Exploring Group-Level Signals for Robust Many-Domain Generalization

First Author[1 [0000−1111−2222−3333]], Second Author[2,3 [1111−2222−3333−4444]], and Third Author[3 [2222−−3333−4444−5555]]

[1] Princeton University, Princeton NJ 08544, USA
[2] Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany
lncs@springer.com
http://www.springer.com/gp/computer-science/lncs
[3] ABC Institute, Rupert-Karls-University Heidelberg, Heidelberg, Germany
{abc,lncs}@uni-heidelberg.de

Abstract. Data spanning a vast number of domains poses a significant challenge in building machine learning models that can generalize robustly to out-of-distribution (OOD) data. Traditional domain generalization approaches have primarily focused on scenarios with a limited number of domains, which fails to account for the real-world complexity. This work explores group-wise reweighting strategies tailored for the challenging setting of many-domain generalization, where both the training set and test set contain a vast number of (unseen) domains. Rather than solely relying on group error metrics which is traditionally considered, we investigate leveraging diverse group features such as label entropy, representation statistics, and gradient properties to determine the relative importance of groups during reweighting. Across multiple datasets, our results demonstrate substantial improvements in worst-group and tail performance metrics by selectively upweighting groups based on features other than error alone. To integrate multiple group features jointly, we adopt a learning-to-rank framework. While the learning-to-rank approach achieves significant gains over empirical risk minimization, the difficulties in outperforming individual features highlight the inherent challenges of achieving transferable robustness amid the differing group characteristics and distributions across datasets. This study provides an initial attempt to develop robust models for the most challenging many-domain generalization setting arising in real-world data.

Keywords: Many Domain Generalization · Group Features · Group Ranking and Reweighting

1 Introduction and Related Work

In the rapidly evolving field of machine learning, the phenomenon of data originating from distinct groups or domains is increasingly recognized. Domain Generalization (DG), as articulated by [30], precludes access to test data during training, presenting a formidable challenge in ensuring models generalize well

across unseen or out-of-distribution (OOD) domains. Our work is situated within the DG framework while delving into the nuanced landscape of DG within the context of many-domain scenarios. Notably, traditional DG approaches have primarily focused on scenarios involving a limited number of domains. However, real-world applications often feature a vast array of domains, ranging from hundreds to millions, as exemplified by users in review systems [12], patients [32] in healthcare settings, or IoT sensors deployed across a city to collect air quality data [16]. This discrepancy between theoretical assumptions and practical realities necessitates a shift toward addressing the complexities of many-domain scenarios. In this work, We predominantly explore within text classifications but extend its application to the novel setting with many domains as described in §2, significantly expanding the domain landscape beyond prior endeavors.

Consider the task of rating classification in an online user review system [24,20], where new users with distinct demographics, purchasing habits, and product preferences join daily. These diverse characteristics influence their rating patterns, making categorizing them into only one or a few domains impractical. Instead, one user should be treated as one domain. In contrast to traditional methods that limit to a handful of domains, our study encompasses datasets with a vast number of training domains, akin to having data from a large, diverse user base. Additionally, the emergence of many new users unseen during training also presents a challenge. These users may exhibit diverse rating behaviors distinctly different from the training set, influenced by temporal shifts or personal biases. If only one or a few unseen test domains are present, such a difference is more easily captured. For instance, a widespread shopping promotion during the test time may homogenize users' rating behavior, making DG easier compared to modeling daily individual user behavior. Our work thus also highlights the setting of limited access to numerous unseen users in the test time to approximate real-world applications. In this setting, following [12], we prioritize tail performance metrics, such as the worst domain's performance or the 10th percentile performance, over average metrics. This ensures the model's robustness for the most challenging user cases, such as those with highly specific preferences or inconsistent rating patterns. To navigate this complexity, we may interchangeably use the concept of 'group' with 'domain' under this challenging setting to emphasize its abundance. We believe that using 'group' to describe a domain in the many-domain setting more closely aligns with the nature of data, which is particularly evident when discussing group features. In the example above, a group contains all reviews from one user and many groups will be in the datasets of interest.

Current DG methods are broadly segmented into three categories, addressing the challenge of generalizing across unseen domains. Firstly, domain matching techniques focus on achieving invariance at different levels: feature invariance [29,21,8,9] aims at aligning feature distributions across domains, while classifier invariance [2,1,33] seeks to identify a classifier that performs consistently across varying domains. Style augmentation methods [23,11] further tailor these methods by modifying data styles or feature statistics to ensure consistent label

predictions across domains. Secondly, reweighting strategies can be categorized into two primary directions: one direction focuses on adjusting weights for individual examples based on errors [17,4,6] or gradients [25], potentially overlooking valuable group-level information when such labels are available. In contrast, the other one emphasizes reweighting based on group features [27,2], yet these methods may overly rely on singular group error metrics. Lastly, meta-learning methods [15,3,32] address DG by simulating train/test domain shifts during episodic training, thereby fostering a model's ability to adapt to new domains with minimal data. This is achieved through optimizing a meta-objective, with approaches like the ensemble of experts [34] exemplifying this strategy.

Our study breaks new ground by treating each user as a distinct domain, introducing a level of diversity across domains that is less explored. This ambitious setting not only tests the limits of existing DG methodologies but also propels us to develop a method that explicitly incorporates domain information, aiming for a nuanced understanding and enhanced generalization across the vast domains. Through this exploration, we contribute to the ongoing dialogue in DG, expanding its applicability and pushing the boundaries of what is achievable within this challenging machine-learning paradigm.

Recognizing the structured nature of data in groups, we proposes a novel avenue for group-wise reweighting by leveraging diverse group features, such as group label entropy, group gradient statistics, and group size, among others introduced in §3.1. Our results demonstrate that reweighting based on the ranking of features other than group error, which has been the focus of past work, yields significantly better or comparable performance across datasets. Features like group label entropy and group representation statistics can achieve 10%+ improvements in worst-group performance on multiple datasets. To explore the potential of incorporating multiple features jointly for group-wise reweighting, we adopt a Learning to Rank (LTR) framework to learn the group importance from these features. The results show that while using multiple features together via LTR can improve upon Empirical Risk Minimization (ERM) significantly, it is not necessarily performing better than using each feature individually. We hypothesize that this is primarily due to the varying distributions of different features across datasets and their differing impacts on group heterogeneity. Further SHAP analysis confirms the varying importance of different features across datasets, highlighting the challenges in achieving transferable robustness.

