Correct-N-Contrast: A Contrastive Approach for Improving Robustness to Spurious Correlations

Michael Zhang*[†], Nimit S. Sohoni[††], Hongyang R. Zhang[‡], Chelsea Finn[†] & Christopher Ré[†]
[†]Department of Computer Science, Stanford University
[††]Institute for Computational and Mathematical Engineering, Stanford University
[‡]Khoury College of Computer Sciences, Northeastern University
{mzhang, nims, cbfinn, chrismre}@cs.stanford.edu
ho.zhang@northeastern.edu

Abstract

We propose Correct-N-Contrast (CNC), a contrastive learning method to improve robustness to spurious correlations when training group labels are unknown. Our motivating observation is that worst-group performance is related to a representation alignment loss, which measures the distance in feature space between different groups within each class. We prove that the gap between worst-group and average loss for each class is upper bounded by this alignment loss for that class. Thus, CNC aims to improve representation alignment via contrastive learning. First, CNC uses an ERM model to infer the group information. Second, with a careful sampling scheme, CNC trains a contrastive model to encourage similar representations for groups in the same class. We show that CNC significantly improves worst-group accuracy over existing state-of-the-art methods on popular benchmarks, e.g., achieving 7.7% absolute lift in worst-group accuracy on the CelebA dataset, and performs almost as well as methods trained with group labels. CNC also learns better-aligned representations between different groups in each class, reducing the alignment loss substantially compared to prior methods.

1 Introduction

For many tasks, deep neural networks are negatively affected by spurious correlations—dependencies between observed features and class labels that only hold for certain groups of the data. For example, a model may learn to classify animals solely based on background [35, 6, 19]. In this work, we focus on improving a model's robustness to spurious correlations via its learned *representations*. We support this direction with two key motivations. First, we find that higher worst-group performance consistently correlates with hidden-layer representations exhibiting higher dependence on class labels than spurious attributes. We quantify this via a representation *alignment* loss [47], which measures the closeness of sample representations with the same class but different spurious attributes, and mutual information. Second, we theoretically show that this alignment loss for a class can upper bound the worst-group vs. average loss gap for that class. Thus, if we can reduces alignment loss for a class, we can reduce the gap between worst-group and average loss for that class.

We thus propose Correct-N-Contrast (CNC), a two-stage procedure using contrastive learning to encourage better representation alignment within each class. First, we train an ERM model similar to prior work [25, 12], under the premise that ERM predictions help infer group information (i.e., spurious attributes). Second, we improve representation alignment by "pulling together" same-class datapoints and "pushing apart" different-class datapoints, regardless of their spurious features. To do so via contrastive learning, we reason that samples with the same ERM predictions exhibit similar spurious features (and vice versa). With a randomly sampled anchor, we select samples with the same class but different ERM predictions as "positives" to pull together, and samples from different classes

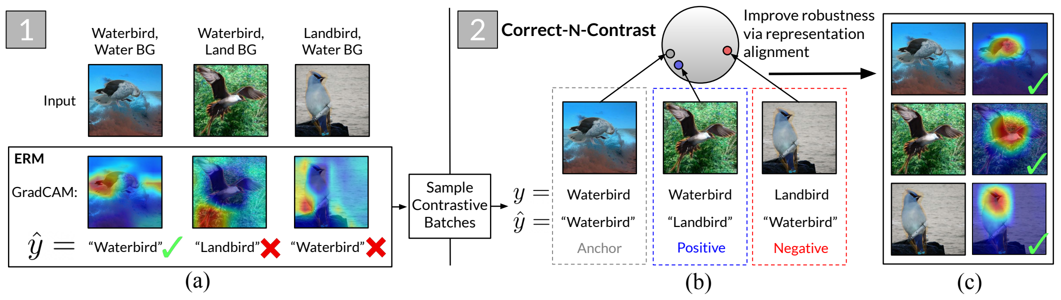

1
Waterbird, Water BG
Waterbird, Land BG
Landbird, Water BG
Input
ERM
GradCAM:
$\hat{y}$ = "Waterbird" ✓ "Landbird" ✗ "Waterbird" ✗
(a)
Sample Contrastive Batches
2
Correct-N-Contrast
Improve robustness via representation alignment
y = Waterbird Waterbird Landbird
$\hat{y}$ = "Waterbird" "Landbird" "Waterbird"
Anchor
Positive
Negative
(b)
(c)

Figure 1: (a) ERM misclassifies samples by spurious background features, visualized with GradCAM [38]. (b) CNC uses contrastive learning to learn similar representations for same-class samples with different ERM predictions. (c) Resulting models ignore spurious attributes and classify samples correctly.

but the same ERM prediction as *hard* "negatives" to push apart. Training a second model with this sampling scheme and a contrastive loss encourages this model to ignore spurious correlations that the initial ERM model learned, and improves representation alignment between same-class data points. Thus, CNC corrects for the ERM model's mistakes with contrastive learning in the second model.

We evaluate CNC on four popular and diverse spurious correlation benchmarks. Among methods that similarly do not assume training group labels, CNC substantially improves worst-group accuracy, obtaining up to **7.7%** absolute lift over the prior state-of-the-art JTT [25] (from 81.1% to **88.8%** on CelebA), and averaging **3.4%** lift across the four tasks. We also find that CNC nearly closes the gap in worst-group accuracy with methods that assume training group labels such as group DRO (GDRO) [37], only falling short of GDRO's worst-group accuracy by 0.3% absolute. Finally, we validate that CNC indeed reduces the alignment loss compared to prior methods. This corresponds to an up to 71.1% smaller gap between worst-group versus average accuracy for data points in the same class.

2 Preliminaries

Problem setup. Our setting follows that of Sagawa et al. [37]. Let $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_n\}$ be a training dataset of size n. Each data point has an observed feature vector $x_i \in \mathcal{X}$, label $y_i \in \mathcal{Y}$, and *unobserved* spurious attribute $a_i \in \mathcal{A}$. The set of groups $\mathcal{G}$ is defined as the set of all combinations of class label and spurious attribute pairs, i.e. $\mathcal{G} = \mathcal{Y} \times \mathcal{A}$. Let $C = |\mathcal{Y}|$ be the number of classes and $K = |\mathcal{G}|$ be the number of groups. We assume that each example (x_i, y_i, a_i) is drawn from an unknown joint distribution P, with at least one sample from each group observed in the training data. Let P_g be the distribution conditioning on $(y, a) = g$, for any $g \in \mathcal{G}$. Spurious correlations can cause ERM to obtain high error on minority groups even when average error is low. We tackle the setting where group labels are *not available* during training.

Contrastive learning. We briefly describe contrastive learning [9] as it applies here. Let f_θ be a neural network with parameters θ. Let the encoder $f_{\text{enc}} : \mathcal{X} \mapsto \mathbb{R}^d$ be the feature representation layers of f_θ. Let $f_{\text{cls}} : \mathbb{R}^d \mapsto \mathbb{R}^C$ be the classification layer of f_θ, which maps encoder representations to one-hot label vectors. We learn f_{enc} with a *supervised contrastive loss* $\mathcal{L}_{\text{con}}^{\text{sup}}$ [20]. For each anchor x, we sample M positives $\{x_i^+\}_{i=1}^M$ and N negatives $\{x_i^-\}_{i=1}^N$. Let $y, \{y_i^+\}_{i=1}^M, \{y_i^-\}_{i=1}^N$ be the labels and $z, \{z_i^+\}_{i=1}^M, \{z_i^-\}_{i=1}^N$ be the normalized outputs of $f_{\text{enc}}(x)$ for the anchor, positives, and negatives respectively. With input x mapped to z, the training objective for the encoder is to minimize:

$$\mathcal{L}_{\text{con}}^{\text{sup}}(x; f_{\text{enc}}) = \mathop{\mathbb{E}}_{x, \{x_i^+\}_{i=1}^M, \{x_i^-\}_{j=1}^N} \left[-\log \frac{\exp(z^\top z_i^+/\tau)}{\sum_{m=1}^M \exp(z^\top z_m^+/\tau) + \sum_{n=1}^N \exp(z^\top z_n^-/\tau)} \right] \quad (1)$$

where $\tau > 0$ is a scalar temperature hyperparameter. We discuss further related work in Appendix C.

3 Motivations for representation alignment

3.1 Empirical relation between worst-group performance and representation alignment

Experimental setup. We model spurious correlations with CMNIST*, a colored MNIST dataset inspired by [3]. There are 5 digit classes and 5 colors. We color a fraction p_{corr} of the training samples with a color a associated with each class y, and color the test samples uniform-randomly. To analyze learned representations, we train a LeNet-5 CNN [23] with ERM to predict digit classes, and inspect the outputs of the last hidden layer $z = f_{\text{enc}}(x)$ using the following representation metrics.

Representation metrics. First, we compute an *alignment loss* $\hat{\mathcal{L}}_{\text{align}}(f_{\text{enc}}; g, g')$ for two groups $g = (y, a)$ and $g' = (y, a')$ where $a \neq a'$. This measures how closely f_{enc} maps samples with the same class, but different spurious attributes, via Euclidean distance. Letting G and G' be the training data subsets in groups g and g' respectively, and x and x' be any two samples in G and G', we define:

$$\hat{\mathcal{L}}_{\text{align}}(f_{\text{enc}}; g, g') := \frac{1}{|G|}\frac{1}{|G'|} \sum_{(x,y,a)\in G} \sum_{(x',y,a')\in G'} \|f_{\text{enc}}(x) - f_{\text{enc}}(x')\|_2. \tag{2}$$

Thus, lower $\hat{\mathcal{L}}_{\text{align}}$ means better alignment. We also quantify representation dependence by estimating the mutual information (MI) of a model's learned representations with the class label, i.e. $\hat{I}(Y; Z)$ and the spurious attributes $\hat{I}(A; Z)$. We defer computational details to Appendix D.

Results. In Fig. 2 we show a strong association between worst-group error and alignment and mutual information. As p_{corr} increases, ERM models not only drop in worst-group accuracy, but also incur higher alignment loss (Fig. 2ab). Fig. 2c further illustrates this with mutual information. We plot the estimated mutual information and worst-group accuracy for models at each epoch. High worst-group accuracy (>80%) corresponds to a combination of both high $\hat{I}(Y; Z)$ and low $\hat{I}(A; Z)$. Fig. 2d also captures this trend with a trade off between high $\hat{I}(Y; Z)$ with $\hat{I}(A; Z)$ as p_{corr} increases (Fig. 2a).

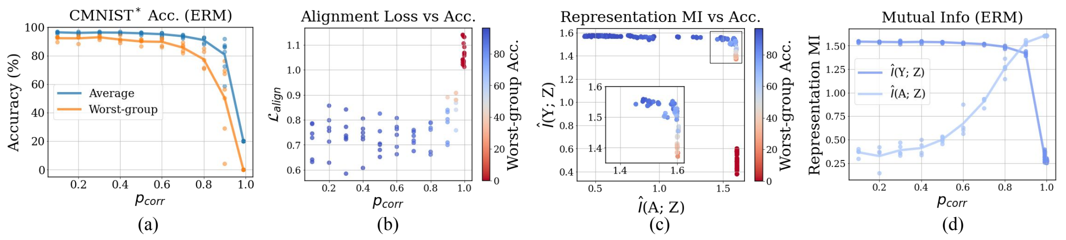


Figure 2: Accuracy and representation metrics from ERM models trained on increasingly spuriously correlated Colored MNIST. High worst-group accuracy corresponds to both $\hat{I}(Y; Z) \gg \hat{I}(A; Z)$ and small alignment loss.

3.2 Theoretical analysis of the alignment loss

The empirical observations in Fig. 2 suggest that lower alignment loss correlates with lower worst-group error. Next, we show that this connection applies much more generally. We show that the maximum of $\hat{\mathcal{L}}_{\text{align}}(f_{\text{enc}}; g, g')$, over any two groups g, g' within the same class, can be used to upper bound the gap between the worst-group loss and average loss for that class. We set up several notations before stating the result. For any class label $y \in \mathcal{Y}$, let $\mathcal{G}_y$ be the set of groups with label y in $\mathcal{G}$. Define the worst-group loss among groups in $\mathcal{G}_y$ as $\mathcal{L}_{\text{wg}}(f_\theta; y) := \max_{g\in\mathcal{G}_y} \mathbb{E}_{(x,y,a)\sim P_g}\left[\ell(f_\theta(x), y)\right]$ and the average loss among groups in $\mathcal{G}_y$ as $\mathcal{L}_{\text{avg}}(f_\theta; y) := \mathbb{E}_{(x,y,a)\sim P:\forall a\in\mathcal{A}}\left[\ell(f_\theta(x), y)\right]$. Additionally, let $\mathcal{L}_{\text{align}}(f_{\text{enc}}; y)$ be the class-specific alignment loss for groups in $\mathcal{G}_y$. Finally let $\mathcal{L}_{\text{align}}(f_{\text{enc}}; y)$ be the largest cross-group alignment loss among groups in $\mathcal{G}_y$:

$$\hat{\mathcal{L}}_{\text{align}}(f_\theta; y) := \max_{g\in\mathcal{G}_y, g'\in\mathcal{G}_y:\, g\neq g'} \hat{\mathcal{L}}_{\text{align}}(f_{\text{enc}}; g, g'). \tag{3}$$

Our main result is that $\hat{\mathcal{L}}_{\text{align}}(f_\theta; y)$ is an upper bound on the gap between $\mathcal{L}_{\text{wg}}(f_\theta; y)$ and $\mathcal{L}_{\text{avg}}(f_\theta; y)$ (up to a norm multiplier and a concentration error), for any $y \in \mathcal{Y}$.

