - Z \sum_{j=1}^{D} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j} + Z \sum_{e_{z,j} \in S} \langle w^*, e_{z,j} \rangle e_{z,j} + Z \sum_{e_{z,j} \in S} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j}\|_2^2] \tag{33}$$

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:

$$\mathbb{E}[\| - Z \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j} + Z \sum_{e_{z,j} \in S} \langle w^*, e_{z,j} \rangle e_{z,j}\|_2^2] = \mathbb{E}[\|V - B\|_2^2] \tag{34}$$

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 2.1), restricted to the eigenvectors that weren't projected out:

$$V^\top V = (Z \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j})^\top Z \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j} \tag{35}$$

$$= (\sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j})^\top Z^\top Z \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j} \tag{36}$$

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

$$= \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle e_{z,j}^\top \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle \lambda_{z,j}^2 e_{z,j} \tag{37}$$

$$= \sum_{e'_{z,j} \in S^c} \sum_{e_{z,j} \in S^c} \langle V_X D_X^\dagger U_X^\top \epsilon, e'_{z,j} \rangle \langle V_X D_X^\dagger U_X^\top \epsilon, e_{z,j} \rangle \lambda_{z,j}^2 e'^\top_{z,j} e_{z,j} \tag{38}$$

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, $e'^{\top}_{z,j} e_{z,j} = 0$.

$$= \sum_{e_{z,j} \in S^c} \langle V_X D_X^{\dagger} U_X^{\top} \epsilon, e_{z,j} \rangle^2 \lambda_{z,j}^2 = \sum_{e_{z,j} \in S^c} \epsilon^{\top} X^{\dagger\top} e_{z,j} e_{z,j}^T X^{\dagger} \epsilon \lambda_{z,j}^2 \tag{39}$$

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

$$\mathbb{E}[V^{\top} V] = \mathbb{E}[\sum_{e_{z,j} \in S^c} \epsilon^{\top} X^{\dagger\top} e_{z,j} e_{z,j}^T X^{\dagger} \epsilon \lambda_{z,j}^2] \tag{40}$$

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

$$= \sum_{e_{z,j} \in S^c} \sigma^2 tr(e_{z,j}^{\top} X^{\dagger} X^{\dagger\top} e_{z,j}) \lambda_{z,j}^2 \tag{41}$$

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:

$$e_{z,j}^{\top} X^{\dagger} X^{\dagger\top} e_{z,j} = \sum_{i=1}^{d} \sum_{k=1}^{d} \frac{1}{\lambda_{x,i}} \langle e_{z,j}, e_{x,i} \rangle \frac{1}{\lambda_{x,k}} \langle e_{z,j}, e_{x,k} \rangle u_{x,i}^{\top} u_{x,k} \mathbb{1}[\lambda_{x,i} > 0] \mathbb{1}[\lambda_{x,k} > 0] \tag{42}$$

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:

$$\mathbb{E}[V^{\top} V] = \sigma^2 \sum_{i=1}^{D} \sum_{j, e_{z,j} \in S^c} \frac{\lambda_{z,j}}{\lambda_{x,i}} \langle e_{x,i}, e_{z,j} \rangle^2 \mathbb{1}[\lambda_{x,i} > 0] \tag{43}$$

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

$$\mathbb{E}[B^T B] = B^T B \tag{44}$$

$$= \sum_{e_{z,j} \in S} \langle w^*, e_{z,j} \rangle e_{z,j}^{\top} Z^{\top} Z \sum_{e_{z,j} \in S} \langle w^*, e_{z,j} \rangle e_{z,j} = \sum_{j, e_{z,j} \in S} \langle w^*, e_{z,j} \rangle^2 \lambda_{z,i}^2 \tag{45}$$

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

$$\mathbb{E}[\| Z w^* - Z(\hat{w} - \sum_{e_{z,j} \in S} \langle \hat{w}, e_{z,j} \rangle e_{z,j}) \|_2^2] = \mathbb{E}[V^{\top} V] + \mathbb{E}[B^{\top} B] \tag{46}$$

$$= \sigma^2 \sum_{i=1}^{D} \sum_{j, e_{z,j} \in S^c} \frac{\lambda_{z,j}}{\lambda_{x,i}} \langle e_{x,i}, e_{z,j} \rangle^2 \mathbb{1}[\lambda_{x,i} > 0] + \sum_{j, e_{z,j} \in S} \langle w^*, e_{z,j} \rangle^2 \lambda_{z,j}^2 \tag{47}$$

F. Appendix: Proof of Theorem B.1

In this section, we provide the proof of Theorem B.1.

