A. Training Details

In this section, we detail the model architectures and hyperparameters used by each approach. Within each dataset, we used the same model architecture across all approaches: ResNet-50 (He et al., 2016) for Waterbirds and CelebA, and BERT for MultiNLI and CivilComments (Devlin et al., 2019). For ResNet-50, we used the PyTorch (Paszke et al., 2017) implementation of ResNet-50, starting from ImageNet-pretrained weights. For BERT, we used the the HuggingFace implementation (Wolf et al., 2019) of BERT, also starting from pretrained weights.

We use the LfF implementation released by Nam et al. (2020). We use the group DRO and ERM implementations released by Sagawa et al. (2020a) and also implement CVaR DRO and JTT on top of this code base, with the CVaR DRO implementation adapted from Levy et al. (2020). For the group DRO experiments on Waterbirds, CelebA, and MultiNLI, we directly use the reported performance numbers from Sagawa et al. (2020a). We note that these numbers utilize group-specific loss adjustments that encourage the model to attain lower training losses on smaller groups, which was shown to improve worst-group generalization. We train our own group DRO model on CivilComments-WILDS as it was not included in Sagawa et al. (2020a); for this, we did not implement these group adjustments. We train our own models for all other algorithms.

For all approaches, we tune all hyperparameters as well as early stop based on highest worst-group accuracy on the validation set. On top of the hyperparameters shared between all algorithms (e.g., learning rate, ℓ_2 regularization), which we detail for each dataset below, CVaR DRO, JTT, and LfF have additional hyperparameters that we tuned separately for each dataset:

- For **CVaR DRO**, we tune the size of the worst-case subpopulation $\alpha \in \{0.1, 0.2, 0.5\}$. For CelebA, we additionally tried $\alpha = \frac{\text{\# smallest group examples}}{\text{\# training examples}} = 0.00852$.
- For **LfF**, we tuned the hyperparameter q by grid searching over $q \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and also sampling two values log-uniformly from $(0, 0.1]$ in all datasets except CelebA, where we use the value of $q = 0.7$ used in Nam et al. (2020).
- For **JTT**, we alternated between grid searching over (learning rate, weight decay) and (T, λ_{up}) to avoid searching over all combinations. We first heuristically chose initial values for T and λ_{up}, and then alternated between searching over learning rates and weight decays while keeping T and λ_{up} fixed, and searching over T and λ_{up} while keeping the learning rate and weight decay fixed. We searched over both pairs twice. We describe the dataset-specific grids for learning rate and weight decay below. We used initial values of $T = 50$ for Waterbirds, and $T = 1$ for the other datasets. We additionally tuned over $T \in \{40, 50, 60\}$ for Waterbirds and $T \in \{1, 2\}$ for the other datasets. We used initial value of $\lambda_{\text{up}} = 50$ for Waterbirds and CelebA, and $\lambda_{\text{up}} = 10$ for MultiNLI and CivilComments. We additionally tuned of $\lambda_{\text{up}} \in \{5, 10, 20, 30, 40, 50, 100, \frac{|\text{error set}|}{|\text{training set}|}\}$ for all datasets. For MultiNLI and CivilComments, we found smaller λ_{up} to perform better, so we additionally tuned over $\lambda_{\text{up}} \in \{3, 4, 6, 7\}$.

ERM has no additional algorithm-specific hyperparameters. For group DRO, we fixed the step size η_q for updating group weights to its default value of 0.01 from Sagawa et al. (2020a), without tuning.

In general, for JTT, we fixed the initialization model and the final model to share the same hyperparameters, with two exceptions. First, the initialization model is trained only for T epochs, whereas the final model is trained for longer; exact values vary by dataset. Second, for BERT, we found that it was helpful for JTT to be able to choose different optimizers for the initialization model and final model. Specifically, for the ResNet-50 models, we used SGD with momentum 0.9 and no learning rate scheduler or gradient clipping. For BERT, we additionally considered the standard AdamW optimizer Loshchilov & Hutter (2017) with a linearly-decaying learning rate and gradient clipping (setting the max ℓ_2-norm of the gradients to 1), but we set this as a hyperparameter and allowed the initialization and final models to independently be optimized by SGD or AdamW.