Theorem 3.1 (Alignment loss upper bounds the gap between worst-group and average-group loss)**.** *In the setting described above, let f_θ be any neural network satisfying that the weight matrix of the linear classification layer W in f_{cls} satisfies that $\|W\|_2 \leq B$, for some constant B. Let n_g be the size of any group $g \in \mathcal{G}$ in the training data set. Assume that the loss function $\ell(x, y)$ is 1-Lipschitz in x and bounded from above by one. Then, with probability at least $1 - \delta$ over the randomness of the training data set samples, for any class $y \in \mathcal{Y}$, the following holds:*

$$\mathcal{L}_{wg}(f_\theta; y) \leq \mathcal{L}_{avg}(f_\theta; y) + B \cdot \hat{\mathcal{L}}_{align}(f_\theta; y) + \max_{g\in\mathcal{G}_y} \sqrt{\frac{8\log(|\mathcal{G}_y|/\delta)}{n_g}}. \tag{4}$$

The proof of Theorem 3.1 is deferred to Sec. A. Since we also know that $\mathcal{L}_{\text{avg}}(f_\theta; y) \leq \mathcal{L}_{\text{wg}}(f_\theta; y)$, the result implies that in order to reduce the gap between the worst-group loss and the average loss for class y, it suffices to reduce the alignment loss $\hat{\mathcal{L}}_{\text{align}}(f_\theta; y)$.

4 Correct-N-Contrast (CNC)

CNC involves two stages to improve robustness to spurious correlations. In **Stage 1: ERM training**, we train an ERM model $f_{\hat{\theta}}$ on the training dataset $\{(x_i, y_i)\}_{i=1}^n$ and save predictions $\{\hat{y}_i\}_{i=1}^n$. While this model is trained to predict the ground-truth class labels, we reason that because it is susceptible to learning spurious correlations, its predictions will actually be good proxies for the spurious attributes of the datapoints. We then use these inferred spurious labels on the training data for the next stage (regularizing $f_{\hat{\theta}}$ to prevent memorization). In **Stage 2: Contrastive learning**, we train a robust model with supervised contrastive learning, using $\{\hat{y}_i\}_{i=1}^n$ and two components:

Contrastive batch sampling. We sample points such that by maximizing the similarity between anchors and positives (and keeping anchors and negatives apart), the Stage 2 model "ignores" spurious correlations. With prediction set $\{\hat{y}_i\}_{i=1}^n$, for each batch we randomly sample an anchor $x_i \in X$ (with label y_i and ERM prediction $\hat{y}_i$), M positives with the same class as y_i but different ERM prediction than $\hat{y}_i$, and N negatives with different classes as y_i but the same ERM prediction as $\hat{y}_i$.

Optimization objective and updating procedure. We wish to learn aligned representations via contrastive learning while also classifying datapoints correctly. As we have the training class labels, we jointly update both the model's encoder layers f_{enc} with a standard contrastive loss, and the full model f_θ with a cross-entropy loss (with $\lambda \in [0, 1]$ as a balancing hyperparameter):

$$\hat{\mathcal{L}}(f_\theta; x, y) = \lambda \hat{\mathcal{L}}_{\text{con}}^{\text{sup}}(f_{\text{enc}}; x, y) + (1 - \lambda)\hat{\mathcal{L}}_{\text{cross}}(f_\theta; x, y). \tag{5}$$

We summarize CNC in Algorithm 1, and include further implementation details in Appendix B.

Algorithm 1 Correct-N-Contrast (CNC)

```
Input: Training data set $(X, Y)$; # positives $M$; # negatives $N$; learning rate $\eta$, # epochs $K$.
   Stage 1: ERM Training
1: Train a regularized ERM model $f_{\hat{\theta}}$ on $(X, Y)$; save the predictions $\hat{y}_i := f_{\hat{\theta}}(x_i)$.
   Stage 2: Contrastive learning
2: for each epoch $1, \ldots, K$ do
3:   for each anchor $(x, y) \in (X, Y)$ do
4:     Let $\hat{y}$ be the predicted (group) label of $x$ from Stage 1's ERM model.
5:     Get $M$ positives $\{(x_m^+, y_m^+)\}$ where $y_m^+ = y$ but $\underline{\hat{y}_m^+ \neq \hat{y}}$, for $m = 1, \ldots, M$.
6:     Get $N$ negatives $\{(x_q^-, y_q^-)\}$ where $y_q^- \neq y$ but $\underline{\hat{y}_q^- = \hat{y}}$, for $q = 1, \ldots, N$.
7:     Update $f_\theta$ by $\theta \leftarrow \theta - \eta \cdot \nabla \hat{\mathcal{L}}(f_\theta; x, y)$ (cf. Eq. (5)) with anchor, $M$ positives, and $N$ negatives.
   return final model $f_\theta$ from Stage 2, and throw away the ERM model from Stage 1.
```

5 Experimental results

We conduct experiments to answer the following questions: (1) Does CNC improve worst-group performance over prior state-of-the-art methods on spurious correlation benchmarks? (2) Does CNC encourage learning network representations with greater alignment and class-label-only dependence?

5.1 CNC improves worst-group performance

We evaluate CNC on image classification and NLP datasets with spurious correlations (Table 1). We compare against ERM, GDRO, and comparable methods to tackle spurious correlations without training group labels: CVaR DRO [24], GEORGE [40], Learning from Failure (LfF) [30], Environment Inference for Invariant Learning (EIIL) [12], Contrastive Input Morphing (CIM) [42], and Just Train Twice (JTT) [25] (Table 1). CNC achieves highest worst-group accuracy among methods without training group labels on the CMNIST*, Waterbirds and CelebA datasets, and near-SOTA worst-group accuracy on CivilComments.

While LfF, GEORGE, EIIL, and JTT similarly use a trained ERM model to estimate groups, CNC uniquely uses ERM predictions to encourage the robust model to learn desirable representations via contrastive learning. We reason that with this approach, by sampling positives and negatives from the ERM predictions, CNC more directly encourages the robust model to ignore learnable spurious correlations compared to previous invariant learning, input transformation, or upweighting approaches. We include additional evidence of this via GradCAM visualizations in Appendix F.6.

Table 1: Worst-group and average accuracies, averaged over three seeds. On image data sets, CNC obtains significantly higher worst-group accuracy than comparable methods without group labels, competing with GDRO. CNC also competes with SoTA on CivilComments-WILDS. Implementation details are in Appendix D.

Method	CMNIST*		Waterbirds		CelebA		CivilComments-WILDS	
Accuracy (%)	Worst-group	Avg.	Worst-group	Avg.	Worst-group	Avg.	Worst-group	Avg.
ERM	0.0	20.1	72.6	97.3	47.2	95.6	57.4	92.6
CVaR DRO	22.1	61.3	75.9	96.0	64.4	82.5	60.5	92.5
LfF	0.0	25.0	78.0	91.2	77.2	85.1	58.8	92.5
GEORGE	76.4	89.5	83.8	90.5	54.9	94.5	-	-
CIM	0.0	36.8	77.2	95.6	83.6	90.6	N/A	N/A
EIIL	72.8	90.7	78.7	96.9	81.7	85.7	67.0	90.5
JTT	74.5	90.2	86.7	93.3	81.1	88.0	**69.3**	91.1
CNC (Ours)	**77.4**	90.9	**89.7**	90.8	**88.8**	89.9	69.2	82.1
GDRO	78.5	90.6	91.1	92.4	88.9	93.9	69.9	88.9

5.2 CNC learns representations less reliant on spurious features

To shed light on CNC's worst-group accuracy gains, we investigate if models trained with CNC actually learn representations with higher alignment. Compared to ERM and JTT (the next-best performing method that does not require subgroup labels), CNC learns representations with significantly higher alignment (lower alignment loss) and lower mutual information with spurious attributes (while having comparable mutual information with class labels) (Fig. 3).

We find that CNC representations exhibit the lowest alignment loss consistently for these data sets; this also corresponds to CNC models achieving the highest worst-group accuracy. Furthermore, while all methods result in representations that exhibit high mutual information with the class label (Fig. 3b), only CNC results in representations that drastically reduce mutual information with spurious attributes (Fig. 3c). In Fig. 4, we also illustrate this result on the Waterbirds data set via UMAP visualizations of the learned representations. Notably, all training methods result in representations separable by class label. Yet ERM models exhibit strong separability by spurious attributes, and JTT models interestingly also still depict some learned dependency on the spurious attribute. However, CNC uniquely learns representations that strongly depict class-label-only dependence.

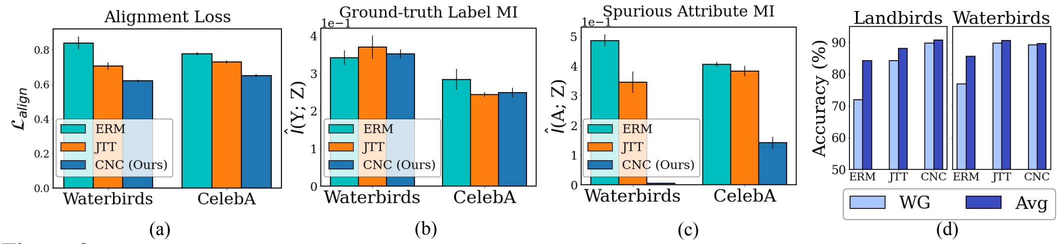


Figure 3: CNC most effectively removes dependence on the spurious attribute (abc), and obtains smaller gaps for per-class worst-group vs. average error (d), as supported by Theorem 3.1.

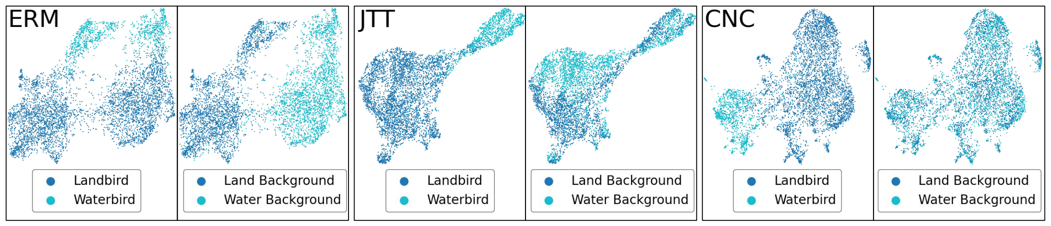


Figure 4: UMAPs of Waterbirds representations, colored by class (left) and spurious attribute (right). ERM depends on both classes $\mathcal{Y}$ and spurious attributes $\mathcal{A}$, though with greater separability for the latter. JTT representations depend more on $\mathcal{Y}$, but also on $\mathcal{A}$. CNC gets closer to fully removing the dependence on $\mathcal{A}$.

6 Conclusion

We present CNC, a two-stage contrastive approach that learn representations robust to spurious correlations. We theoretically analyze the connection between alignment and worst-group vs. average-group losses. We show that CNC improves the robustness of learned representations by making them more class-dependent and less spurious-attribute-dependent, and that CNC achieves SOTA or near-SOTA worst-group performance across several benchmark datasets.

References

[1] Kartik Ahuja, Karthikeyan Shanmugam, Kush Varshney, and Amit Dhurandhar. Invariant risk minimization games. In *International Conference on Machine Learning*, pages 145–155. PMLR, 2020.

[2] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

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

[4] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[5] Ananth Balashankar, Alyssa Lees, Chris Welty, and Lakshminarayanan Subramanian. What is fair? exploring Pareto-efficiency for fairness constrained classifiers. *arXiv preprint arXiv:1910.14120*, 2019.

[6] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 456–473, 2018.

[7] Aharon Ben-Tal, Dick Den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013.

[8] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion proceedings of the 2019 world wide web conference*, pages 491–500, 2019.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[10] Ching-Yao Chuang, J. Robinson, Yen-Chen Lin, A. Torralba, and S. Jegelka. Debiased contrastive learning. In *Advances in Neural Information Processing Systems*, volume abs/2007.00224, 2020.

[11] Remi Tachet des Combes, Mohammad Pezeshki, Samira Shabanian, Aaron Courville, and Yoshua Bengio. On the learning dynamics of deep neural networks. *arXiv preprint arXiv:1809.06848*, 2018.

[12] Elliot Creager, Jörn-Henrik Jacobsen, and Richard Zemel. Environment inference for invariant learning. In *International Conference on Machine Learning*, pages 2189–2200. PMLR, 2021.

[13] Sebastian Curi, Kfir Y. Levy, Stefanie Jegelka, and Andreas Krause. Adaptive sampling for stochastic risk-averse learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1036–1047, 2020.

[14] John Duchi and Hongseok Namkoong. Variance-based regularization with convex objectives. *The Journal of Machine Learning Research*, 20(1):2450–2504, 2019.

[15] Karan Goel, Albert Gu, Yixuan Li, and Christopher Ré. Model patching: Closing the subgroup performance gap with data augmentation. In *International Conference on Learning Representations*, 2020.

[16] Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. Supervised contrastive learning for pre-trained language model fine-tuning. In *International Conference on Learning Representations*, 2021.

[17] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

[18] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR, 2020.

[19] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4):1–21, 2012.

[20] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673, 2020.

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

[22] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). *arXiv preprint arXiv:2003.00688*, 2020.

[23] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

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

[25] Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*, pages 6781–6792. PMLR, 2021.

[26] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[27] Natalia Martinez, Martin Bertran, and Guillermo Sapiro. Minimax pareto fairness: A multi objective perspective. In *International Conference on Machine Learning (ICML)*, 2020.

[28] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *NIPS*, 2013.

[29] Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. Understanding the failure modes of out-of-distribution generalization. *arXiv preprint arXiv:2010.15775*, 2020.

[30] Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: De-biasing classifier from biased classifier. In *Advances in Neural Information Processing Systems*, volume 33, pages 20673–20684, 2020.