In summary, our work embarks on an exploration of group-wise reweighting in the context of domain generalization, with a special focus on many-domain scenarios. By integrating diverse group features into a coherent reweighting strategy, we pave the way for more robust and generalizable machine learning models capable of navigating the complexities of real-world data landscapes.

2 Problem Formulation

While many methods are good at the traditional domain generalization tasks, they are not designed to optimize the most challenging task: **many-domain**

generalization problem [13]. The many-domain generalization problem requires these four conditions:

(1) The training set includes numerous domains (usually ≥ 100), instead of only a handful of domains;

(2a) The test set also has numerous domains, instead of only one test domain in most domain generalization tasks;

(2b) As a result of (2a), from the fairness aspect, the focus extends beyond average performance to include tail performance, such as the worst domain's performance and the 10th percentile performance;

(3) The test domains are Out Of Distribution (OOD) unseen domains, instead of the same domains as training as in the subpopulation shift setting.

Now, we formally define our **many-domain generalization problem**: We consider the standard supervised learning setup of classifying an input $x \in \mathcal{X}$ as a label $y \in \mathcal{Y}$. The training data comprises groups from a set $\mathcal{G}_{train}$, where each group $g \in \mathcal{G}_{train}$ consists of n_g data points from a probability distribution $P_g(\mathcal{X}, \mathcal{Y})$. Similarly, in the test set $\mathcal{D}_{test}$, there are another groups set $\mathcal{G}_{test}$. The groups in $\mathcal{G}_{test}$ are new to the $\mathcal{G}_{train}$, i.e., $\mathcal{G}_{test} \cap \mathcal{G}_{train} = \emptyset$. We want to emphasize that the number of groups in the training set $\mathcal{D}_{train}$ and in the test set $\mathcal{D}_{test}$ i.e., $|\mathcal{G}_{train}|$ and $|\mathcal{G}_{train}|$ are large, at least hundreds. The goal is to learn a model $f_\theta : \mathcal{X} \times \mathcal{G} \to \mathcal{Y}$ parameterized by $\theta \in \Theta$ so that the optimal model could have satisfying test average and the tail performances.

Many real-world applications fall into the many-domain generalization problem, which is very meaningful despite being less focused in academia due to its difficulty. For example, in the healthcare context, grouping the data from one patient allows for personalized analysis and treatment planning for each individual. This leads to many domains in the training as well as many unseen test domains after the deployment. Under the **many-domain generalization problem**, we want a model with strong generalizability towards many challenging unseen cases.

The similar but less restrictive problem settings to the **many-domain generalization problem** would be **Common Domain Generalization** and **subpopulation shift**. We summarize the characteristics of the three settings in Table 1. Although these three settings all fall into the broader concept of domain generalization, here the common domain generalization we refer is the traditional setting that generalizes over a handful of domains for an invariant rule or subspace and evaluates one unseen domain. In our setting, there would be many more domains, at least hundreds in both training and test. Common Domain Generalization problems satisfy condition (3), violate condition (2b), and are loose on conditions (1) and (2a). Many existing algorithms are designed

based on common domain generalization (implicitly assuming a small number in the training due to the limitation of the benchmark datasets). For example, the matching methods [29,21,2] are always designed to find an invariant subspace by penalizing the domain differences such as in the mean, variance, or the classifiers' last layer's parameters. However, Scaling up in the training domains raises two problems: first, it is very time-consuming to do the matching for every pair of groups. Second, even if the matching process could be done within each batch, it is not efficient and effective because minority groups are less likely to be present than the majority groups. Moreover, most algorithms for domain generalization assume the test distribution would fall into the convex hull of the training domain distributions, which is not necessary in our setting.

Subpopulation shift problems violate conditions (2b) and (3), and are loose on conditions (1) and (2a). In subpopulation shift problems, the training domains and test domains are completely the same. Although during the evaluation, the worst test domain performance is considered, with the assumption that for each domain the distribution stays the same, the problem simplifies to generalizing well across a limited handful of seen domains.

Condition	(1)Numerous groups in training	(2a)Numerous groups in test	(2b)Care about the tail performance	(3)OOD unseen test groups
Common Domain Generalization	loose, usually handful	loose, usually one	×	✓
Sub-population Shift	loose, usually handful	loose, usually handful	✓	×
Ours	✓	✓	✓	✓

Table 1: Summarization on the Problem Settings. The citations followed by the setting name are the algorithms initially designed for the setting.

3 Methods

As mentioned above, previous methods have all focused on group error or loss. This is reasonable because improving group performance is the ultimate goal, but focusing solely on group error during the training stage does not necessarily lead to better OOD group performance, especially in the setting of many-domain generalization. Therefore, we decided to explore some other features that may help improve the worst group performance in the most challenging setting. In the next subsection, we will go over the features we initially consider including group error, group representation standard deviation, group gradient mean, group gradient standard deviation, group size, group label entropy, and group label&error mutual information. Then, we will talk about how we rank and reweight groups

based on individual features in §3.2 and how we adopt a learning-to-rank (LTR) approach to use multiple features together to rank and reweight groups in §3.3.

3.1 Different Group Features for Many-Domain Generalization

Following §2, assume a group g from the training set $\mathcal{D}_{train}$ contains a set of instances $S_g = [(x_1, y_1), (x_2, y_2), ..., (x_{n_g}, y_{n_g})]$. A model f is trained on $\mathcal{D}$ and gives the prediction labels to instances of S_g as $[\hat{y}_1, \hat{y}_2, ..., \hat{y}_{n_g}]$. Additionally, f learns a representation function $\phi : x \mapsto \mathbb{R}^M$ for x_i to predict its label where M is the dimension of the representation.

Group Error: Group error is the most commonly considered feature in domain generalization. Poorly performed groups in the training time should be weighted higher. Formally, it is defined as $\varepsilon_g = \sum_{i=1}^{n_g} \mathbb{1}\{\hat{y}_i \neq y_i\}$.