Waterbirds. All approaches are optimized for up to 300 epochs with batch size 64, using batch normalization (Ioffe & Szegedy, 2015), and no data augmentation. We chose this smaller batch size (compared to the batch size of 128 used in Sagawa et al. (2020a)) for computational convenience.

For LfF, we adopt the learning rate, ℓ_2-regularization strength, and optimizer from the CelebA experiments in Nam et al. (2020). However, we note that we change the batch size, total number of epochs, and the model from ResNet-18 to ResNet-50 for consistency with the other methods.

Accordingly, LfF uses the Adam optimizer (Kingma & Ba, 2015), while the rest of the approaches use stochastic gradient descent (SGD) with momentum 0.9.

We apply ℓ_2-regularization with regularization weight $\lambda = 0.0001$ for LfF. For the other approaches, we grid searched between the regularization weights $\lambda \in \{0.0001, 1\}$. For ERM, this yields $\lambda = 0.0001$, and for JTT and CVaR DRO, this yields $\lambda = 1$.

We use learning rate 0.0001 for LfF. For the other approaches, we also grid searched over the learning rates $\{0.001, 0.00001\}$. For ERM, this yields learning rate 0.001, and for JTT and CVaR DRO, this yields learning rate 0.00001.

Our grid search over α for CVaR DRO yielded $\alpha = 0.1$. Our grid search over q for LfF yielded $q = 0.001$. For JTT, we use $\lambda_{\text{up}} = 50$ and $T = 50$ epochs.

CelebA. For LfF, we use the same learning rate, ℓ_2-regularization strength, and optimizer as in Waterbirds, which are taken from Nam et al. (2020). As above, we note that we change the batch size, total number of epochs, and the model from ResNet-18 to ResNet-50 for consistency with the other methods. Also, we note that Nam et al. (2020) selected those hyperparameters based on the performance of both $y \neq a$ groups, whereas we select all other hyperparameters based on performance on the single worst group.

We train all approaches for up to 50 epochs. For JTT, CVaR DRO, and ERM, we grid searched between the regularization weights $\lambda \in \{0.1, 0.0001\}$ and learning rates $\{0.0001, 0.00001\}$ from Sagawa et al. (2020a). For ERM, this yields $\lambda = 0.0001$ and learning rate 0.0001. For the others, this yields $\lambda = 0.1$ and learning rate 0.00001.

Following Nam et al. (2020), we use $q = 0.7$ for LfF. Our grid search over α for CVaR DRO yielded $\alpha = 0.00852$. For JTT, we use $\lambda_{\text{up}} = 50$ and $T = 1$ epoch.

MultiNLI. We train each approach for up to 5 epochs with default tokenization, dropout, batch size 32, no ℓ_2-regularization, and an initial learning rate of 0.00002.

JTT achieves the highest validation worst-group accuracy using SGD optimization without clipping for the initial model, and using the AdamW optimizer with clipping for the final model. All other approaches achieve highest validation worst-group accuracy using AdamW with clipping. Our grid search over α for CVaR DRO yielded $\alpha = 0.5$. Our grid search over q for LfF yielded $q = 0.1$. For JTT, we use $\lambda_{\text{up}} = 4$ and $T = 2$ epochs.

CivilComments-WILDS. All approaches use the details from Koh et al. (2021) We capped the number of tokens per example at 300 and used an initial learning rate of 0.00001. We train all approaches for up to 5 epochs with batch size 16 and ℓ_2 regularization strength of 0.01.

JTT achieves the highest validation worst-group accuracy using SGD optimization without clipping for the initial model, and using the AdamW optimizer with clipping for the final model. All other approaches achieve highest validation worst-group accuracy using AdamW with clipping. Our grid search over α for CVaR DRO yielded $\alpha = 0.5$. Our grid search over q for LfF yielded $q = 0.00001$. For JTT, we use $\lambda_{\text{up}} = 6$ and $T = 2$ epochs.