[31] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[32] Yonatan Oren, Shiori Sagawa, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust language modeling. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[33] Giambattista Parascandolo, Alexander Neitz, Antonio Orvieto, Luigi Gresele, and Bernhard Schölkopf. Learning explanations that are hard to vary. *arXiv preprint arXiv:2009.00329*, 2020.

[34] Mohammad Pezeshki, Sékou-Oumar Kaba, Yoshua Bengio, Aaron Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. *arXiv preprint arXiv:2011.09468*, 2020.

[35] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “why should i trust you?” explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[36] J. Robinson, Ching-Yao Chuang, S. Sra, and S. Jegelka. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021.

[37] 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. In *International Conference on Learning Representations*, 2019.

[38] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[39] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, S. Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141, 2018.

[40] Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. In *Advances in Neural Information Processing Systems*, volume 33, pages 19339–19352, 2020.

[41] Jiaming Song and Stefano Ermon. Multi-label contrastive predictive coding. In *Advances in Neural Information Processing Systems*, volume 33, pages 8161–8173, 2020.

[42] Saeid Asgari Taghanaki, Kristy Choi, Amir Khasahmadi, and Anirudh Goyal. Robust representation learning via perceptual similarity metrics. *arXiv preprint arXiv:2106.06620*, 2021.

[43] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

[44] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. *arXiv preprint arXiv:2005.10243*, 2020.

[45] Julius von Kügelgen, Yash Sharma, Luigi Gresele, Wieland Brendel, Bernhard Schölkopf, Michel Besserve, and Francesco Locatello. Self-supervised learning with data augmentations provably isolates content from style. *arXiv preprint arXiv:2106.04619*, 2021.

[46] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[47] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

[48] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[49] Wolfram Wiesemann, Daniel Kuhn, and Melvyn Sim. Distributionally robust convex optimization. *Operations Research*, 62(6):1358–1376, 2014.

[50] M. Wu, M. Mosse, Chengxu Zhuang, D. Yamins, and Noah D. Goodman. Conditional negative sampling for contrastive learning of visual representations. In *International Conference on Learning Representations*, 2021.

[51] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[52] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

[53] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Our contributions and claims are outlined in the introduction, and reflected in Sections 3.1, 3.2, and 5.
   (b) Did you describe the limitations of your work? [Yes] See Appendix E.1
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix E.2
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Sec 3.2
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See link in Appendix G
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix D
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Appendix F.5
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Hardware resource details in Appendix D.3
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] Licenses for code and datasets mentioned in the appendix where applicable.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Code linked in Appendix G
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Omitted Proofs from Section 3.2

In this section, we prove that within any class, the gap between the worst-group error and the average error can be upper bounded by the alignment loss times the Lipschitz constant, plus another concentration error term.

Proof of Theorem 3.1. Consider two arbitrary groups, denoted by $g_1 = (y, a_1)$ and $g_2 = (y, a_2)$, whose class labels are both $y \in \mathcal{Y}$, whose spurious attributes are $a_1 \in \mathcal{A}$ and $a_2 \in \mathcal{A}$ such that $a_1 \neq a_2$. Let G_1 and G_2 be the subset of training data that belong to groups g_1 and g_2, respectively. We note that both G_1 and G_2 are non-empty since we have assumed that (in Section 2) there is at least one sample from each group in the training data set. Let $n_{g_1} = |G_1|$ and $n_{g_2} = |G_2|$ be the size of these two groups, respectively. Recall that f_{enc} denotes the mapping of the encoder layers of the full neural network model f_θ. Since the classification layer f_{cls} is a linear layer, we have used W to denote the weight matrix of this layer. Our definition of the cross-group alignment loss in equation (3), denoted as $\hat{\mathcal{L}}_{\text{align}}(f_\theta; y)$, implies that for g_1 and g_2,

$$\frac{1}{n_{g_1}} \frac{1}{n_{g_2}} \sum_{(x,y,a_1)\in G_1} \sum_{(x',y,a_2)\in G_2} \|f_{\text{enc}}(x) - f_{\text{enc}}(x')\|_2 \leq \hat{\mathcal{L}}_{\text{align}}(f_\theta; y). \tag{6}$$

Next, let $\mathbb{E}_{(x,y,a_1)\sim\mathcal{P}_{g_1}} [\mathcal{L}_{\text{avg}}(W f_{\text{enc}}(x), y)]$ be the average loss conditioning on a data point being sampled from group g_1 (and similarly for group g_2). Let $\Delta(g_1, g_2)$ be the difference between the population average losses:

$$\Delta(g_1, g_2) = \left| \mathop{\mathbb{E}}_{(x,y,a_1)\sim\mathcal{P}_{g_1}} [\mathcal{L}_{\text{avg}}(W f_{\text{enc}}(x), y] - \mathop{\mathbb{E}}_{(x,y,a_2)\sim\mathcal{P}_{g_2}} [\mathcal{L}_{\text{avg}}(W f_{\text{enc}}(x), y)] \right|.$$

Recall that $\mathcal{G}_y \subseteq \mathcal{G}$ is the set of groups that have class label y. Since the loss $\ell(\cdot)$ is bounded above by 1 according to our assumption, and is at least zero, by the Hoeffding's inequality, the following result holds with probability at least $1 - \delta$, for all $|\mathcal{G}_y|$ groups $g \in \mathcal{G}_y$,

$$\left| \mathop{\mathbb{E}}_{(x,y,a)\sim\mathcal{P}_g} [\mathcal{L}_{\text{avg}}(W f_{\text{enc}}(x), y)] - \frac{1}{n_g} \sum_{(x,y)\in(X,Y)} \ell(W f_{\text{enc}}(x), y) \right| \leq \sqrt{\frac{2\log\left(|\mathcal{G}_y|/\delta\right)}{n_g}}. \tag{7}$$

Thus, with probability at least $1 - \delta$, the following holds for any g_1 and g_2 in class y (but having different spurious attributes)

$$\Delta(g_1, g_2) \leq \left| \frac{1}{n_{g_1}} \sum_{(x,y,a_1)\in G_1} \mathcal{L}_{\text{avg}}(W f_{\text{enc}}(x), y) - \frac{1}{n_{g_2}} \sum_{(x',y,a_2)\in G_2} \mathcal{L}_{\text{avg}}(W f_{\text{enc}}(x'), y) \right| \tag{8}$$
$$+ \left(\sqrt{\frac{2\log(|\mathcal{G}_y|/\delta)}{n_{g_1}}} + \sqrt{\frac{2\log(|\mathcal{G}_y|/\delta)}{n_{g_2}}} \right).$$

Next, we focus on the RHS of equation (8). First, equation (8) is also equal to the following:

$$\left| \frac{1}{n_{g_1}} \frac{1}{n_{g_2}} \sum_{(x,y,a_1)\in G_1} \sum_{(x',y,a_2)\in G_2} \ell(W f_{\text{enc}}(x), y)) - \frac{1}{n_{g_1}} \frac{1}{n_{g_2}} \sum_{(x,y,a_1)\in G_1} \sum_{(x',y,a_2)\in G_2} \ell(W f_{\text{enc}}(x'), y)) \right|.$$

Since we have also assumed that the loss function $\ell(x, y)$ is 1-Lipschitz in x[1], the above is at most:

$$\begin{aligned}
&\left| \frac{1}{n_{g_1} n_{g_2}} \sum_{(x,y,a_1)\in G_1} \sum_{(x',y,a_2)\in G_2} |\ell(W f_{\text{enc}}(x), y) - \ell(W f_{\text{enc}}(x'), y)| \right| \\
\leq &\frac{1}{n_{g_1} n_{g_2}} \sum_{(x,y,a_1)\in G_1} \sum_{(x',y,a_2)\in G_2} \|W f_{\text{enc}}(x) - W f_{\text{enc}}(x')\|_2 \qquad \text{(since } y \text{ is the same for } x, x'\text{)} \\
\leq &\frac{B}{n_{g_1} n_{g_2}} \sum_{(x,y,a_1)\in G_1} \sum_{(x',y,a_2)\in G_2} \|f_{\text{enc}}(x) - f_{\text{enc}}(x')\|_2 \qquad \text{(because } \|W\|_2 \leq B \text{ as assumed)} \\
\leq &B \cdot \hat{\mathcal{L}}_{\text{align}}(f_\theta; y). \qquad \text{(because of equation (6))}
\end{aligned}$$

[1] In other words, we assume that $|\ell(z, y) - \ell(z', y)| \leq \|z - z'\|_2$, for any z, z' and y.

Thus, we have shown that for any g_1 and g_2 within class y,

$$\Delta(g_1, g_2) \le B \cdot \hat{\mathcal{L}}_{\text{align}}(f_\theta; y) + \left(\sqrt{\frac{2\log(|\mathcal{G}_y|/\delta)}{n_{g_1}}} + \sqrt{\frac{2\log(|\mathcal{G}_y|/\delta)}{n_{g_2}}}\right)$$
$$\le B \cdot \hat{\mathcal{L}}_{\text{align}}(f_\theta; y) + \max_{g\in\mathcal{G}_y}\sqrt{\frac{8\log(|\mathcal{G}_y|/\delta)}{n_g}}. \tag{9}$$

Finally, we use the above result to bound the gap between the worst-group loss and the average loss. For every group $g \in \mathcal{G}$, let p_g denote the prior probability of observing a sample from $\mathcal{P}$ in this group. Let $q_y = \sum_{g'\in\mathcal{G}_y} p_{g'}$. Let $h(g)$ be a short hand notation for

$$h(g) = \mathop{\mathbb{E}}_{(x,y,a)\sim\mathcal{P}_g}\left[\mathcal{L}_{\text{avg}}(W f_{\text{enc}}(x), y)\right].$$

The average loss among the groups with class label y is $\mathcal{L}_{\text{avg}}(f_\theta; y) = \sum_{g\in\mathcal{G}_y} \frac{p_g}{q_y} h(g)$. The worst-group loss among the groups with class label y is $\mathcal{L}_{\text{wg}}(f_\theta; y) = \max_{g\in\mathcal{G}_y} h(g)$. Let $g^\star$ be a group that incurs the highest loss among groups in $\mathcal{G}_y$. We have $\mathcal{L}_{\text{wg}}(f_\theta; y) - \mathcal{L}_{\text{avg}}(f_\theta; y)$ is equal to

$$h(g^\star) - \sum_{g\in\mathcal{G}_y}\frac{p_g}{q_y}h(g) = \sum_{g\in\mathcal{G}_y}\frac{p_g}{q_y}\left(h(g^\star) - h(g)\right) \tag{10}$$
$$\le \sum_{g\in\mathcal{G}_y}\frac{p_g}{q_y}\Delta(g^\star, g) \tag{11}$$
$$\le B \cdot \hat{\mathcal{L}}_{\text{align}}(f_\theta; y) + \max_{g\in\mathcal{G}_y}\sqrt{\frac{8\log(|\mathcal{G}|/\delta)}{n_g}}. \tag{12}$$

The last step uses equation (9) on $\Delta(g^\star, g)$ and the fact that $q_y = \sum_{g'\in\mathcal{G}_y} p_{g'}$. Thus, we have shown that the gap between the worst-group loss and the average loss among the groups with the same class label is bounded by the above equation. The proof is now complete. $\square$

As a remark, the proof of Theorem 3.1 can be adapted to the case when the loss function $\ell(\cdot)$ is Lipschitz for some fixed constant and when the loss function is bounded from above by some fixed constant as well. We leave the details to the reader.

The astute reader will note that Theorem 3.1 focuses on comparing groups within the same class y, for any $y \in \mathcal{Y}$. A natural follow-up question is what happens when comparing across groups with different labels. Let $\mathcal{L}_{\text{wg}}(f_\theta) = \max_{y\in\mathcal{Y}} \mathcal{L}_{\text{wg}}(f_\theta; y)$ be the worst-group loss across all the labels. Recall that $\mathcal{L}_{\text{avg}}(f_\theta)$ is the average loss for the entire population of data. We generalize Theorem 3.1 to this setting in the following result.

Corollary A.1 (Extension of Theorem 3.1 to compare across different classes)**.** *In the setting of Theorem 3.1, let $q_y = \sum_{g\in\mathcal{G}_y} p_g$ be the prior probability of observing a sample drawn from $\mathcal{P}$ with label y, for any $y \in \mathcal{Y}$. We have that with probability at least $1-\delta$, the following holds:*

$$\mathcal{L}_{wg}(f_\theta) \le \left(\min_{y\in\mathcal{Y}} q_y\right)^{-1} \mathcal{L}_{avg}(f_\theta) + B \cdot \max_{y\in\mathcal{Y}} \hat{\mathcal{L}}_{align}(f_\theta; y) + \max_{g\in\mathcal{G}}\sqrt{\frac{8\log(|\mathcal{G}|/\delta)}{n_g}}. \tag{13}$$

Proof. We generalize the argument in the previous result to compare across different labels. The worst-group loss across different labels is

$$\max_{y\in\mathcal{Y}}\max_{g\in\mathcal{G}_y} h(g)$$
$$\le \max_{y\in\mathcal{Y}}\left(\sum_{g\in\mathcal{G}_y}\frac{p_g}{q_y}h(g) + B\cdot\hat{\mathcal{L}}_{\text{align}}(f_\theta; y) + \max_{g\in\mathcal{G}_y}\sqrt{\frac{8\log(|\mathcal{G}_y|/\delta)}{n_g}}\right) \quad \text{(because of equation (12))}$$
$$\le \frac{1}{\min_{y\in\mathcal{Y}} q_y}\sum_{g\in\mathcal{G}_y} p_g h(g) + B\cdot\max_{y\in\mathcal{Y}}\hat{\mathcal{L}}_{\text{align}}(f_\theta; y) + \max_{g\in\mathcal{G}}\sqrt{\frac{8\log(|\mathcal{G}|/\delta)}{n_g}}.$$

Since $\sum_{g\in\mathcal{G}} p_g h(g) = \mathcal{L}_{\text{avg}}(f_\theta)$, we thus conclude that

$$\mathcal{L}_{\text{wg}}(f_\theta) \leq \left(\min_{y\in\mathcal{Y}} q_y\right)^{-1} \mathcal{L}_{\text{avg}}(f_\theta) + B \cdot \max_{y\in\mathcal{Y}} \hat{\mathcal{L}}_{\text{align}}(f_\theta; y) + \max_{g\in\mathcal{G}} \sqrt{\frac{8\log(|\mathcal{G}|/\delta)}{n_g}}.$$

The proof is now complete. □

An example showing that Corollary A.1 is tight. We describe a simple example in which the factor $\left(\min_{y\in\mathcal{Y}} q_y\right)^{-1}$ in equation (13) is tight (asymptotically). Suppose there are k perfectly balanced classes so that $q_y = 1/k$, for every $y \in \mathcal{Y}$. There is one data point from each class, with loss equal to 0 for all except one of them. The worst-group loss is 1 whereas the average loss is $1/k$. Thus, there is a factor of k between the worst-group loss and the average loss. For equation (13), the factor

$$\left(\min_{y\in\mathcal{Y}} q_y\right)^{-1} = k,$$

since $q_y = 1/k$ for every $y \in \mathcal{Y}$ in this example. Thus, this factor matches the (multiplicative) factor between the worst-group loss and the average loss in this example.