Group Representation Standard Deviation: Inspired by [14], let $d_i^\phi = ||\phi(x_i), \bar{\phi}_g||_2$ where $\bar{\phi}_g = \frac{1}{n_g} \sum_{i=1}^{n_g} \phi(x_i)$ is the average representation of all instances of the group. It means the distance of an instance representation to the central representation of the group. Then the group representation standard deviation can be expressed as $\sigma_g^\phi = \sqrt{\frac{1}{n_g} \sum_{i=1}^{n_g} (d_i^\phi - \bar{d}_g^\phi)^2}$ where $\bar{d}_g^\phi = \frac{1}{n_g} \sum_{i=1}^{n_g} d_i^\phi$ is the average distance. σ_g^ϕ reflects the homogeneity of a group. A group is likely to have either extremely high or low accuracy if the instances of the group are similar to each other.

Group Gradient Mean and Group Gradient Standard Deviation: Recent work [25] shows that the gradient is informative about the group robustness. Here, we take the gradient of the loss of each instance to the last prediction layer $W \in \mathbb{R}^{M \times T}$ of the model f: $\nabla l_i = flatten(\frac{\partial l(f(x_i), y_i)}{\partial W})$ where T is the number of distinct output labels and l is the loss function. The original gradient is also in $\mathbb{R}^{M \times T}$ so we flatten the gradient array to $\mathbb{R}^{MT \times 1}$. Again we compute the distance of each gradient of a group to the group's mean gradient: $d_i^{\nabla l} = ||\nabla l_i, \nabla \bar{l}_g||_2$ where $\nabla \bar{l}_g = \frac{1}{n_g} \sum_{i=1}^{n_g} \nabla l_i$ is the group gradient mean. Then the Group Gradient Standard Deviation $\sigma_g^{\nabla l} = \sqrt{\frac{1}{n_g} \sum_{i=1}^{n_g} (d_i^{\nabla l} - \bar{d}_g^{\nabla l})^2}$ where $\bar{d}_g^{\nabla l} = \frac{1}{n_g} \sum_{i=1}^{n_g} d_i^{\nabla l}$ is the average distance. As the gradient mean is high dimensional, we conduct a PCA on the raw gradient mean $\nabla \bar{l}_g$ to a two-dimensional vector $\in \mathbb{R}^2$. We use the absolute value of the two dimensions as our final gradient mean feature. The absolute values reflect the relative position of the average gradient of a group to other groups and allow a flexible transfer across datasets.

Group Size: The most natural way to improve group robustness is by balancing the size of groups in a dataset [26]. The group size should be informative of the importance of groups in the training time. Formally, it is the total number of instances n_g as defined above.

Group Label Entropy: The group label entropy $H_g(Y) = -\mathbb{E}[-log_2(P_g(Y))]$ where $P_g(Y)$ is the label distribution within the group g. We expect the group label entropy to reflect the spurious correlation between the group features and labels, which makes the model suffer from distribution shifts [22,7]. In the past

study of spurious correlation, the group is usually defined with binary and known or even synthetic spurious features [26,17,27]. However, this is almost impossible in many-domain settings as the spurious correlations may be a lot more and high-dimensional. We hope the group label entropy is a simple metric to reflect the spuriosity of a group. The group with higher label entropy may suffer less from spurious correlations.

Group Label&Error Mutual Information: Even though the label entropy reflects the spurious correlation, it does not necessarily lead to the poor performance of a group as the model may be learned to disentangle the group features and labels independently. We hope to give more importance to the groups that not only have strong spurious prior but poor performances in training as well. Here, we use the mutual information of label and error of a group g: $I_g(Y;E) = H_g(Y) - H_g(Y|E)$ where $H_g(Y|E)$ is the conditional entropy of label given error. The conditional entropy equals $H_g(Y,E) - H_g(E)$ where $H_g(Y,E)$ is the entropy of the joint distribution of label and error of g $((y_i,\ y_i == \hat{y}_i))$. and $H_g(E)$ is the entropy of error distribution of g. $I_g(Y;E)$ measures whether the error and labels are independent. Presumably, if g has a high $I_g(Y;E)$, the model may be unfair to this group, and therefore one may upweight this group.

3.2 Ranking and Reweighting Based on Individual Features

To uniformly compare all features, inspired by the recent work [17,26] for group distributional robustness, we propose a method that performs iterative upweighting of training examples. Our novel approach iteratively ranks the groups by their features and then upweights poorly performing groups. The key idea is to upweight training samples from the groups with the higher ranking of feature values and assign them differential importance commensurate with their ranking.

The direction of the ranking of groups according to a feature is determined by its correlation with error. A group with a higher error is deemed more important for reweighting. Therefore, if a feature positively correlates with error, groups with higher values in this feature are ranked higher.

To map the rankings into actual weights for each group, we use a log reverse rank weighting scheme inspired by the Discounted Cumulative Gain [10]. DCG uses a logarithmic discount on the ranking of items, highlighting the importance of top-ranked items while diminishing it of lower-ranked ones. Similarly, given a group g that ranks at the i^{th} percentile according to the learned rank, its corresponding weight is determined by:

$$w_g = max(\frac{\log C}{\log i}, 1) \tag{1}$$

where C represents a predefined cutoff threshold. The cutoff ensures that any group ranked above the C^{th} percentile is assigned a weight of 1. This weighting mechanism provides a gradual and logarithmic discount in weight as the group is ranked higher, reflecting a group's relative importance. The introduction of the cutoff threshold C allows for the saturation of weights beyond a certain rank, ensuring that lower-ranked groups receive equal and default weighting.

The group size and group label entropy do not change with the training so the reweighting is only applied once. The ranking of groups according to other features will be reevaluated after each epoch of training and reweighted accordingly. Regardless of the features, the weight of groups will always be decided by the equation 1 from the ranking. The first epoch will follow standard Empirical Risk Minimization (ERM) without reweighting. We show proof of convergence of our reweighting algorithm according to the ranking of group error in §9.