We also note that our group DRO approach uses a different spurious attribute compared to the group DRO results reported in Koh et al. (2021). Our group DRO uses the spurious attribute of any demographic identity being mentioned, while the one in Koh et al. (2021) uses only mentions of the Black demographic. Both perform similarly: ours achieves 0.3% lower worst-group accuracy, but 0.5% higher average accuracy.

B. Dataset Details

B.1. Waterbirds

We use the Waterbirds dataset introduced by Sagawa et al. (2020a), which is constructed by cropping out images of birds from the CUB dataset (Wah et al., 2011) and pasting them on backgrounds from the Places dataset (Zhou et al., 2017). In this dataset, images of seabirds (albatross, auklet, cormorant, frigatebird, fulmar, gull, jaeger, kittiwake, pelican, puffin, or tern) and waterfowl (gadwall, grebe, mallard, merganser, guillemot, or Pacific loon) are labeled as *waterbirds*, and all other birds are labeled as *landbirds*.

Backgrounds from the *ocean* and *natural lake* categories in the Places dataset are considered to have spurious attribute $a =$ *water background*, while backgrounds from the *bamboo forest* or *broadleaf forest* categories are considered to have spurious attribute $a =$ *land background*.

There are two minority groups: (land background, waterbird) and (water background, landbird); and two majority groups: (land background, landbird) and (water background, waterbird). We use the same training / valid / test splits from Sagawa et al. (2020a). In the training data, 95% of the waterbirds appear on water backgrounds, and 95% of the landbirds appear on land backgrounds, so the minority groups contain far fewer examples than the majority groups. In the validation and test sets, both the landbirds and waterbirds are evenly split between the water and land backgrounds.

B.2. CelebA

We use the task setup from Sagawa et al. (2020a) on the CelebA celebrity face dataset (Liu et al., 2015). The label y is set to be the *Blond_Hair* attribute, and the spurious attribute a is set to be the *Male* attribute: being female spurious correlates with having blond hair. The minority groups are (blond, male) and (not blond, female), although the (blond, male) group is significantly smaller than the (not blond, female) group. The majority groups are (blond, female) and (not blond, male). We use the standard train /

Group	Enrichment	ERM test acc.
(muslim, toxic)	8.58x	62.1%
(christian, toxic)	8.58x	57.4%
(LGBTQ, toxic)	8.58x	72.1%
(other religion, toxic)	8.56x	62.9%
(black, toxic)	8.53x	74.6%
(white, toxic)	8.49x	68.6%
(female, toxic)	8.49x	64.7%
(male, toxic)	8.48x	66.6%
(white, non-toxic)	0.09x	84.7%
(LGBTQ, non-toxic)	0.07x	82.2%
(black, non-toxic)	0.07x	74.6%
(male, non-toxic)	0.06x	93.0%
(muslim, non-toxic)	0.05x	89.4%
(female, non-toxic)	0.05x	94.7%
(other religion, non-toxic)	0.04x	93.4%
(christian, non-toxic)	0.03x	96.3%

Table 7. CivilComments error set breakdowns.

valid / test splits from Sagawa et al. (2020a).

B.3. MultiNLI

We use the task setup from Sagawa et al. (2020a) on the MultiNLI natural language inference dataset (Williams et al., 2018). Given two sentences, a premise and a hypothesis, the task is to predict whether the hypothesis is *entailed by*, *neutral with*, or *contradicted by* the premise. The spurious attribute a is a binary indicator for when any of the negation words *nobody*, *no*, *never*, or *nothing* appear in the second sentence (the hypothesis), which spuriously correlates with the *contradiction* label. We use the standard train / valid / test splits from Sagawa et al. (2020a).