B Contrastive algorithm design details

In this section, we provide further details on the training setup and contrastive batch sampling, pseudocode, and additional properties related to CNC's implementation.

B.1 Algorithm

Algorithm 2 Correct-N-Contrast (CNC)

```
Input: Training dataset $(X, Y)$; # positives $M$; # negatives $N$; learning rate $\eta$, # epochs $K$.
   Stage 1: ERM Training
1: Train a regularized ERM model $f_{\hat{\theta}}$ on $(X, Y)$; save the predictions $\hat{y}_i := f_{\hat{\theta}}(x_i)$.
   Stage 2: Supervised contrastive learning
2: for each epoch $1, \ldots, K$ do
3:   for each anchor $(x, y) \in (X, Y)$ do
4:     Let $\hat{y}$ be the predicted (group) label of $x$ from Stage 1's ERM model.
5:     Get $M$ positives $\{(x_m^+, y_m^+)\}$ where $y_m^+ = y$ but $\underline{\hat{y}_m^+ \neq \hat{y}}$, for $m = 1, \ldots, M$.
6:     Get $N$ negatives $\{(x_q^-, y_q^-)\}$ where $y_q^- \neq y$ but $\underline{\hat{y}_q^- = \hat{y}}$, for $q = 1, \ldots, N$.
7:     Update $f_\theta$ by $\theta \leftarrow \theta - \eta \cdot \nabla\hat{\mathcal{L}}(f_\theta; x, y)$ (cf. Eq. (5)) with anchor, $M$ positives, and $N$ negatives.
   return final model $f_\theta$ from Stage 2, and throw away the ERM model from Stage 1.
```

B.2 Training setup

In Fig. 5, we illustrate the two training stages of Correct-N-Contrast described in Sec. 4. In Stage 1, we first train an ERM model with a cross-entropy loss. For consistency with Stage 2, we depict the output as a composition of the encoder and linear classifier layers. Then in Stage 2, we train a new model with the same architecture using contrastive batches sampled with the Stage 1 ERM model and a supervised contrastive loss (1) (which we compute after the depicted representations are first normalized) to update the encoder layers. Note that unlike prior work in contrastive learning [9, 20], as we have the class labels of the anchors, positives, and negatives, we also continue forward-passing the unnormalized representations (encoder layer outputs) and compute a cross-entropy loss to update the classifier layers while jointly training the encoder layers as well.

We also note that unlike prior work, we wish to learn invariances between anchors and positives that maximally reduce the presence of features not needed for classification. We thus do not pass the representations through an additional *projection network* [9]. Instead, we use Eq. 1 to compute the supervised contrastive loss directly on the encoder outputs $z = f_{\text{enc}}(x)$. In Appendix **??**, we study ablations with both design choices.

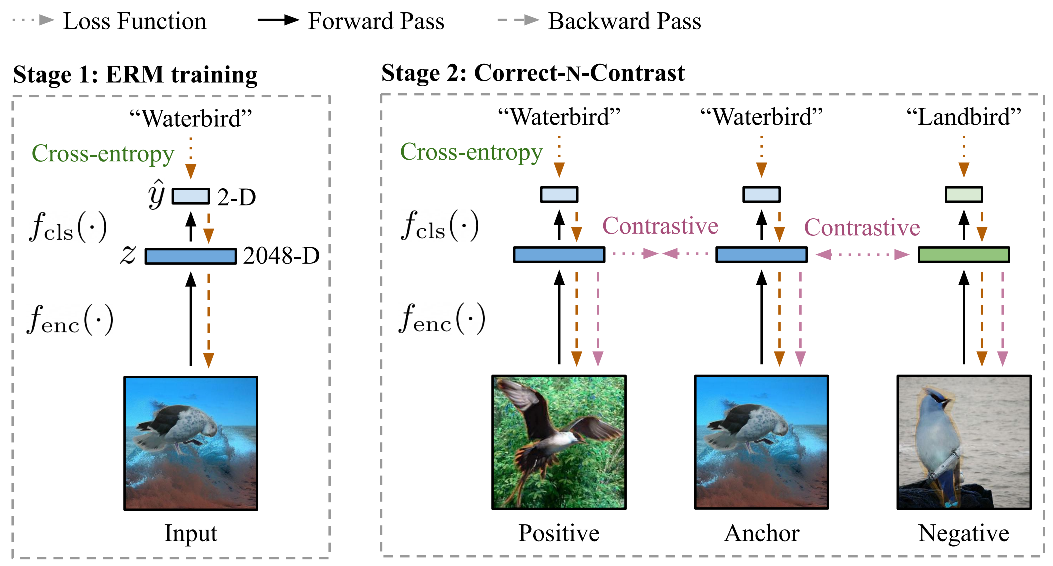


Figure 5: The two stages of Correct-N-Contrast. In Stage 1, we train a model with standard ERM and a cross-entropy loss. Then in Stage 2, we train a new model with the same architecture, but specifically learn spurious-attribute-invariant representations with a contrastive loss (1) and batches of anchors, positives, and negatives sampled with the ERM model's predictions. We also update the full model jointly with a cross-entropy loss on the classifier layer output and the input class labels. Dimensions for ResNet-50 and Waterbirds.

B.3 Two-sided contrastive batch implementation

We provide more implementation details on our default contrastive batch sampling approach described in Sec. 4. For additional contrastive signal per batch, we can double the pairwise comparisons in a training batch by switching the anchor and positive roles. This is similar to the *NT-Xent* loss in prior contrastive learning work [9]. We switch the role of the anchor and first positive sampled in a contrastive batch, and sample additional positives and negatives using the same guidelines but adjusting for the "new" anchor. We denote this as "two-sided" sampling in contrast with the "one-sided" comparisons we get with just the original anchor, positives, and negatives.

Implementing this sampling procedure in practice is simple. First, recall our initial setup with trained ERM model $f_{\hat{\theta}}$, its predictions $\{\hat{y}_i\}_{i=1}^n$ on training data $\{(x_i, y_i)\}_{i=1}^n$ (where $\hat{y}_i = f_{\hat{\theta}}(x_i)$), and number of positives and negatives to sample M and N. We then sample batches with Algorithm 3.

Algorithm 3 Sampling two-sided contrastive batches

```
Require: Number of positives $M$ and number of negatives $N$ to sample for each batch.
1: Initialize set of contrastive batches $B = \{\}$
2: for each $x_i \in \{x_i \in X : \hat{y}_i = y_i\}$ do
3:     Sample $M-1$ additional "anchors" to obtain $\{x_i\}_{i=1}^M$ from $\{x_i \in X : \hat{y}_i = y_i\}$
4:     Sample $M$ positives $\{x_m^+\}_{m=1}^M$ from $\{x_m^- \in X : \hat{y}_m^- = \hat{y}_i,\ y_m^- \neq y_i\}$
5:     Sample $N$ negatives $\{x_n^-\}_{n=1}^N$ from $\{x_n^- \in X : \hat{y}_n^- = \hat{y}_i,\ y_n^- \neq y_i\}$
6:     Sample $N$ negatives $\{x_n'^-\}_{n=1}^N$ from $\{x_n'^- \in X : \hat{y}_n'^- = \hat{y}_1^+, y_n'^- \neq y_1^+\}$
7:     Update contrastive batch set: $B \leftarrow B \cup \left(\{x_i\}_{i=1}^M, \{x_m^+\}_{m=1}^M, \{x_n^-\}_{n=1}^N, \{x_n'^-\}_{n=1}^N\right)$
```

Because the initial anchors are then datapoints that the ERM model gets correct, under our heuristic we infer $\{x_i\}_{i=1}^M$ as samples from the majority group. Similarly the M positives $\{x_m^+\}_{m=1}^M$ and N negatives $\{x_n^-\}_{n=1}^N$ that it gets incorrect are inferred to belong to minority groups.

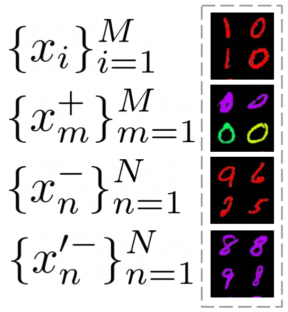
For one batch, we then compute the full contrastive loss with

$$\hat{\mathcal{L}}^{\text{sup}}_{\text{con}}(f_{\text{enc}}) = \hat{\mathcal{L}}^{\text{sup}}_{\text{con}}\left(x_1, \{x^+_m\}^M_{m=1}, \{x^-_n\}^N_{n=1}; f_{\text{enc}}\right) + \hat{\mathcal{L}}^{\text{sup}}_{\text{con}}\left(x^+_1, \{x_i\}^M_{i=1}, \{x'^-_n\}^N_{n=1}; f_{\text{enc}}\right) \tag{14}$$

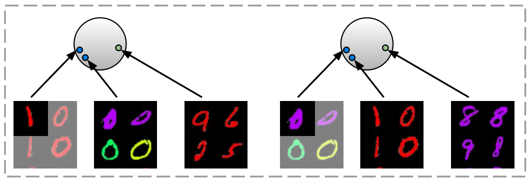
where e.g. $\hat{\mathcal{L}}^{\text{sup}}_{\text{con}}\left(x_1, \{x^+_m\}^M_{m=1}, \{x^-_n\}^N_{n=1}; f_{\text{enc}}\right)$ is given by:

$$-\frac{1}{M}\sum_{m=1}^{M}\log\frac{\exp(z_1^\top z^+_m/\tau)}{\sum_{m=1}^{M}\exp(z_1^\top z^+_m/\tau) + \sum_{n=1}^{N}\exp(z_1^\top z^+_n/\tau)} \tag{15}$$

and again let z be the normalized output $f_{\text{enc}}(x)$ for corresponding x. We compute the cross-entropy component of the full loss for each x in the two-sided batch with its corresponding label y.



(a) Contrastive Batch



(b) Two-sided Update

Figure 6: Illustration of two-sided contrastive batch sampling with Colored MNIST as an example. From a single batch (a), we can train a contrastive model with two anchor-positive-negative pairings (b). Aside from increasing the total number of "hard negatives" considered for each anchor-positive pair, this intuitively "pushes" together anchors and positives from two different directions for greater class separation.

B.4 Summary of CNC design choices and properties

We summarize CNC's design choices, additional properties, and differences from standard supervised contrastive learning below. In Appendix F.4, we empirically validate each component.

No projection network. As we wish to learn data representations that maximize the alignment between anchor and positive datapoints, we do not compute the contrastive loss with the outputs of an additional nonlinear projection network. This is inspired by the logic justifying a projection head in prior contrastive learning, e.g. SimCLR [9], where the head is included because the contrastive loss trains representations to be "invariant to data transformation" and may encourage removing information "such as the color or orientation of objects". In our case, we view inferred datapoints with the same class but different spurious attributes as "transformations" of each other, and we hypothesize that maximally removing these differences can help us improve worst-group performance.

Two-sided contrastive sampling. To incorporate additional comparisons between datapoints that only differ in spurious attribute during training, we employ "two-sided" contrastive batch sampling. This lets us equally incorporate instances where the second contrastive model in CNC treats datapoints that the initial ERM model got incorrect and correct as anchors.

Additional intrinsic hard positive/negative mining. Because the new model corrects for potentially learned spurious correlations by only comparing and contrasting datapoints that differ in class label or spurious attribute, but not both (as dictated by the initial ERM model's outputs), the contrastive batches naturally carry "hard" positives and negatives. Thus, our approach provides a natural form of hard negative mining (in addition to the intrinsic hard positive / negative mining at the gradient level with InfoNCE-style contrastive losses [9, 20]) while avoiding class collisions, two nontrivial challenges in standard self-supervised contrastive learning [36, 50, 10].

C Further related work discussion

We provide additional discussion of related work and connections to our work below.