3.3 Combining Features Together: A LTR Approach

The combination of individual features, which independently improve the performance if possible, may enhance overall performance more. Multiple features cannot be ranked directly in the same way as a single feature. Instead, learning to rank (LTR) can learn to rank multiple features. In the traditional learning-to-rank task, the input comprises a set of queries Q, a set of documents D, and a set of relevance labels R associated with each query-document pair. Specifically, for each query $q \in Q$ and document $d \in D$, there exists a relevance label $r \in R$ indicating the degree of relevance of the document to the query. The objective is to learn a ranking function $f : Q \times D \to \mathbb{R}$ that, given a new query and a set of documents, produces a ranking score for each document, with higher scores denoting greater relevance to the query.

Let $\mathcal{G}$ represent a set of groups in a dataset $\mathcal{D}$, where each group $g \in \mathcal{G}$ is analogous to a document D in the traditional learning-to-ranking scenario. A 'query' Q in this context is an assignment of groups: $A : \mathcal{G} \mapsto \{\mathcal{D}_{train}, \mathcal{D}_{test}\}$. Each group is assigned to either the training set or the test set. For a set of groups $\mathcal{G}$ under assignment A, each group is associated with a relevance label r which denotes group importance. We will have better tail performance in the test dataset if we reweight groups by some function of the group importance $h(r)$ during the training. The training objective will be $\mathcal{L}_{weighted} = \sum_{g \in \mathcal{G}} h(r)\mathcal{L}(g)$ where $\mathcal{L}(g)$ is the loss of group g. The goal is to optimize test performance by emphasizing tail groups and reducing the reliance on top-performed ones in the training process.

Given a source dataset $\mathcal{D}$, our objective is to robustly learn an LTR model that maps groups to a ranking of relevance r under an assignment A and then apply the LTR model to reweight many groups in other datasets for better generalizations under those datasets. Note that this is different from training a model on one dataset and hoping to generalize to another dataset. We expect an LTR model to learn to rank groups by capturing useful group features jointly that can be also used for ranking groups in new datasets. To achieve that, we need to construct the training data for the LTR from the source dataset including features of the groups and their associated rankings. In the following sections, we will first introduce the features we designed for groups, the pseudo-ranking of groups for training, and the training procedure for LTR once we obtain the features and rankings.

Let $\mathcal{D}$ be our source dataset where each instance x_i is associated with a specific group g_i. Our goal is to ensure that there's no overlap of groups between

training and testing datasets while introducing distribution shifts for evaluation purposes.

Step 1: Group-wise Splitting. The original dataset $\mathcal{D}$ is group-wise split into a training set $\mathcal{D}_{train}$ and a testing set $\mathcal{D}_{test}$ so that no group is shared between $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$. As mentioned below, we used existing data that already completed the data split and ensured the distribution shift from training to test, as there are large performance gaps between in-domain and out-of-domain.

Step 2: Generating Training Subsets. To create K distinct distribution shifts from the training to test, we sample K different subsets of groups from $\mathcal{D}_{train}$. Let the set of all groups in $\mathcal{D}_{train}$ be $\mathcal{G}_{train}$ and all groups in $\mathcal{D}_{test}$ as $\mathcal{G}_{test}$. For each $k \in \{1, 2, ..., K\}$, we form a subset of groups $\mathcal{G}^{(k)}_{train}$ by randomly sampling from $\mathcal{G}_{train}$ without replacement.

Each training subset $\mathcal{D}^{(k)}_{train}$ then includes instances of all $g_i \in \mathcal{G}^{(k)}_{train}$. Hence, each $\mathcal{D}^{(k)}_{train}$ is a distinct collection of groups, ensuring different distribution shifts relative to $\mathcal{D}_{test}$. Additionally, we ensure that the number of groups in each subset $\mathcal{G}^{(k)}_{train}$ is consistent across all K subsets. Here, one pair of $\mathcal{G}^{(k)}_{train}$ and $\mathcal{G}_{test}$ corresponds to one assignment of groups $A^{(k)}$ in the set of groups $\mathcal{G}^{(k)} = \mathcal{G}^{(k)}_{train} \cup \mathcal{G}_{test}$. For all assignments, we keep the test data the same ($\mathcal{G}_{test}$) as the goal is to create enough different distribution shifts from training to test that bring different LTR training samples.

Step 3: Getting LTR training features. For each subsampled training set $\mathcal{D}^{(k)}_{train}$, we first train a model f_k on $\mathcal{D}^{(k)}_{train}$. Then, for each group g in $\mathcal{D}^{(k)}_{train}$, evaluate f_k to obtain the seven features for LTR training mentioned in §3.1 $x^{(k)}_g = (\varepsilon^{(k)}_g, \sigma^{\phi(k)}_g, \nabla\bar{l}^{(k)}_g, \sigma^{\nabla l(k)}_g, n^{(k)}_g, H^{(k)}_g(Y), I^{(k)}_g(Y;E))$. Let $X^{(k)} = \{x^{(k)}_g | g \in \mathcal{G}^{(k)}_{train}\}$

Step 4: Get LTR Pseudo Rank. Our objective in this step is to estimate the relative importance of each group in the subsampled training sets to build training instances for the LTR model. To this end, we create a pseudo rank of the groups based on their prediction errors when a model is trained on $\mathcal{D}_{test}$:

1. Train a model f_{test} on the test dataset $\mathcal{D}_{test}$ from Step 1.

2. For each subsampled training set $\mathcal{D}^{(k)}_{train}$, evaluate f_{test} on it to obtain group-wise errors.

3. Rank all groups in $\mathcal{D}^{(k)}_{train}$ based on their group errors, with groups having higher errors ranked higher. This ranking produces a pseudo rank, $R^{(k)}$, which serves as an approximation of the group's importance during training.

Given a source dataset $\mathcal{D}_{source}$, a bunch of subsampled sets of groups $\mathcal{G}^{(k)}$ obtained via the steps mentioned in §3.3, group assignments $A^{(k)}$, surrogate rankings $R^{(k)}$ and training features $X^{(k)}$, we can train an LTR model for many-domain generalization the same as traditional LTR model for information retrieval. We use ListNet [5] as our LTR model. ListNet is one of the pioneering approaches in the learning-to-rank which first considers the relation across documents by using list-wise loss instead of pair-wise loss. Once we train the LTR model, we will apply it in the training of a new dataset $\mathcal{D}_{target}$ to rank the importance of new groups in the dataset. We again reweight new groups according

to the learned ranking using equation 1 as discussed in §3.2. During the training of the new dataset, the ranking of groups will be reevaluated from the LTR model after each epoch and then reweighted accordingly.