To start, we note that if $S_\phi = \{\}$, then the projected regressor created with this set is the pseudoinverse solution $\hat{w}$:

$$w_{projS_\phi} = \hat{w} - \sum_{e \in S_\phi} \langle \hat{w}, e \rangle e = \hat{w}. \tag{48}$$

Therefore, by Theorem 2.2, we know that the loss of this regressor will consist entirely of the variance terms associated with the eigenvectors. This is essentially a recovery of Theorem 2.1:

$$\mathbb{E}[\|Y_Z - Z\hat{w}\|_2^2] = \sum_{j=1}^{D} \sigma^2 \sum_{i=1}^{D} \frac{\lambda_{z,j}^2}{\lambda_{x,i}^2} \langle e_{x,i}, e_{z,j} \rangle^2 \mathbb{1}[\lambda_{x,i} > 0] = \sum_{j=1}^{D} \mathrm{Var}_{z,j}. \tag{49}$$

We now compare this with the loss of the regressor created using the set S^*, which is:

$$w^*_{\mathrm{proj}} = \hat{w} - \sum_{e \in S^*} \langle \hat{w}, e \rangle e \tag{50}$$

Again invoking Theorem 2.2, the expected loss of this estimator is:

$$\mathbb{E}[\|Y_Z - Z w^*_{\mathrm{proj}}\|_2^2] = \sum_{e_{z,j} \in S^{*c}} \sigma^2 \sum_{i=1}^{D} \frac{\lambda_{z,j}^2}{\lambda_{x,i}^2} \langle e_{x,i}, e_{z,j} \rangle^2 \mathbb{1}[\lambda_{x,i} > 0] + \sum_{e_{z,j} \in S^*} \langle w^*, e_{z,j} \rangle^2 \lambda_{z,i}^2 \tag{51}$$

$$= \sum_{e_{z,j} \in S^{*c}} \mathrm{Var}_{z,j} + \sum_{e_{z,j} \in S^*} \mathrm{Bias}_{z,j} \tag{52}$$

We can now compare the two expected losses:

$$\mathbb{E}[\|Y_Z - Z\hat{w}\|_2^2] - \mathbb{E}[\|Y_Z - Z w^*_{\mathrm{proj}}\|_2^2] = \sum_{e_{z,j} \in S^*} \mathrm{Var}_{z,j} - \mathrm{Bias}_{z,j} \tag{53}$$

From the definition of S^*, we know that for all $e_{z,j} \in S^*, \mathrm{Var}_{z,j} \geq \mathrm{Bias}_{z,j}$. Therefore, for all $e_{z,j} \in S^*, \mathrm{Var}_{z,j} - \mathrm{Bias}_{z,j} \geq 0$, and the sum of these terms will be greater than zero as well. This gives us:

$$\mathbb{E}[\|Y_Z - Z\hat{w}\|_2^2] - \mathbb{E}[\|Y_Z - Z w^*_{\mathrm{proj}}\|_2^2] \geq 0 \implies \mathbb{E}[\|Y_Z - Z\hat{w}\|_2^2] \geq \mathbb{E}[\|Y_Z - Z w^*_{\mathrm{proj}}\|_2^2] \tag{54}$$

G. Distribution of $\widehat{\mathrm{Bias}}$

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

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

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

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

With this in mind, 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 C:

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

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

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

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

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

H. Proof of Proposition 1

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

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

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

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

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

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

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

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

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

I. Proof of Lemma C.2

Proposition C.1 gives us an expression for the probability that a given eigenvector is included in the set S. Lemma C.2 will use this proposition to demonstrate the tail behaviour of this expression. We will first note that since the expression in Proposition C.1 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 and 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 and 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.

J. Spectral Inflation in Real World Datasets

In this work, we have focussed on a phenomenon we have called *Spectral Inflation*. Our method, SpAR, identifies eigenvectors presenting Spectral Inflation and produces a regressor orthogonal to these directions. We present evidence of Spectral Inflation occurring in DNN representations in a real-world dataset in Figure 2.