C.1 Improving robustness to spurious correlations

Our core objective is to improve model robustness to group or subpopulation distribution shifts that arise from the presence of spurious correlations, specifically for classification tasks. Because these learnable correlations hold for some but not all samples in a dataset, standard training with ERM may result in highly variable performance: a model that learns to classify datapoints based on these spurious correlations does well for some subsets or "groups" of the data but not others. To improve model robustness and avoid learning spurious correlations, prior work introduces the goal to maximize the worst-group accuracy [37]. Related works broadly fall under two categories:

Improving robustness with group information. If information such as spurious attribute labels is provided, one can divide the data into explicit groups as defined in Sec. 2, and then train to directly minimize the worst group-level error among these groups. This is done in group DRO (GDRO) [37], where the authors propose an online training algorithm that focuses training updates over datapoints from higher-loss groups. Goel et al. [15] also adopt this approach with their method CycleGAN Augmented Model Patching (CAMEL). However, similar to our motivation, they argue that a stronger modeling goal should be placed on preventing a model from learning group-specific features. Their approach involves first training a CycleCAN [53] to learn the data transformations from datapoints in one group to another that share the same class label. They then apply these transformations as data augmentations to different samples, intuitively generating new versions of the original samples that take on group-specific features. Finally they train a new model with a consistency regularization objective to learn invariant features between transformed samples and their sources. Unlike their consistency loss, we accomplish a similar objective to learn group-invariant features with contrastive learning. Our first training stage is also less expensive. Instead of training a CycleGAN and then using it to augment datapoints, we train a relatively simple standard ERM classification model, sometimes with only a few number of epochs, and use its predictions to identify pairs of datapoints to serve a similar purpose. Finally, unlike both CAMEL and GDRO, we do not require spurious attribute or group labels for each training datapoints. We can then apply CNC in less restrictive settings where such information is not known.

Related to GDRO are methods that aim to optimize a "Pareto-fair" objective, more general than simply the worst-case group performance. Notable examples are the works of Balashankar et al. [5] and Martinez et al. [27]. However, these approaches similarly do not directly optimize for good representation alignment (unlike our work).

Improving robustness without training group information. More similar to our approach are methods that do not assume group information at training time, and only require validation set spurious attribute labels for fine-tuning. As validation sets are typically much smaller in size than training sets, an advantage of CNC and comparable methods is that we can improve the accessibility of robust training methods to a wider set of problems. One popular line of work is distributionally robust optimization (DRO), which trains models to minimize the worst loss within a ball centered around the observed distribution [7, 49, 14, 24, 13, 32]. This includes the CVaR DRO [24] method we evaluate against. However, prior work has shown that these approaches may be too pessimistic, optimizing not just for worst-group accuracy but worst possible accuracy within the distribution balls [37], or too undirected, optimizing for too many subpopulations, e.g. by first upweighting minority points but then upweighting majority points in later stages of training [25]. Pezeshki et al. [34] instead suggest that *gradient starvation* (GS), where neural networks only learn to capture statistically dominant features in the data [11], is the main culprit behind learning spurious correlations, and introduce a "spectral decoupling" regularizer to alleviate GS. However this does not directly prevent models from learning dependencies on spurious attributes. Similar to CAMEL, [42] propose Contrastive Input Morphing (CIM), an image dataset-specific method that aims to learn input feature transformations that remove the effects of spurious or task-irrelevant attributes. They do so without group labels, training a transformation network with a triplet loss to transform input images such that a given transformed image's *structural similarity metric* (based on luminance, contrast, and structure [48]) is more similar to a "positive" image from the same class than a "negative" image from a different class. They then train a classifier on top of these representations. Instead of pixel-level similarity metrics, CNC enforces similarity in a neural network's hidden-layer representations, allowing CNC to apply to non-image modalities. Additionally, we sample positives and negatives not just based on class label, but also the learned spurious correlations of an ERM model (via its trained predictions). We hypothesize that our sampling scheme, which intuitively provides "harder" positive and negative examples, allows CNC to more strongly overcome spurious correlations.

Most similar to our approach are methods that first train an initial ERM model with the class labels as a way to identify data points belonging to minority groups, and subsequently train an additional model with greater emphasis on the estimated minority groups. Sohoni et al. [40] demonstrate that even when only trained on the class labels, neural networks learn feature representations that can be clustered into groups of data exhibiting different spurious attributes. They use the resulting cluster labels as estimated group labels before running GDRO on these estimated groups. Meanwhile, Nam et al. [30] train a pair of models, where one model minimizes a generalized cross-entropy loss [51], such that the datapoints this model classifies incorrectly largely correspond to those in the minority group. They then train the other model on the same data but upweight the minority-group-estimated points. While they interweave training of the biased and robust model, Liu et al. [25] instead train one model first with a shortened training time (but the standard cross-entropy objective), and show that then upsampling the incorrect data points and training another model with ERM can yield higher worst-group accuracy. [12] first train an ERM model, and then softly assign the training data into groups under which the initial trained ERM model would maximally violate the invariant risk minimization (IRM) objective. In particular, the IRM objective is maximally satisfied if a model's optimal classifier is the same across groups [3], and EIIL groups are inferred such that the initial ERM model's representations exhibit maximum variance within each group. Finally, Nagarajan et al. [29] provides a theoretical understanding of how ERM picks up spurious features under dataset imbalance. They consider a setting involve a single spurious feature that is correlated with the class label and analyze the max-margin classifier in the presence of this spurious feature.

In our work, we demonstrate that the ERM model's predictions can be leveraged to not only estimate groups and train a new model with supervised learning but with different weightings. Instead, we can specifically identify pairs of points that a contrastive model can then learn invariant features between. Our core contribution comes from rethinking the objective with a contrastive loss that more directly reduces the model's ability to learning spurious correlations.

C.2 Contrastive learning

Our method also uses contrastive learning, a simple yet powerful framework for both self-supervised [9, 31, 43, 41, 39, 18, 36] and supervised [20, 16] representation learning. The core idea is to learn data representations that maximize the similarity between a given input "anchor" and distinct different views of the same input ("positives"). Frequently this also involves *contrasting* positives with "negative" data samples without any assumed relation to the anchor [4]. Core components then include some way to source multiple views, e.g. with data transformations [9], and training objectives similar to noise contrastive estimation [17, 28].

An important component of contrastive learning is the method by which appropriate positives and negatives are gathered. For sampling positives, Chen et al. [9] show that certain data augmentations (e.g. crops and cutouts) may be more beneficial than others (e.g. Gaussian noise and Sobel filtering) when generating anchors and positives for unsupervised contrastive learning. [45] theoretically study how data augmentations help contrastive models learn core content attributes which are invariant to different observed "style changes". They propose a latent variable model for self-supervised learning. Tian et al. [44] further study what makes good views for contrastive learning. They propose an "InfoMin principle", where anchors and positives should share the least information necessary for the contrastive model to do well on the downstream task. For sampling negatives, Robinson et al. [36] show that contrastive learning also benefits from using "hard" negatives, which (1) are actually a different class from the anchor (which they approximate in the unsupervised setting) and (2) embed closest to the anchor under the encoder's current data representation. Both of these approaches capture the principle that if positives are always too similar to the anchor and negatives are always too different, then contrastive learning may be inefficient at learning generalizable representations of the underlying classes.

In our work, we incorporate this principle by sampling data points with the same class label but different ERM predictions–presumably because of spurious attribute differences–as anchor and positive views, while sampling negatives from data points with different class labels but the same ERM prediction as the anchor. The anchors and positives are different enough that a trained ERM model predicted them differently, while the anchors and negatives are similar enough that the trained ERM model predicted them the same. Contrasting the above then allows us to exploit both "hard" positive and negative criteria for our downstream classification task. In Appendix F.4.1, we show that removing this ERM-guided sampling and only sampling positives and negatives based on class information leads to substantially lower worst-group accuracy with CNC.

One limitation of our current theoretical analysis regarding the alignment loss (cf. Section 3.2) is that we require knowing the group labels to compute the RHS of equation (4) (in particular, the alignment loss). An interesting question for future work is to provide a better theoretical understanding of the alignment induced by CNC in the context of spurious correlations.

C.3 Learning invariant representations

Finally, our work is also similar in motivation to Invariant Risk Minimization (IRM) [3] and other related works in domain-invariant learning [22, 33, 1, 12]. These methods aim to train models that learn a single invariant representation that is consistently optimal (e.g. with respect to classifying data) across different domains or environments. These environments can be thought of as data groups, and while traditionally methods such as IRM require that environment labels are known, recent approaches such as Environment Inference for Invariant Learning (EIIL) [12] similarly aim to infer environments with an initial ERM model. In EIIL, they next train a more robust model with an invariant learning objective, similarly selecting models based on the worst-group error on the validation set. However, their main goal for learning these invariances is to extrapolate to out-of-domain distribution shifts not seen during training, as opposed to improving worst-group performance / robustness to group shifts, which is made challenging by rare groups as in our setting.

D Additional experimental details

We first further describe our evaluation benchmarks in Appendix D.1. We next provide further details on how we calculate the reported metrics and the experimental hyperparameters of our main results in Appendix D.1. For all methods, following prior work [25, 40, 30, 37, 12] we report the test set worst-group and average accuracies from models selected through hyperparameter tuning for the best validation set worst-group accuracy. While different methods have different numbers of tunable hyperparameters, we try to keep the number of validation queries as close as possible while tuning for fair comparison.

D.1 Dataset details

Colored MNIST (CMNIST*). We evaluate with a version of the Colored MNIST dataset proposed in Arjovsky et al. [3]. The goal is to classify MNIST digits belonging to one of 5 classes $\mathcal{Y} = \{(0, 1), (2, 3), (4, 5), (6, 7), (8, 9)\}$, and treat color as the spurious attribute. In the training data, we color p_{corr} of each class's datapoints with an associated color a, and color the rest randomly. If p_{corr} is high, trained ERM models fail to classify digits that are not the associated color. We pick a from uniformly interspersed intervals of the `hsv` colormap, e.g. 0 and 1 digits may be spurious correlated with the color red (`#ff0000`), while 8 and 9 digits may be spuriously correlated with purple (`#ff0018`). The full set of colors in class order are $\mathcal{A} = \{$`#ff0000`, `#85ff00`, `#00fff3`, `#6e00ff`, `#ff0018`$\}$. For validation and test data, we color each datapoint randomly with a color $a \in \mathcal{A}$. We use the default test set from MNIST, and allocate 80%-20% of the default MNIST training set to the training and validation sets. For main results, we set $p_{\text{corr}} = 0.995$.

Waterbirds. We evaluate with the Waterbirds dataset, which was introduced as a standard spurious correlations benchmark in Sagawa et al. [37]. In this dataset, masked out images of birds from the CUB dataset [46] are pasted on backgrounds from the Places dataset [52]. Bird images are labeled either as waterbirds or landbirds; background either depicts water or land. From CUB, waterbirds consist of seabirds (ablatross, auklet, cormorant, frigatebird, fulmar, gull, jaeger, kittiwake, pelican, puffin, tern) and waterfowl (gadwell, grebe, mallard, merganser, guillemot, Pacific loon). All other birds are landbirds. From Places, water backgrounds consist of ocean and natural lake classes, while land backgrounds consist of bamboo forest and broadleaf forest classes.

The goal is to classify the foreground bird as $\mathcal{Y} = \{\text{waterbird, landbird}\}$, where there is spurious background attribute $\mathcal{A} = \{\text{water background, land background}\}$. We use the default training, validation, and test splits [37], where in the training data 95% of waterbirds appear with water backgrounds and 95% of landbirds appear with land backgrounds. Trained ERM models then have trouble classifying waterbirds with land backgrounds and landbirds with water backgrounds. For validation and test sets, water and land backgrounds are evenly split among landbirds and waterbirds.

CelebA. We evaluate with the CelebA spurious correlations benchmark introduced in Sagawa et al. [37]. The goal is to classify celebrities' hair color $\mathcal{Y} = \{\text{blond, not blond}\}$, which is spuriously correlated with the celebrity's identified gender $\mathcal{A} = \{\text{male, female}\}$. We use the same training, validation, test splits as in Sagawa et al. [37]. Only 6% of blond celebrities are male; trained ERM models perform poorly on this group.

CivilComments-WILDS. We evaluate with the CivilComments-WILDS dataset from Koh et al. [21], derived from the Jigsaw dataset from Borkan et al. [8]. Each datapoint is a real online comment curated from the Civil Comments platform, a commenting plugin for independent news sites. For classes, each comment is labeled as either toxic or not toxic. For spurious attributes, each comment is also labeled with the demographic identities {male, female, LGBTQ, Christian, Muslim, other religions, Black, White} mentioned; multiple identities may be mentioned per comment.

The goal is to classify the comment $\mathcal{Y} = \{\text{toxic, not toxic}\}$. As in Koh et al. [21], we evaluate with $\mathcal{A} = \{\text{male, female, LGBTQ, Christian, Muslim, other religions, Black, White}\}$. There are then 16 total groups corresponding to (toxic, identity) and (not toxic, identity) for each identity. Groups may overlap; a datapoint falls in a group if it mentions the identity. We use the default data splits [21]. In Table 2, we list the percentage of toxic comments for each identity based on the groups. Trained ERM models in particular perform less well on the rarer toxic groups.

Table 2: Percent of toxic comments for each identity in the CivilComments-WILDS training set.

Identity	male	female	LGBTQ	Christian	Muslim	other religions	Black	White
% toxic	14.9	13.7	26.9	9.1	22.4	15.3	31.4	28.0

D.2 Implementation details

D.2.1 Reported metrics

Main results. For the CMNIST*, Waterbirds, and CelebA datasets, we run CNC with three different seeds, and report the average worst-group accuracy over these three trials in Table 1. As we use the same baselines and comparable methods as Liu et al. [25], we referenced their main results for the reported numbers, which did not have standard deviations or error bars reported. For CivilComments-WILDS, due to time and compute constraints we only reported one run. We note that CMNIST* here is extremely challenging, as minority groups together only make up 0.5% of the training set. This severe imbalance explains the very poor worst-group performance of ERM (as well as a couple other methods that fail to sufficiently remediate issue).