4 Experiments and Results

4.1 Datasets

We use three real-world text classification datasets that have been commonly used in prior group robustness literature: AMAZON-WILDS [12], Yelp Open Dataset[4], CivilComments [12], as well as a variation of the Civilcomments dataset. The prediction task for Amazon and yelp datasets is to classify the review text into its corresponding 1-to-5-star rating. Each review is associated with a group, which corresponds to all reviews written by the same *user*. Each dataset has an in-distribution (ID) training set and out-of-distribution (OOD) validation and test sets. The OOD validation and test sets comprise reviews from disjoint sets of users (groups). For more preprocessing and descriptive dataset details, please refer to the Appendix 7.

We rebuild the CivilComments dataeset [12] into two variations which also follow the most challenging many-domain generalization setting we mentioned above. In the original dataset, each instance is associated with an 8-dimensional binary vector where each component corresponds to whether the comment mentions one of the 8 demographic identities: male, female, LGBTQ, Christian, Muslim, other religions, Black, and White. Then the author reports the data performance under each demographic. Here, in the first variation (CivilComments1), we rebuild the data where instances were grouped by their unique binary vector combinations. Thus, comments with identical demographic mentions were categorized together. Ideally, there could be 2^8=256 groups but we filtered out the groups with less than 10 instances. We get 137 groups and randomly select 22 groups as the validation data, 20 groups as the test data, and the remaining 95 groups as training data.

For all the datasets we see so far, there is no instance overlap between groups. In real-world contexts, individual identities are multifaceted and seldom exist in isolation. Here, in the second variation (CivilComments2), we rebuild the data where instances sharing mentions of at least three common identities are considered to belong to the same group. As a result, groups can have overlaps. For instance, a group mentioning Male, LGBTQ, and Christian would have overlaps with a group mentioning Male, LGBTQ, and White. The overlap instances will mention Male, LGBTQ, White, and Christian. This overlapping nature allows for a more fluid understanding of the group features we designed above. Ideally, there will be $\binom{8}{3}$=56 groups but we filtered out groups with less than 10 instances. We get 55 groups in total and randomly select 30 as training groups and 25 as test groups. As there will be overlaps in the training groups and test groups, we don't use it as the target evaluation dataset as it falls out of the many-domain

[4] https://www.yelp.com/dataset/

generalization setting. We will merely use it as a source dataset for LTR model training.

We use Yelp, CivilComments1 and CivilComments2 as the source dataset for LTR training and then apply it to the target dataset Yelp, Civilcomments1, and Amazon for the test of learned LTR models from the source. We don't use Amazon for LTR training due to the relatively large size of the dataset.

4.2 Experiment Setup

We compare our ranking and reweighting methods against baseline methods Empirical Risk Minimization **(ERM)**, **Group DRO** [26], and **JTT** [17]. ERM is the standard training method that trains models to minimize the average training loss. Group DRO uses distributionally robust optimization to explicitly minimize the loss on the worst-case domain (or group) during training. JTT has shown superior performance over Group DRO and its variations on several challenging benchmark applications. JTT involves a two-stage training approach which first trains a standard ERM model for several epochs and then trains a second model that upweights the training examples that the first model has misclassified. We finetuned the base-uncased DistilBERT [28] model as our base learner for all task training methods. The number of epochs is 5 with early stopping; the batch size is 16; the learning rate is 1×10^{-5} for AdamW optimizer; L2-regularization strength is 0.01. For the ranking and reweighting algorithm, we use each feature mentioned in §3.1 to rank except gradient mean as it is multi-dimensional. We performed a grid search to tune the cutoff hyperparameter $C \in [10, 20, 50, 100]$ for reweighting. For **JTT**, the number of epoch in the first stage $T \in \{0, 1, 2\}$ and the reweighting factor $\lambda \in \{2, 3, 5\}$. For **Group DRO** we fixed the step size as 0.01 following the best practice reported in [13].

For the neural ranking model ListNet, we simplify it to just one linear layer for better interpretability features. The learning rate is 1×10^{-2} for AdamW optimizer [18] with a weight decay of 1×10^{-2}. The batch size is 128 and the number of epochs is 100. To get the training features for ListNet training, we first finetuned the entire source dataset $\mathcal{D}_{source}$ again on a base-uncased Distilbert. Then, we fine-tuned only the last layer of the model for each subsampled training set $\mathcal{D}^{(k)}_{train}$ for one epoch. Other hyperparameters are the same as the model training mentioned above. The number of training subsets for all three datasets is 500, which means we have 500 ranking instances for the training of ListNet for each dataset. All the experiments are run on the NVIDIA GEFORCE RTX 2080 Ti using the PyTorch Framework.

4.3 Results Using Individual Features

The reweighting results according to the ranking of each feature are shown in Table 2. The results are in the format of (average accuracy/10th percentile accuracy/worst group accuracy). It can be observed that the performance of the worst group after reweighting using different features is generally better than that of

Datasets	Error	Grad_std	LE	Rep_std
Yelp	62.5/52.0/21.0	62.8/52.0/19.0	61.9/51.0/23.0	61.9/52.0/**28.0**
CivilComments1	77.9/64.3/53.8	79.6/70.2/53.8	**80.5**/72.1/**69.2**	79.6/70.8/53.8
Amazon	70.2/54.7/**17.3**	70.5/53.3/13.3	71.2/54.7/13.3	71.7/53.3/8.0
MI	**Group Size**	**ERM**	**JTT**	**GroupDRO**
62.8/51.2/23.0	63.0/52.0/23.0	63.0/52.0/18.0	61.7/51.0/19.0	59.2/49.0/27.0
79.9/66.5/61.5	79.3/72.7/53.8	80.1/72.7/53.8	78.4/57.9/53.8	79.3/69.0/61.1
71.4/54.7/12.0	70.1/53.3/16.0	**71.9**/53.3/12.0	71.6/53.3/9.3	70.0/53.3/8.0

Table 2: Performance of ranking and reweighting according to each feature. The results are in the format of (average accuracy/10th percentile accuracy/worst group accuracy). The best performance of each metric is highlighted in bold, and the tied best performance is emphasized with an underline.