B.4. CivilComments-WILDS

We use the CivilComments-WILDS dataset from Koh et al. (2021), which is derived from the Jigsaw dataset (Borkan et al., 2019). Given a real online comment, the task is to predict whether the comment is *toxic* or *not toxic*. The spurious attribute a is an 8-dimensional binary vector, where each entry is a binary indicator of whether the following 8 demographic identities are mentioned in the online comment: *male*, *female*, *LGBTQ*, *Christian*, *Muslim*, *other religion*, *Black*, and *White*.

Following Koh et al. (2021), we consider the 16 *potentially overlapping* groups equal to (*identity*, *toxic*) and (*identity*, *not toxic*) for all 8 identities. We use the standard train / valid / test splits from Koh et al. (2021).

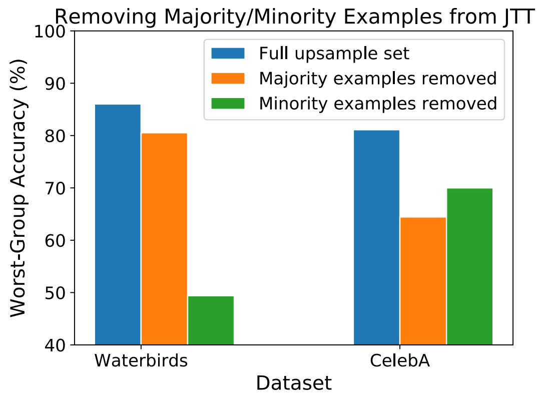

Figure 5. Effect on worst-group accuracy of removing the $y = a$ and $y \neq a$ examples from JTT's error set. Both upsampling $y = a$ examples and upsampling $y \neq a$ examples substantially contribute to improving worst-group accuracy.

C. Additional Experimental Results

C.1. Error set analysis for CivilComments

We include the CivilComments error set analysis in Table 7 for space constraints.

C.2. Additional analysis

Below, we present a series of analyses that involve partitioning the dataset into two groups: groups in which spurious correlation holds with $y = a$, and groups in which spurious correlation does not h old with $9 \neq a$. For this investigation, we focus on Waterbirds and CelebA, where all groups can be clearly partitioned as above because we consider binary classification tasks with binary spurious attributes. In Waterbirds, the $y = a$ groups are waterbirds on water backgrounds and landbirds on land backgrounds; the $y \neq a$ groups are waterbirds on land backgrounds and landbirds on water backgrounds. In CelebA, the $y = a$ groups are blond females and non-blond males; the $y \neq a$ groups are non-blond females and blond males. In contrast, it is unclear how to partition the groups as above in MultiNLI, in which we consider a multi-class classification problem, and in CivilComments-WILDS, in which we have multiple spurious a ttributes corresponding to different demographic identities.

Impact of $y = a$ and $y \neq$ examples in the error set. We first study how worst-group accuracy changes when we remove $y = a$ examples or $y \neq a$ examples from the error set, as summarized in Figure 5. In both datasets, removing either the $y = a$ or $y \neq a$ examples from the error set significantly decreases worst-group accuracy. While this reduction in worst-group accuracy could stem from the fact that we consider a fixed set of hyperparameters including

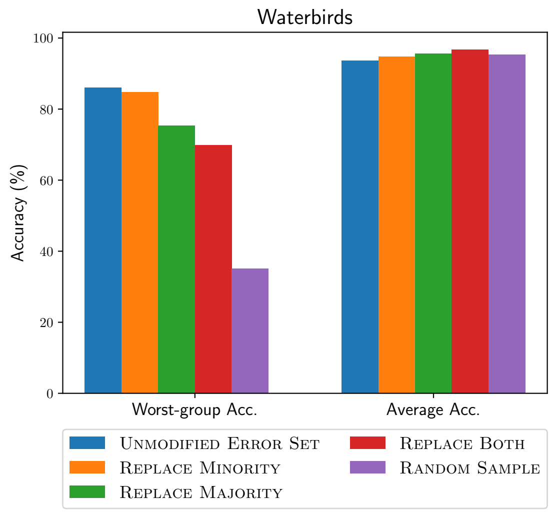

Figure 6. Effect of replacing the $y \neq a$ and $y = a$ examples in JTT's error set with randomly-selected $y \neq a$ and $y = a$ examples on Waterbirds. Worst-group accuracy decreases when replacing either the $y = a$ or $y \neq a$ examples, suggesting that JTT successfully automatically identifies informative examples that improve worst-group accuracy when upsampled.