Estimated mutual information. We give further details for calculating the representation metric introduced in Sec. 3. As a reminder, we report both alignment and estimated mutual information metrics to quantify how dependent a model's learned representations are on the class labels versus the spurious attributes, and compute both metrics on the representations $Z = \{f_{\text{enc}}(x)\}$ over all test set data points x. Then to supplement the alignment loss calculation in Sec. 3, we also estimate $I(Y;Z)$ and $I(A;Z)$, the mutual information between the model's data representations and the class labels and spurious attribute labels respectively.

To first estimate mutual information with Y, we first approximate $p(y \mid z)$ by fitting a multinomial logistic regression model over all representations Z to classify y. With the empirical class label distribution $p(y)$, we compute:

$$\hat{I}(Y;Z) = \frac{1}{|Z|} \sum_{z \in Z} \sum_{y \in Y} p(y \mid z) \log \frac{p(y \mid z)}{p(y)} \tag{16}$$

We do the same but substitute the spurious attributes a for y to compute $\hat{I}(A;Z)$.

D.2.2 Stage 1 ERM training details

We describe the model selection criterion, architecture, and training hyperparameters for the initial ERM model in our method. To select this model, recall that we first train an ERM model to predict the class labels, as the model may also learn dependencies on the spurious attributes. Because we then use the model's predictions on the training data to infer samples with different spurious attribute values but the same class label, we prefer an initial ERM model that better learns this spurious dependency, and importantly also does not overfit to the training data. Inspired by the results in prior work [40, 25], we then explored using either a standard ERM model, one with high ℓ-2 regularization (`weight decay = 1`), or one only trained on a few number of epochs. To select among these, because the validation data has both class labels and spurious attributes, we choose the model with the largest gap between worst-group and average accuracy on the validation set. For fair comparison to JTT, we use the same batch size, learning rate, momentum, optimizer, default weight decay, and number of epochs as reported in [25] to obtain these models. We detail the ERM architecture and hyperparameters for each dataset below:

Colored MNIST. We use the LeNet-5 CNN architecture in the `pytorch` image classification tutorial. We train with SGD, few epochs $E = 5$, SGD, learning rate 1e-3, batch size 32, default weight decay 5e-4, and momentum 0.9.

Waterbirds. We use the `torchvision` implementation of ResNet-50 with pretrained weights from ImageNet as in Sagawa et al. [37]. Also as in [37], we train with SGD, default epochs $E = 300$, learning rate 1e-3, batch size 128, and momentum 0.9. However we use high weight decay 1.0.

CelebA. We also use the `torchvision` ImageNet-pretrained ResNet-50 and default hyperparameters from Sagawa et al. [37] but with high weight decay: we train with SGD, default epochs $E = 50$, learning rate 1e-4, batch size 128, momentum 0.9, and high weight decay 0.1.

CivilComments-WILDS. We use the HuggingFace (`pytorch-transformers`) implementation of BERT with pretrained weights and number of tokens capped at 300 as in Koh et al. [21]. As in Liu et al. [25], with other hyperparameters set to their defaults [21] we tune between using the AdamW optimizer with learning rate 1e-5 and SGD with learning rate 1e-5, momentum 0.9, and the PyTorch `ReduceLROnPlateau` learning rate scheduler. Based on our criterion, we use SGD, few number of epochs $E = 2$, learning rate 1e-5, batch size 16, default weight decay 1e-2, and momentum 0.9.

D.2.3 Contrastive batch sampling details

We provide further details related to collecting predictions from the trained ERM models, and the number of positives and negatives that determine the contrastive batch size.

ERM model prediction. To collect trained ERM model predictions on the training data, we explored two approaches: (1) using the actual predictions, i.e. the argmax for each classifier layer output vector, and (2) clustering the representations, or the last hidden-layer outputs, and assigning a cluster-specific label to each data point in one cluster. This latter approach is inspired by Sohoni et al. [40], and we similarly note that ERM models trained to predict class labels in spuriously correlated data may learn data representations that are clusterable by spurious attribute. As a viable alternative to collecting the "actual" predictions of the trained ERM model on the training data, with C classes, we can then cluster these representations into C clusters, assign the same class label only to each data point in the same cluster, and choose the label-cluster assignment that leads to the highest accuracy on the training data. We also follow their procedure to first apply UMAP dimensionality reduction to 2 UMAP components, before clustering with K-means or GMM [40]. To choose between all approaches, we selected the procedure that lead to highest worst-group accuracy on the validation data after the second-stage of training. While this cluster-based prediction approach was chosen as a computationally efficient heuristic, we found that in practice it either lead to comparable or better final worst-group accuracy on the validation set. To better understand this, as a preliminary result we found that when visualizing the validation set predictions with the Waterbirds dataset, the cluster-based predictions captured the actual spurious attributes better than the classifier layer predictions (Fig. 7). We defer additional discussion to Sohoni et al. [40] and leave further analysis to future work.

Number of positives and negatives per batch. One additional difference between our work and prior contrastive learning methods [9, 20] is that we specifically construct our contrastive batches by sampling anchors, positives, and negatives first. This is different from the standard procedure of randomly dividing the training data into batches first, and then assigning the anchor, positive, and

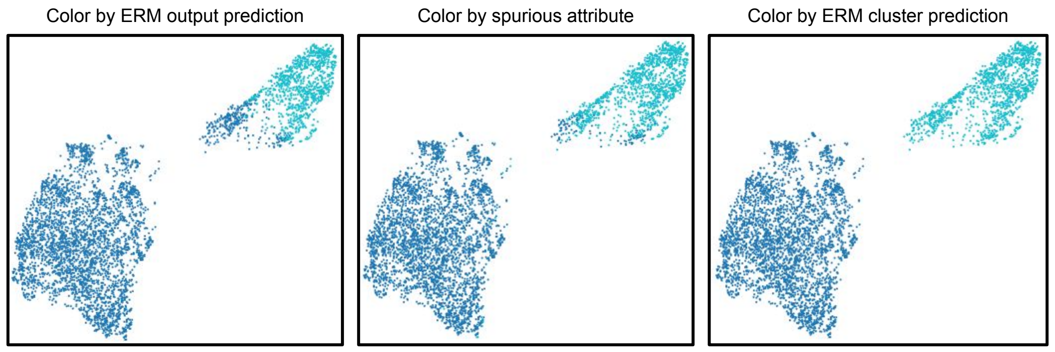


Figure 7: UMAP visualization of ERM data representations for the Waterbirds training data. We visualize the last hidden layer outputs for a trained ERM ResNet-50 model given training samples from Waterbirds, coloring by either the ERM model's "standard" predictions, the actual spurious attribute values (included here just for analysis), and predictions computed by clustering the representations as described above. Clustering-based predictions more closely align with the actual spurious attributes than the ERM model outputs.

negative roles to each datapoint in a given batch. As a result, we introduce the number of positives M and the number of negatives N as two hyperparameters that primarily influence the size of each contrastive batch (with number of additional anchors and negatives also following M and N with two-sided batches). To maximize the number of positive and negative comparisons, as a default we set M and N to be the maximum number of positives and negatives that fit the sampling criteria specified under Algorithm 3 that also can fit in memory. In Appendix D.2.4, for each dataset we detail the ERM prediction method and number of positives and negatives sampled in each batch.

D.2.4 Stage 2 contrastive model training details

In this section we describe the model architectures and training hyperparameters used for training the second model of our procedure, corresponding the reported worst-group and average test set results in Table 1. In this second stage, we train a new model with the same architecture as the initial ERM model, but now with a contrastive loss and batches sampled based on the initial ERM model's predictions. We report test set worst-group and average accuracies from models selected with hyperparameter tuning and early stopping based on the highest validation set worst-group accuracy. For all datasets, we sample contrastive batches using the clustering-based predictions of the initial ERM model. Each batch size specified here is also a direct function of the number of positives and negatives: $2M + 2N$.

Colored MNIST. We train a LeNet-5 CNN. For CNC, we use $M = 32$, $N = 32$, batch size 128, temperature $\tau = 0.05$, contrastive weight $\lambda = 0.75$, SGD optimizer, learning rate 1e-3, momentum 0.9, and weight decay 1e-4. We train for 3 epochs, and use gradient accumulation to update model parameters every 32 batches.

Waterbirds. We train a ResNet-50 CNN with pretrained ImageNet weights. For CNC, we use $M = 17$, $N = 17$, batch size 68, temperature $\tau = 0.1$, contrastive weight $\lambda = 0.75$, SGD optimizer, learning rate 1e-4, momentum 0.9, weight decay 1e-3. We train for 5 epochs, and use gradient accumulation to update model parameters every 32 batches.

CelebA. We train a ResNet-50 CNN with pretrained ImageNet weights. For CNC, we use $M = 64$, $N = 64$, batch size 256, temperature $\tau = 0.05$, contrastive weight $\lambda = 0.75$, SGD optimizer, learning rate 1e-5, momentum 0.9, and weight decay 1e-1. We train for 15 epochs, and use gradient accumulation to update model parameters every 32 batches.

CivilComments-WILDS. We train a BERT model with pretrained weights and max number of tokens 300. For CNC, we use $M = 16$, $N = 16$, batch size 64, temperature $\tau = 0.1$, contrastive weight $\lambda = 0.75$, AdamW optimizer, learning rate 1e-4, weight decay 1e-2, and clipped gradient norms. We train for 10 epochs, and use gradient accumulation to update weights every 128 batches.

D.2.5 Comparison method training details

As reported in the main results (Table 1) we compare CNC with the ERM and Group DRO baselines, as well as robust training methods that do not require spurious attribute labels for the training data: CVaR DRO [24], GEORGE [24], Learning from Failure (LfF) [24], Contrastive Input Morphing (CIM) [42], Environment Inference for Invariant Learning (EIIL) [12], and Just Train Twice (JTT) [25]. For each dataset, we use the same model architecture for all methods. For the Waterbirds, CelebA, and CivilComments-WILDS datasets, we report the worst-group and average accuracies reported in Liu et al. [25] for ERM, CVaR DRO, LfF, and JTT. For GEORGE, we report the accuracies reported in Sohoni et al. [40]. For CIM, we report results from Waterbirds and CelebA from [42] using the CIM + variational information bottleneck implementation [2], which achieves the best worst-group performance in their results. For EIIL, we report results from Waterbirds and CivilComments-WILDS from [12]. For these hyperparameters, we defer to the original papers. For GDRO, we reproduce the results with the same optimal hyperparameters over three seeds. For Colored MNIST, we run implementations for GEORGE, CIM, EIIL, JTT, and GDRO, using code from their authors respectively. LfF and CVaR DRO are also run with code from the JTT authors. We include training details for our own implementations below:

Colored MNIST (CMNIST*). We run all methods for 100 epochs, reporting test set accuracies with early stopping. For JTT, we train with SGD, learning rate 1e-3, momentum 0.9, weight decay 5e-4, batch size 32. We use the same initial ERM model as CNC, with hyperparameters described in Appendix D.2.2. For upsampling we first tried constant factors $\{10, 100, 1000\}$. We also tried a resampling strategy where for all the datapoints with the same initial ERM model prediction, we upsample the incorrect points such that they equal the correct points in frequency, and found this worked the best. With $p_{\text{corr}} = 0.995$, this upsamples each incorrect point by roughly 1100. We also use this approach for the results in Fig. 9. For GDRO we use the same training hyperparameters as JTT, but without the upsampling and instead set group adjustment parameter $C = 0$. For LfF, we use the same hyperparameters as JTT, but instead of upsampling gridsearched the q parameter $\in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, using $q = 0.7$. For CVaR DRO we do the same but use hyperparameter $\alpha = 0.1$. For GEORGE we train with SGD, learning rate 1e-3, momentum 0.9, weight decay 5e-4. For CIM, we use the CIM + VIB implementation. We train with SGD, learning rate 1e-3, weight decay 5e-4, β parameter 10, and λ parameter 1e-5. For EIIL, for environment inference we use the same initial ERM model as CNC and JTT, and update the soft environment assignment distribution with Adam optimizer, learning rate 1e-3, and 10000 steps. Following [12]'s own colored MNIST experiment, we train the second model with IRM, using learning rate 1e-2, weight decay 1e-3, penalty weight 100, and penalty annealing parameter 80.

CelebA. We also tune EIIL for CelebA. We again use the same initial ERM model as CNC, and update the soft environment assignment distribution with Adam optimizer, learning rate 1e-3, and 10000 steps. We train the second model with GDRO, using SGD, 50 epochs, learning rate 1e-5, batch size 128, weight decay 0.1, and group adjustment parameter 3.

In Section D.2.6, we include our sweeps for both the method-specific and general hyperparameters in CIM, EIIL, JTT, and CNC.

Comparison limitations. One limitation of our comparison is that because for each dataset we sample new contrastive batches which could repeat certain datapoints, the number of total batches per epoch changes. For example, 50 epochs training the second model in CNC does not necessarily lead to the same total number of training batches as 50 epochs training with ERM, even if they use the same batch size. However, we note that the numbers we compare against from Liu et al. [25] are reported with early stopping. In this sense we are comparing the best possible worst-group accuracies obtained by the methods, not the highest worst-group accuracy achieved within a limited number of training batches. We also found that although in general the time to complete one epoch takes much longer with CNC, CNC requires fewer overall training epochs for all but the CivilComments-WILDS dataset to obtain the highest reported accuracy.

D.2.6 Hyperparameter sweeps

To fairly compare with previous methods [25, 12, 42], we use the same evaluation scheme (selecting models based on worst-group validation error), and sweep over a consistent number of hyperparameters, i.e. number of validation set queries. We set this number for CNC to be a comparable number of queries that is reported in prior works. We break this down into method-specific (e.g. contrastive temperature in CNC, upweighting factor in JTT), and shared (e.g. learning rate) hyperparameter categories.