ERM on all three datasets. Specifically, using group representation standard deviation on Yelp resulted in a 10% worst group improvement, using group label entropy (LE) on CivilComments led to a 15.4% enhancement, and using group error on Amazon achieved a 5.3% increase. ERM generally performs well in average group performance, as expected from its nature to optimize average training loss. JTT doesn't show improvements over ERM or even harm the performance in many-domain settings. While GroupDRO improves worst-group performance over ERM in Yelp and Civilcomments1 datasets, it significantly harms average performance. We can see that other features apart from the traditionally focused group error may also yield equivalent or better performance.

4.4 Results Using LTR

Datasets	LTR (Yelp)	LTR (C1)	LTR (C2)	ERM	JTT	GroupDRO
Yelp	61.8/51.0/**28.0**	62.5/53.0/19.0	**63.1**/53.0/23.0	63.0/52.0/18.0	61.7/51.0/19.0	59.2/49.0/27.0
CivilComments1	79.8/70.2/**61.5**	**80.4**/72.3/53.8	79.5/69.7/58.3	80.1/**72.7**/53.8	78.4/57.9/53.8	79.3/69.0/61.1
Amazon	70.7/**54.7**/13.3	70.8/53.3/13.3	70.6/53.3/12.0	**71.9**/53.3/12.0	71.6/53.3/9.3	70.0/53.3/8.0

Table 3: Performance of reweighting groups using LTR. The ones in parentheses are the source datasets for training LTR. The results are in the format of (average accuracy/10th percentile accuracy/worst group accuracy). The best performance of each metric is highlighted in bold, and the tied best performance is emphasized with an underline.

The results of LTR methods and other baseline methods are shown in Table 3. Our methods are denoted as 'LTR (source dataset)'. The source dataset is used to train the LTR model. All LTR methods can consistently outperform or be comparable with ERM across all three datasets. LTR(Yelp) shows the best results by improving 10% on Yelp and 7.3% on CivilComments. We hope that combining multiple features will result in improved performance. However, it does not significantly outperform the use of a single feature; in some cases, employing multiple features is even less effective than using just one.

We believe there are two possible reasons for this: 1. As can be seen from Table 2, for the same dataset, different features contribute differently to performance. For example, in the case of the CivilComments dataset, the improvement of the worst group performance by gradient standard deviation is far less than that of group label entropy, hence their combined use is not as effective as using label entropy alone; 2. Similarly, according to Table 2, the contribution of the same feature varies across different datasets. A LTR method requires a training source dataset, where the distribution of these features may likely differ from that of the test dataset. Therefore, an LTR model trained on the source dataset might not transfer effectively to the test dataset. Similar conclusions can also be reflected in the SHAP analysis in §5. How to effectively use these individually useful features represents a potential direction for future research.

5 Analysis of Feature Importance by SHAP

We use SHapley Additive exPlanations (SHAP) [19] to quantify the importance of each of the features for predicting rankings. SHAP is a game theoretic approach to explain the output of any machine learning model by providing a comparable measurement of the importance of each feature. By doing so, we aim to validate the selection of predictive features and identify the importance of features across different source datasets.

For each source dataset used to train the LTR model, we randomly subsampled 100 instances to initialize a SHAP Kernel Explainer. We applied the explainer with the trained LTR model on another randomly sampled 2000 instances to compute SHAP values of each feature.

The SHAP result when the Yelp dataset is used as the source training dataset for LTR model is shown Figure 1. The results for the other two datasets are shown in Appendix 8. In the example of Yelp, we notice that all features positively contribute to the prediction of the ranking which again showcases the validity of using features other than group error for reweighting. The group label entropy is the most important feature in all three datasets which aligns with the performance in Table 2. As we mentioned in §3.1, the label entropy reflects the spurious correlation of group features with the labels. This implies that spurious correlations are still the main challenges in the many-domain settings of the distribution shift. The significance of both group size and error features validates traditional methods to tackle distribution shifts like upsampling and boosting. When comparing across different datasets, it is evident that the importance of

features varies with each dataset. For instance, the contribution of error is the lowest in CivilComments1, whereas it is comparatively higher in Yelp. This could help to explain why the same feature shows different levels of contribution to performance across various datasets as seen in Table 2, and why it is challenging to transfer an LTR model trained on one dataset to another test dataset when integrating all features as shown in Table 3.

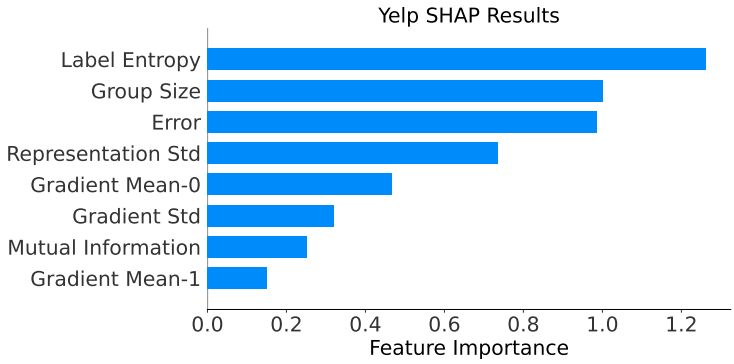


Fig. 1: SHAP feature importance of the Yelp dataset. All features positively contribute to the prediction of the ranking of groups. The label entropy is the most significant feature in all datasets which highlights the challenge of spurious features in the many-domain settings.

6 Conclusion

In this work, we have explored group-wise reweighting strategies for robust and generalizable machine learning models on the challenging yet practical setting of many-domain generalization. We propose using alternative group features beyond the traditionally utilized group error to determine the relative importance of groups for reweighting, and our results demonstrate that features like group label entropy and group representation statistics can achieve over 10% improvements on multiple datasets' worst-group performance compared to standard Empirical Risk Minimization. To further investigate the potential benefits of incorporating multiple predictive group features jointly, we adopted a learning-to-rank framework, but found that using multiple features via learning-to-rank does not necessarily outperform the use of each feature individually, likely due to the varying distributions and differing impacts on group heterogeneity of features across datasets, posing challenges for transferable learning. Our SHAP analysis provides supporting evidence by showcasing the varying significance of features across datasets. We hope our exploration of group features and algorithmic paradigms provides a promising path forward for many-domain

generalization and future work will study how to learn the joint application of multiple features to achieve cross-dataset robustness.