the upweight factor (which was tuned for JTT with the full error set), it is possible that both $y = a$ and $y \neq a$ examples contribute to the improvement in worst-group accuracy. In particular, because both datasets have substantial label imbalance, it is expected that upweighting groups from rare labels is important to perform well on all groups, and in fact, groups with $y \neq a$ and with rare y are upweighted as discussed in Section 5.3.

Next, we explore if the particular $y = a$ or $y \neq a$ examples that JTT upsamples is important, or if upsampling *any* collection of examples in these groups yields high worst-group accuracy. To do this, we study how average and worst-group accuracies change when we replace examples in JTT's error set with randomly selected examples from specific groups. Specifically, we study what happens when we upsample the following four variants of JTT's error set:

- REPLACE $y \neq a$: We replace the $y \neq a$ examples in the error set with an equal number of randomly selected $y \neq a$ examples, leaving the $y = a$ examples in the error set uncha nged.
- REPLACE $y = a$: We replace the $y = a$ examples in the error set with an equal number of randomly selected $y = a$ examples, leaving the $y \neq a$ examples in the error set unchanged.
- REPLACE BOTH: We replace both the $y \neq a$ examples in the error set with an equal number of randomly selected $y \neq a$ examples, and the $y = a$ examples in the error set with an e qual number of randomly selected $y = a$ examples.
- RANDOM SAMPLE: We replace all examples in the error set with an equal number of randomly selected examples. This yields a different fraction of $y \neq a$ examples in the error set compared to REPLACE BOTH.

	Waterbirds		CelebA	
	Avg.	Worst-group	Avg.	Worst-group
UPSAMPLE MINORITY	92.84%	75.86%	93.44%	57.22%
JTT	90.33%	**86.03%**	87.96%	**81.11%**

Table 8. Average and worst-group test accuracies. UPSAMPLE MINORITY, which upsamples $y \neq a$ examples, improves worst-group accuracy.

Figure 6 compares upsampling these variants of the error set with the original unmodified error set (UNMODIFIED ERROR SET). Compared to upsampling the original error set, upsampling REPLACE $y \neq a$ slightly decreases worst-group accuracy and leaves average accuracy unchanged. This suggests that upsampling most $y \neq a$ examples helps improve worst-group accuracy, though the particular $y \neq a$ examples JTT identifies in the error set are still slightly better than rand om. On the other hand, upsampling REPLACE $y = a$ significantly decreases worst-group accuracy, although it slightly improves average accuracy compared to the original error set. This suggests that the particular $y = a$ examples JTT identifies in the error set are important for improving worst-group accuracy. This could be because the label balance within upsampled $y = a$ changes, or for other reasons. Finally, the low worst-group and average accuracies of both REPLACE BOTH and RANDOM SAMPLE show that merely upsampling random $y = a$ and $y \neq a$ examples is insufficient to achieve high worst-group accuracy.

Upsampling $y \neq a$ groups. We present the performance of a simple baseline, in which we upweight $y \neq a$ examples using ground-truth group annotations, in Table 8. While UPSAMPLE MINORITYimproves worst-group error and JTT outperforms UPSAMPLE MINORITY, the baseline is limited in a few ways. First, $y \neq a$ groups are not necessarily groups with the worst accuracies or smallest number of examples, for example due to label imbalance. So while we $y \neq a$ examples are counter-examples to the spurious correlations, it's not necessarily expected that they improve the worst-group performance well. Secondly, in the presence of ground-truth examples, it is possible to reweight each of the groups independently, rather than reweighting $y = a$ and $y \neq a$ groups. Prior work has observed much higher worst-group performance by reweighting the groups than UPSAMPLE MINORITY(Sagawa et al., 2020a).