Method-specific For CNC, we tune three method-specific hyperparameters: contrastive loss temperature (Eq. 1), contrastive weight (Eq. 5), and gradient accumulation steps values as in Table 3.

Table 3: Method-specific hyperparameters for CNC.

Hyperparameter	Dataset	Values
Temperature (τ)	All	$\{0.05, 0.1\}$
Contrastive Weight (λ)	All	$\{0.5, 0.75\}$
Gradient Accumulation Steps	CMNIST*, Waterbirds, CelebA	$\{32, 64\}$
	CivilComments-WILDS	$\{32, 64, 128\}$

For JTT, the reported results and our CMNIST* implementation are tuned over the following hyperparameters in Table 4.

Table 4: Method-specific hyperparameters for JTT.

Hyperparameter	Dataset	Values
Stage 1 Training Epochs	Waterbirds	$\{40, 50, 60\}$
	CMNIST*, CelebA, CivilComments-WILDS	$\{1, 2\}$
Upweighting Factor	CMNIST*	$\{10, 100, 1000, 1100\}$
	Waterbirds, CelebA	$\{20, 50, 100\}$
	CivilComments-WILDS	$\{4, 5, 6\}$

For EIIL, our CMNIST* and CelebA implementations are tuned over hyperparameters reported in Table 5. [12] report that they allow up to 20 evaluations with different hyperparameters for Waterbirds and CivilComments-WILDS. When using GDRO as the second stage model, they also report using the same hyperparameters as the GDRO baseline for Waterbirds. We do the same for our evaluation on CelebA. This amounts to primarily tuning the first stage environment inference learning rate and number of updating steps for CMNIST* and CelebA, and the penalty annealing iterations and penalty weight for the IRM second stage model for CMNIST*.

Table 5: Method-specific hyperparameters for EIIL.

Hyperparameter	Dataset	Values
Environment Inference Learning Rate	CMNIST*, CelebA	{1e-1, 1e-2, 1e-3}
Environment Inference Update Steps	CMNIST*, CelebA	{10000, 20000}
IRM Penalty Weight	CMNIST*	{0.1, 10, 1000, 1e5}
IRM Penalty Annealing Iterations	CMNIST*	{10, 50, 80}
GDRO Group Adjustment	CelebA	{0, 2, 3}

For CIM, our CMNIST* implementation uses CIM + VIB, and is tuned over the β VIB parameter [2] and contrastive weighting parameter λ for CIM in Table 6. [42] report tuning over a range of values within [1e-5, 1] for λ on the CelebA and Waterbirds datasets.

Table 6: Method-specific hyperparameters for CIM.

Hyperparameter	Dataset	Values
CIM λ	CMNIST*	{0.01, 0.05, 0.1}
VIB β	CMNIST*	{1e-5, 1e-3, 1e-1, 10}

Shared For all datasets, we use the same optimizer and momentum (if applicable) as reported in the JTT paper. Table 7 contains the data-specific shared hyperparameter values tried.

Table 7: Shared hyperparameters

	CMNIST*	Waterbirds	CelebA	CivilComments-WILDS
Learning Rate	{1e-4, 1e-3, 1e-2}	{1e-4, 1e-3}	{1e-5, 1e-4}	{1e-5, 1e-4}
Weight Decay	{1e-4, 5e-4}	{1e-4, 1e-3}	{1e-2, 1e-1}	{1e-2}

D.3 CNC compute resources and training time

All experiments for CMNIST*, Waterbirds, and CelebA were run on a machine with 14 CPU cores and a single NVIDIA Tesla P100 GPU. Experiments for CivilComments-WILDS were run on an Amazon EC2 instance with eight CPUs and one NVIDIA Tesla V100 GPU.

Regarding runtime, one limitation with the current implementation of CNC is its comparatively longer training time compared to methods such as standard ERM. This is both a result of training an initial ERM model in the first stage, and training another model with contrastive learning in the second stage. In Table 8 we report both how long it takes to train the initial ERM model and long it takes to complete one contrastive training epoch on each dataset. We observe that while in some cases training the initial ERM model is negligible, especially if we employ training with only a few epochs to prevent memorization (for Colored MNIST it takes roughly two minutes to obtain a sufficient initial ERM model), it takes roughly 1.5 and 3 hours to train the high regularization initial models used for Waterbirds and CelebA. While these hurdles are shared by all methods that train an initial ERM model, we find that the second stage of CNC occupies the bulk of training time. Prior work has shown that contrastive learning typically requires longer training times and converges more slowly than supervised learning [9]. We also observe this in our work.

We note however that because we sample batches based on the ERM model's predictions, the contrastive training duration is limited by how many datapoints the initial ERM model predicts incorrectly. In moderately sized datasets with very few datapoints in minority groups, (e.g. Waterbirds, which has roughly 4794 training points and only 56 datapoints in its smallest group), the total time it takes to train CNC is on par with ERM. Additionally, other methods such as additional hard negative mining [36] have been shown to improve the efficiency of contrastive learning, and we can incorporate these components to speed up training time as well.

Table 8: CNC Average total training time for first and second stages of CNC

Dataset	CMNIST*	Waterbirds	CelebA	CivilComments-WILDS
Stage 1 ERM train time	2 min.	1.5 hrs	3 hrs	3.1 hrs
Stage 2 CNC train time	1.2 hrs	1.8 hrs	32.2 hrs	37.6 hrs

E Limitations and potential negative societal impacts

E.1 Limitations

One limitation of our work is that it requires training an ERM model, and then training a second contrastive model based on the ERM model's predictions. Thus, our approach can be more expensive in terms of training time. This could potentially be reduced by training the ERM model for fewer epochs (as in JTT). Another limitation is that we rely on spurious attribute labels being available on the validation set (in order to select the best model checkpoint). This is a standard assumption made by most methods we compare to, though Sohoni et al. [40] estimate validation group labels as well.

E.2 Potential negative societal impacts

While our work successfully improves worst-group accuracy, this is not the end-all be-all metric - other fairness-based metrics may be more suitable in different settings; misuse of metrics could lead to potential harm. To avoid these pitfalls, it is important for practitioners to understand the limitations and tradeoffs of different metrics when applying methods such as ours.

F Additional evaluations and ablations

F.1 Visualization of learned data representations

In addition to the reported representation metrics in Fig. 3, we visualize and compare the learned representations of test set samples from models trained with ERM, JTT, and CNC in Fig. 8. Compared to ERM models, both JTT and CNC models learn representations that better depict dependencies on the class labels. However, especially with the Waterbirds and CelebA datasets, CNC model representations more clearly depict dependencies only on the class label, as opposed to JTT models which also show some organization by the spurious attribute still.

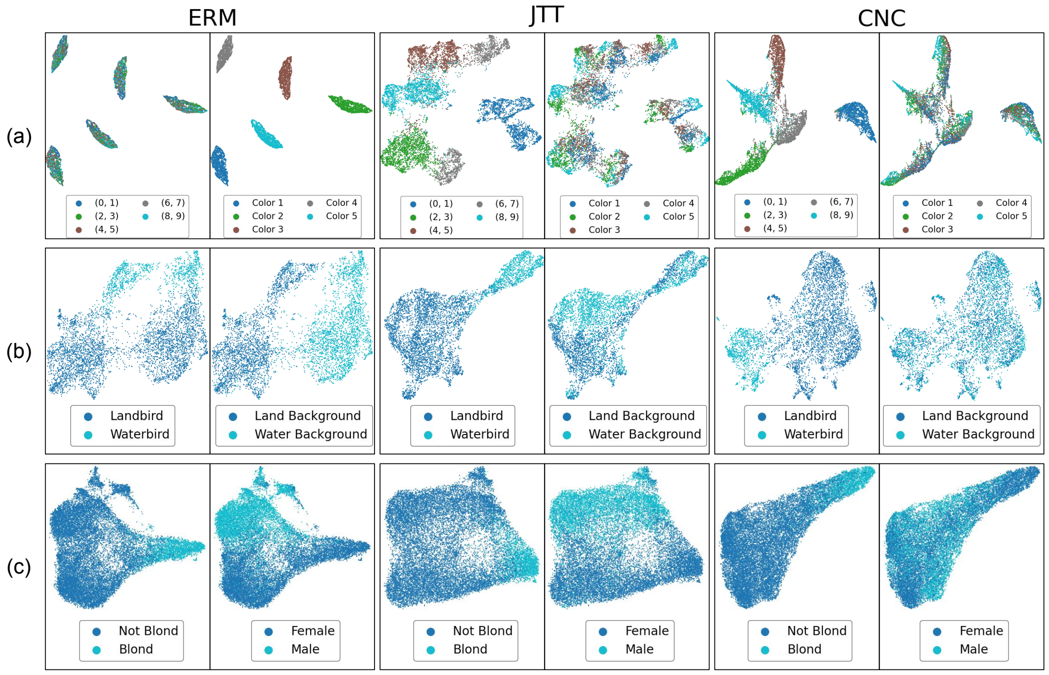


Figure 8: UMAP visualizations of learned representations for Colored MNIST (a), Waterbirds (b), and CelebA (c). We color data points based on the class label (left) and spurious attribute (right). Most consistently across datasets, CNC representations exhibit dependence and separability by the class label but not the spurious attribute, suggesting that they best learn features which only help classify class labels.

F.2 Representation metric ablation with increasingly spurious datasets

To study how the relation between representation metrics and worst-group accuracy reported in Fig. 3 and Table 1 scales with the strength of the spurious correlation, we compute representation metrics with CNC, ERM, and JTT models trained on increasingly spurious ($\uparrow p_{\text{corr}}$) CMNIST* datasets in Fig. 9. We observe that with high spurious correlations, ERM fails to classify digits in the minority classes, while CNC and JTT comparably maintain high worst-group accuracy. CNC also performs better in more spurious settings ($p_{\text{corr}} > 0.95$). These improvements over ERM are reflected by drops in alignment loss (averaged over classes); CNC consistently achieves lowest such loss. Fig. 9c shows that CNC's learned representations maintain a more favorable balance of mutual information between the class label and spurious attribute than JTT. While JTT models exhibit slightly higher estimated $I(Y;Z)$ than CNC models, CNC models exhibit much lower dependence on the spurious attribute.

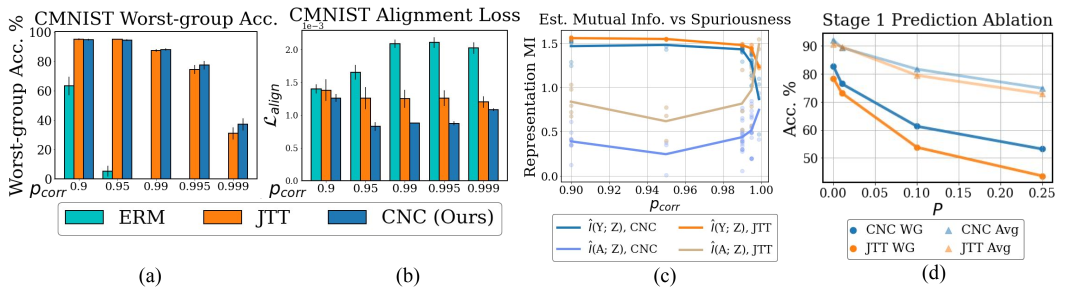


Figure 9: Alignment loss and mutual information representation metrics with worst-group accuracy on increasingly spurious CMNIST*. CNC highest worst-group accuracy (a) coincides with learning representations with better alignment (b) and ratio of mutual information dependence on the labels vs the spurious attribute (c).

F.3 Understanding CNC's sensitivity to Stage 1 predictions

We also study how sensitive CNC is to how closely the Stage 1 ERM model actually predicts the spurious attribute. As JTT also relies on an initial ERM model's predictions, we compare CNC to JTT in this regard. We find that CNC is more robust to noisy ERM predictions than JTT, and that CNC does not require perfectly inferred groups to perform well.

We first conduct an ablation on CNC and JTT's worst-group and average performance in Fig. 9d with the following synthetic experiment. On CMNIST*, we start with the true spurious attribute labels as the Stage 1 "predictions". We then gradually degrade their quality as follows: for each point, with probability p we change its assigned spurious attribute label to a different label chosen uniformly at random. Both methods' performance degrades as p increases and the Stage 1 "predictions" degrade. However, CNC consistently achieves higher worst-group accuracy and smaller worst-group versus average accuracy gaps. We also observe on other datasets that CNC does not require perfectly spurious ERM predictions to work well. For the Waterbirds and CelebA results in Table 1, the Stage 1 ERM model predictions align with the spurious attribute value 94.7% and 59.3% of the time respectively. While the ERM model is far from perfect at recognizing the spurious attributes, CNC still substantially reduces the worst-group vs. average accuracy gap.

F.4 Empirical validation of CNC components

We validate the design choices of CNC through various ablations studying the effects of the individual components of our method on worst-group accuracy.

F.4.1 Importance of ERM-guided contrastive sampling

Although CNC relies on an initial trained ERM model's predictions, can we still improve worst-group accuracy without this step and with supervised contrastive learning alone, i.e. by sampling positives uniform randomly from *all* datapoints with the same label as the anchor? In Table 9, we show that this "vanilla" contrastive learning implementation competes with ERM on Waterbirds and outperforms ERM on CelebA in worst-group accuracy, while achieving similar average accuracy (standard deviation with three seeds recorded in parenthesis).

However, we still achieve substantially higher worst-group accuracy with our full method. This supports that using initial ERM predictions to sample "hard comparisons" helps obtain best observed worst-group accuracy. We conjecture this is the case because positive samples with the ERM model are not only the same class as anchors, but different enough such that an initial trained ERM model classified them differently. Comparing these against negatives which are also presumably more similar to the anchors than in the no-ERM setup lends additional contrastive learning signal to not learn dependencies on spurious differences.