References

1. Ahuja, K., Shanmugam, K., Varshney, K., Dhurandhar, A.: Invariant risk minimization games. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 145–155. PMLR (13–18 Jul 2020), https://proceedings.mlr.press/v119/ahuja20a.html
2. Arjovsky, M., Bottou, L., Gulrajani, I., Lopez-Paz, D.: Invariant risk minimization. arXiv preprint arXiv:1907.02893 (2019)
3. Balaji, Y., Sankaranarayanan, S., Chellappa, R.: Metareg: Towards domain generalization using meta-regularization. Advances in neural information processing systems **31** (2018)
4. Ben-Tal, A., Den Hertog, D., De Waegenaere, A., Melenberg, B., Rennen, G.: Robust solutions of optimization problems affected by uncertain probabilities. Management Science **59**(2), 341–357 (2013)
5. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th international conference on Machine learning. pp. 129–136 (2007)
6. Duchi, J.C., Glynn, P.W., Namkoong, H.: Statistics of robust optimization: A generalized empirical likelihood approach. Mathematics of Operations Research **46**(3), 946–969 (2021)
7. Geirhos, R., Jacobsen, J.H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., Wichmann, F.A.: Shortcut learning in deep neural networks. Nature Machine Intelligence **2**(11), 665–673 (2020)
8. Ghifary, M., Kleijn, W.B., Zhang, M., Balduzzi, D., Li, W.: Deep reconstruction-classification networks for unsupervised domain adaptation. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. pp. 597–613. Springer (2016)
9. Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., Smola, A.: A kernel method for the two-sample-problem. Advances in neural information processing systems **19** (2006)
10. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems (TOIS) **20**(4), 422–446 (2002)
11. Kang, J., Lee, S., Kim, N., Kwak, S.: Style neophile: Constantly seeking novel styles for domain generalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7130–7140 (2022)
12. Koh, P.W., Sagawa, S., Marklund, H., Xie, S.M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R.L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B.A., Haque, I.S., Beery, S., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., Liang, P.: WILDS: A benchmark of in-the-wild distribution shifts. In: International Conference on Machine Learning (ICML) (2021)
13. Koh, P.W., Sagawa, S., Marklund, H., Xie, S.M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R.L., Gao, I., et al.: Wilds: A benchmark of in-the-wild distribution shifts. In: International Conference on Machine Learning. pp. 5637–5664. PMLR (2021)

14. Krueger, D., Caballero, E., Jacobsen, J.H., Zhang, A., Binas, J., Zhang, D., Le Priol, R., Courville, A.: Out-of-distribution generalization via risk extrapolation (rex). In: International Conference on Machine Learning. pp. 5815–5826. PMLR (2021)
15. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.: Learning to generalize: Meta-learning for domain generalization. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
16. Liang, X., Li, S., Zhang, S., Huang, H., Chen, S.X.: Pm2. 5 data reliability, consistency, and air quality assessment in five chinese cities. Journal of Geophysical Research: Atmospheres **121**(17), 10–220 (2016)
17. Liu, E.Z., Haghgoo, B., Chen, A.S., Raghunathan, A., Koh, P.W., Sagawa, S., Liang, P., Finn, C.: Just train twice: Improving group robustness without training group information. In: International Conference on Machine Learning. pp. 6781–6792. PMLR (2021)
18. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
19. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017)
20. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. pp. 142–150. Association for Computational Linguistics, Portland, Oregon, USA (June 2011), http://www.aclweb.org/anthology/P11-1015
21. Muandet, K., Balduzzi, D., Schölkopf, B.: Domain generalization via invariant feature representation. In: Dasgupta, S., McAllester, D. (eds.) Proceedings of the 30th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 28, pp. 10–18. PMLR, Atlanta, Georgia, USA (17–19 Jun 2013), https://proceedings.mlr.press/v28/muandet13.html
22. Nagarajan, V., Andreassen, A., Neyshabur, B.: Understanding the failure modes of out-of-distribution generalization. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=fSTD6NFIW_b
23. Nam, H., Lee, H., Park, J., Yoon, W., Yoo, D.: Reducing domain gap by reducing style bias. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8690–8699 (2021)
24. Ni, J., Li, J., McAuley, J.: Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (2019)
25. Ren, M., Zeng, W., Yang, B., Urtasun, R.: Learning to reweight examples for robust deep learning. In: International conference on machine learning. pp. 4334–4343. PMLR (2018)
26. Sagawa*, S., Koh*, P.W., Hashimoto, T.B., Liang, P.: Distributionally robust neural networks. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=ryxGuJrFvS
27. Sagawa, S., Raghunathan, A., Koh, P.W., Liang, P.: An investigation of why overparameterization exacerbates spurious correlations. In: International Conference on Machine Learning. pp. 8346–8356. PMLR (2020)
28. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019)
29. Sun, B., Feng, J., Saenko, K.: Return of frustratingly easy domain adaptation. In: AAAI (2016)

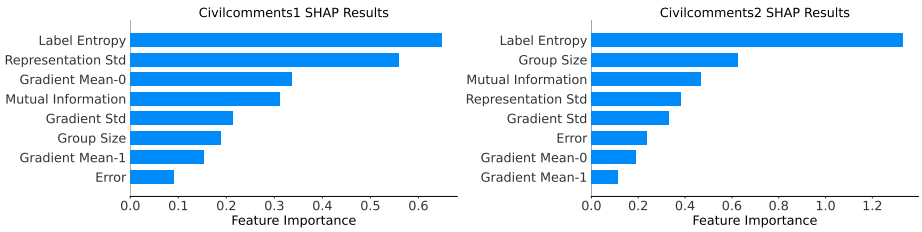
30. Wang, J., Lan, C., Liu, C., Ouyang, Y., Qin, T., Lu, W., Chen, Y., Zeng, W., Yu, P.: Generalizing to unseen domains: A survey on domain generalization. IEEE Transactions on Knowledge and Data Engineering (2022)
31. Xiao, R., Ge, Y., Jiang, R., Yan, Y.: A unified framework for rank-based loss minimization. arXiv preprint arXiv:2310.17237 (2023)
32. Yang, C., Westover, M.B., Sun, J.: Manydg: Many-domain generalization for healthcare applications. In: The 11th International Conference on Learning Representations, ICLR 2023 (2023)
33. Yao, H., Wang, Y., Li, S., Zhang, L., Liang, W., Zou, J., Finn, C.: Improving out-of-distribution robustness via selective augmentation. In: International Conference on Machine Learning. pp. 25407–25437. PMLR (2022)
34. Zhou, K., Yang, Y., Qiao, Y., Xiang, T.: Domain adaptive ensemble learning. IEEE Transactions on Image Processing **30**, 8008–8018 (2021)