K. 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 2.1, with the exception that we add normally distributed noise to the test target labels

$Y_Z = Zw^* + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Specifically, we sample our train and test data X and Z from origin-centered Gaussians with covariance matrices Σ_X, Σ_Z respectively:

$$\Sigma_X = \begin{bmatrix} 5 & 0 \\ 0 & 10^{-5} \end{bmatrix}, \qquad \Sigma_Z = \begin{bmatrix} 1 & 0 \\ 0 & 40 \end{bmatrix}.$$

We refer to the first and second indices of these vectors as the "horizontal" and "vertical" components and plot the vectors accordingly. The test distribution will demonstrate a significantly greater amount of variance along the vertical component in comparison to the training distribution. Furthermore, we experiment with three different true labelling vectors:

$$w_1^* = \begin{bmatrix} 0.01 \\ 0.9999995 \end{bmatrix}, \qquad w_2^* = \begin{bmatrix} 0.9999995 \\ 0.01 \end{bmatrix}, \qquad w_3^* = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{bmatrix}.$$

The first two true labelling vectors are meant to 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. This labelling vector is shown in Figure 1. For each of the three labelling vectors, we randomly sample Z, X and ϵ 10 times and calculate the squared error for both the OLS/Pseudoinverse solution $\hat{w} = X^\dagger Y_X$ (ERM) as well as w_{proj}, the regressor outputted by SpAR (ERM + SpAR). Results are included in Table 4.

Synthetic Data			
Regression Method	Experiment 1 (w_1^*)	Experiment 2 (w_2^*)	Experiment 3 (w_3^*)
ERM	2.54e6 $\pm$ 3.84e6	2.54e6 $\pm$ 3.84e6	2.54e6 $\pm$ 3.84e6
ERM + SpAR	**1.63e6** $\pm$ 3.43e3	**3.99e3**$\pm$1.01e2	**1.31e5** $\pm$ 2.74e3

Table 4. Mean and standard deviation of the squared error of our estimated regressors against various true labelling vectors. Each experiment setting is a different true weight vector (see Section K).

We first note that $\hat{w}$ is expected to have the same error regardless of the true labelling vector. Second, w_{proj} outperforms $\hat{w}$ regardless of which true regressor is chosen. Our projection method is most effective when w_2^* is being used to label the examples. This is due to the fact that 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 labelling 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 labelling function nearly identical to the true labelling function.

L. Additional Training Details

For our experiments in Section 3, 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 following learning rates and bandwidth parameters for C-Mixup:

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

Table 5. Hyperparameters used for training models responsible for the results in Section 3.

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.

M. 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 5 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 hyperparameter settings selected for C-Mixup are presented in Table 6 and hyperparameter settings selected for ERM are presented in Table 7.

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

Table 6. Tuned hyperparameters used for training C-mixup models. 3.

Hyperparameter	CommunitiesAndCrime	SkillCraft	PovertyMap
Learning Rate	0.008246671732726021	0.023276939100527687	0.003630376073213171

Table 7. Tuned hyperparameters used for training ERM models.

N. Tuned Baselines

Using the hyperparameters presented in Tables 6 and 7 which were selected hyperparameter search process described in Section M, 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 9 and 8, respectively.

We find that SpAR can achieve superior worst group performance than any other method presented in either Tables 9 or 8, or in Section 3. For C-Mixup on CommunitiesAndCrime, we see that tuning hyperparameters on the validation set yields poorer performance (Table 8) than using the hyperparameters presented in Yao et al. (2022)'s work (Table 2). 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.

O. Limitations and Broader Impacts

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.

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

SkillCraft		
Method	Average RMSE (↓)	Worst Group RMSE (↓)
ERM	**5.917** ± 0.620	8.308 ± 1.915
ERM + OLS	6.548 ± 0.915	10.219 ± 3.123
ERM + SpAR (Ours)	6.083 ± 0.681	**8.193** ± 1.212
CMixup	**5.816** ± 0.558	8.371 ± 1.611
CMixup + OLS	6.535 ± 0.822	10.297 ± 2.362
CMixup + SpAR (Ours)	5.833 ± 0.580	**7.922** ± 1.043

CommunitiesAndCrime		
Method	Average RMSE (↓)	Worst Group RMSE (↓)
ERM	**0.133** ± 0.004	**0.161** ± 0.010
ERM + OLS	0.149 ± 0.018	0.184 ± 0.032
ERM + SpAR (Ours)	0.134 ± 0.007	0.164 ± 0.013
CMixup	0.133 ± 0.003	0.171 ± 0.012
CMixup + OLS	0.144 ± 0.011	0.177 ± 0.019
CMixup + SpAR (Ours)	**0.132** ± 0.004	**0.164** ± 0.008

Table 8. **Tabular data.** OOD RMSE averaged across 10 seeds.

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

Table 9. **PovertyMap-WILDS.** Average OOD all-group and worst-group Spearman r across 5 splits.

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).