F.4.2 Additional design choice ablations

To validate the additional algorithmic components of CNC, we report how CNC performs on the Waterbirds dataset when modifying the individual design components. We use the same hyperparameters as in the main results, and report accuracies as the average over three training runs for

Table 9: CNC accuracies without initial ERM model for contrastive sampling

Method	Waterbirds		CelebA	
Accuracy (%)	Worst-group	Average	Worst-group	Average
ERM	72.6	85.9	47.2	95.6
CNC (no ERM)	71.0 (1.92)	85.9 (0.8)	60.8 (0.8)	84.9 (0.4)
CNC (Ours)	89.7 (0.2)	90.8 (0.1)	88.8 (0.9)	89.85 (0.5)

Table 10: Ablation over CNC algorithmic components on Waterbirds. Default choices achieve highest worst-group and average accuracy.

Method	CNC (Default)	Projection Head	One-sided Contrasting	Train + Finetune
WG Acc. (%)	**89.7** (0.2)	82.4 (1.8)	85.2 (3.6)	84.0 (1.7)
Avg. Acc. (%)	**90.8** (0.1)	88.7 (0.6)	90.1 (1.6)	87.7 (1.1)

the following ablations. Table 10 summarizes that across these design ablations, default CNC as presented consistently outperforms these alternative implementations.

No projection head. We incorporate a nonlinear projection head as is typical in prior contrastive learning works [9], that maps the encoder output to lower-dimensional representations (from 2048 to 128 in our case). We then update the encoder layers and the projection head jointly by computing the contrastive loss on the projection head's output, still passing the encoder layer's direct outputs to the classifier to compute the cross-entropy loss. We note that using the projection head decreases worst-group accuracy substantially. We reason that as previously discussed, while using the projection head in prior work can allow the model to retain more information in its actual hidden layers [9], in our case to remove dependencies on spurious attributes we actually want to encourage learning invariant representations when we model the differences between anchor and positive datapoints as due to spurious attributes.

Two-sided contrastive batches. Instead of "two-sided" contrasting where we allow both sampled anchors and positives to take on the anchor role, for each batch we only compute contrastive updates by comparing original positives and negatives with the original anchor. When keeping everything else the same, we find that just doing these one-sided comparisons also leads to a drop in performance for worst-group accuracy. This suggests that the increased number of comparisons and training setup where we swap the roles of anchors and positives of the two-sided batches introduces greater contrastive learning signal.

Joint training of encoder and classifier layers. Instead of training the full model jointly, we first only train the encoder layers with the contrastive loss in CNC, before freezing these layers and finetuning the classifier layers with the cross-entropy loss. With this implementation, we also obtain noticeable drop in performance. While we leave further analysis for the joint cross-entropy and contrastive optimization for future work, one conjecture is that the cross-entropy loss may aid in learning separable representations while also training the full model to keep the average error small. From our theory, the contrastive loss can help bound the gap between worst-group and average error, and so to improve worst-group performance it may make sense to also try to minimize average error in the same parameter update.

This also follows prior work, where updating the entire model and finetuning all model parameters instead of freezing the encoder layers leads to higher accuracy [9]. However, we found that with an initial encoder-only training stage, if we did not freeze the trained layers the fine-tuning on a dataset with spurious correlations would "revert" the contrastive training, resulting in a large gap between worst-group and average error similar to ERM.

F.5 Additional results

In Table 11, we provide standard deviations for CNC, as well as GDRO [37] and JTT [25] (both of which we re-ran ourselves using the publicly available source code) and GEORGE [40] (from their paper). For GDRO and JTT, we use the reported hyperparameters in their original works respectively, but note that the JTT numbers here are slightly different than those reported in [26] (and thus Table 1).

Table 11: Worst-group and average accuracies (in percent), along with their standard deviations over 3 trials.

Method	CMNIST*		Waterbirds		CelebA	
Accuracy (St.Dev.)	Worst-group	Avg.	Worst-group	Avg.	Worst-group	Avg.
GEORGE	76.4 (2.3)	89.5 (0.3)	83.8 (1.6)	90.5 (0.9)	54.9 (2.2)	94.5 (0.2)
JTT	74.5 (3.1)	90.2 (0.8)	83.4 (0.9)	90.8 (0.7)	79.6 (0.9)	89.4 (1.5)
CNC (Ours)	**77.4 (3.0)**	90.9 (0.6)	**89.7 (0.2)**	90.8 (0.1)	**88.8 (0.9)**	89.9 (0.5)
GDRO	78.5 (4.5)	90.6 (0.1)	91.1 (0.2)	92.4 (0.2)	88.9 (1.3)	93.9 (0.1)

F.6 Additional GradCAM visualizations

On the next two pages, we include additional GradCAM visualizations depicting saliency maps for samples from each group in the Waterbirds and CelebA datasets. Warmer colors denote higher saliency, suggesting that the model considered these pixels more important in making the final classification as measured by gradient activations. For both datasets, we compare maps from models trained with ERM, the next most competitive method for worst-group accuracy JTT, and CNC. CNC models most consistently measure highest saliency with pixels directly associated with class labels and not spurious attributes.

G Code

Anonymized code for CNC is available at this link.

G.0.1 Waterbirds

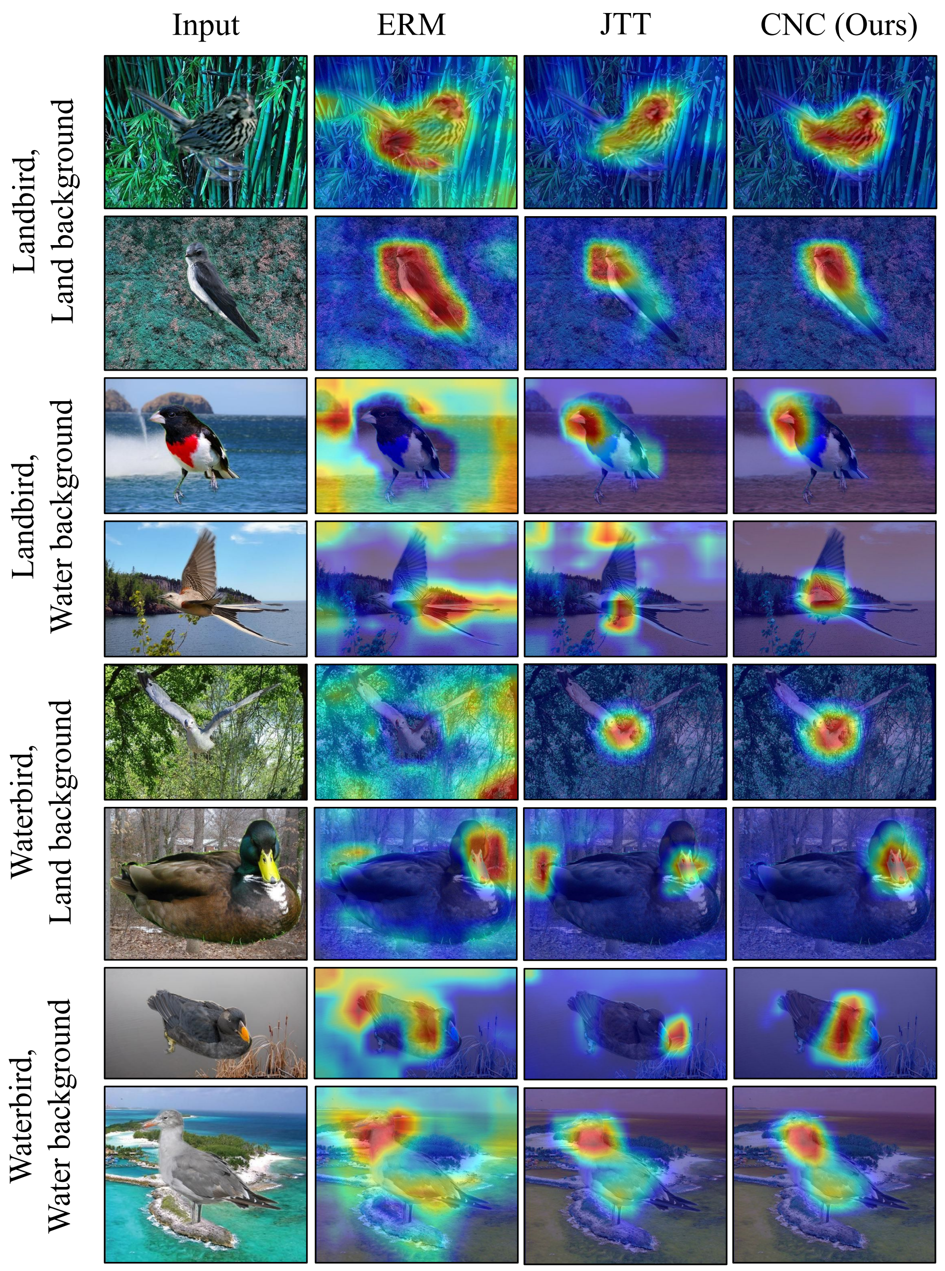

Input
ERM
JTT
CNC (Ours)
Landbird, Land background
Landbird, Water background
Waterbird, Land background
Waterbird, Water background

Figure 10: Additional GradCAM visualizations for the Waterbirds dataset. We use GradCAM to visualize the "salient" observed features used to classify images by bird type for models trained with ERM, JTT, and CNC. ERM models output higher salience for spurious background attribute pixels, sometimes almost exclusively. JTT and CNC models correct for this, with CNC better exclusively focusing on bird pixels.

G.0.2 CelebA

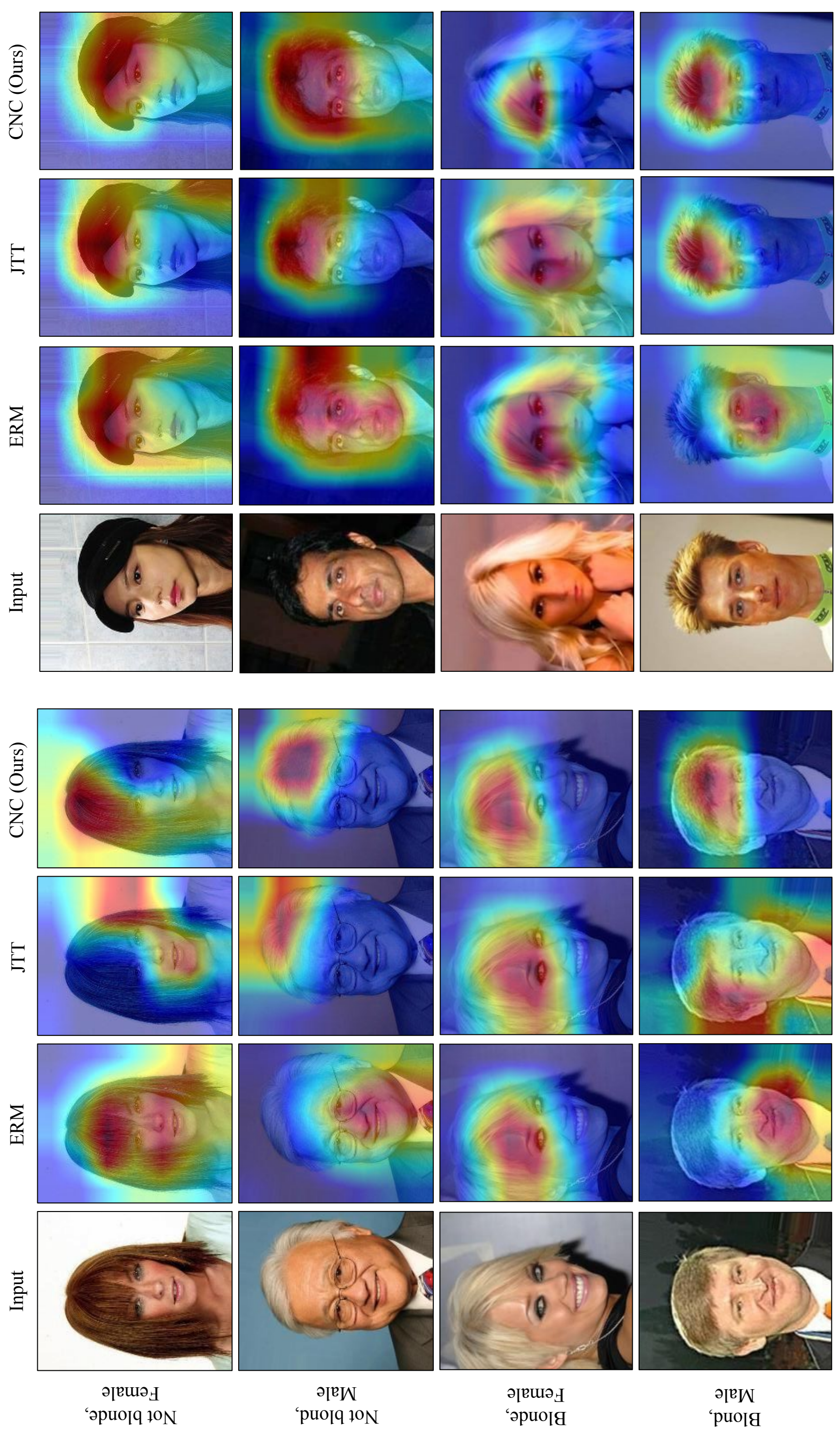


Figure 11: Additional GradCAM visualizations for the CelebA dataset. For models trained with ERM, JTT, and CNC, we use GradCAM to visualize the "salient" observed features used to classify whether a celebrity has blond(e) hair. ERM models interesting also "ignore" the actual hair pixels in favor of other pixels, presumably associated with the spurious gender attribute. In contrast, GradCAMs for JTT and CNC models usually depict higher salience for regions that at least include hair pixels. CNC models most consistently do so.