Appendix

7 Datasets

AMAZON-WILDS: Collected as part of the WILDS dataset suite [12], the Amazon-WILDS dataset involves predicting star ratings from users' reviews of Amazon products. The training set has 245,502 reviews from 1252 users (at least 75 reviews per user). The ID validation set consists of 46,950 reviews from 626 of the 1252 users in the training set. The ID test set is the same size as the ID validation set and contains reviews from the remaining 626 users from the training dataset. Finally, the OOD validation and the OOD test sets each have 100,050 reviews (75 reviews per user) from 1,334 new users.
Yelp Business Reviews: WILDS dataset suite [12] contains a modified version of the Yelp Open Dataset; however, there's no accuracy drop from their ID set to OOD set. Thus, we modify it by clustering at the user level in a similar fashion as we did for the IMDB dataset. We set k=6 and select the two farthest clusters as the OOD and ID sets to have a significant out-of-distribution performance drop. The training set comprises 64,931 reviews from 500 users. The ID validation and test sets consist of 20,070 and 21,083 reviews from 333 out of the 666 users in the training set, respectively. Finally, the OOD validation and test sets include 52,200 and 52,300 reviews from 522 and 523 unseen users, respectively.

8 SHAP results



(a) SHAP feature importance of the Civilcomments1 dataset.

(b) SHAP feature importance of the Civilcomments2 dataset.

Fig. 2: Comparative SHAP feature importance for Civilcomments datasets.

9 Convergence of the Algorithm

We can show the convergence of our reweighting method when ranking by group loss by drawing upon the methodology used by [31], who establish convergence

for rank-based loss minimization. Our objective is compatible with their framework by creating a sequence of ordered groups. The objective function is then defined as a weighted sum of these group losses, where the weights are calculated as the product of group size and reweighting factors assigned based on group rank.

For our proposed algorithm, the objective is

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} w_i n_{[i]} Y_{[i]},$$

where the $\boldsymbol{\theta}$ denotes the model's parameters, and N is the total number of groups in the training set. Y_i represents the average training loss of the i^{th} group for $i \in \{1, ..., n\}$. Let $Y_{[1]} \leq ... \leq Y_{[n]}$ be the order statistics. w_i is the weight corresponding to the i^{th} smallest group and $n_{[i]}$ is the number of instances within group $g_{[i]}$.

When group sizes are the same $n_{[1]} = n_{[2]} = \ldots = n_{[N]} = \mathrm{c}$, we have

$$\mathcal{L}(\boldsymbol{\theta}) = c \left(\sum_{i=1}^{N} w_i Y_{[i]} \right) \tag{2}$$

whose convergence has been proved by [31] under the alternating direction multiplier method (ADMM) optimization.

However, when the group sizes differ, the proof of convergence cannot be directly applied. The main challenge comes from the fact that $w_i * n_{[i]}$ varies with group size since $n_{[i]}$ (the number of instances in a group) depends on $g_{[i]}$. To address this, we employ a simple modification of our approach: Rather than assigning weights to each group, we split the dataset into the same-size quantiles and then assign the weights to each quantile accordingly. This can be better understood through Figure 3, where the groups are arranged in order of their group loss. In this arrangement, $g_{[N]}$ represents the group with the poorest performance, followed by $g_{[N-1]}$ as the second poorest, and so forth. Each triangle represents all instances of the corresponding groups. As the number of instances in each quantile (Q_i) is the same and groups may have different sizes, instances of some groups will be split into different quantiles (e.g. black and yellow part for $g_{[N]}$). The instances that fall in the same quantile (with the same color shown in Figure 3) will be upweighted by the same factor w_q according to Equation 1. Then the new objective should be:

$$\mathcal{L}(\boldsymbol{\theta}) = n \left(\sum_{q=1}^{100} w_q \tilde{Y}_q \right) \tag{3}$$

where $n = \frac{\sum_{i=1}^{N} n_i}{100}$ is the number of instances in each quantile which is a constant. $\tilde{Y}_q$ is the average loss of quantile q. Then as long as the $\tilde{Y}_q$ is sorted, the convergence can be reached again according to [31] under ADMM optimization.

This is feasible when we keep the losses of each part of the group across boundaries the same (e.g. if the black and yellow parts for group $g_{[N]}$ shown in Figure 3 have the same loss and so on for each group then $\tilde{Y}_q$ is sorted.) Given a set of instance losses of a group, deciding whether it can be partitioned into two (or multiple) subsets such that the sum of the numbers in each set is the same is naturally a partition problem (which is NP-Hard).

We side-step solving this intractable problem by assigning weights at the group level by taking a weighted average over the reweighting factors of parts of the group across boundaries. For example, assume there are x instances in $g_{[N]}$ that fall into Q_{100}(black) and y instances that fall into Q_{99}(yellow), then the reweighting factor for the group is $\frac{x*w_{100}+y*w_{99}}{x+y}$. Generally, if a group $g_{[i]}$ spans across quantiles $Q_k, Q_{k-1}, \ldots, Q_{k-j}$, and the number of instances in each quantile is $n_{Q_k}, \ldots, n_{Q_{k-j}}$, then the weight for $g_{[i]}$ will be:

$$w'_i = \frac{\sum_{m=0}^{j} w_{k-m} n_{Q_{k-m}}}{n_{[i]}} \tag{4}$$

where $\sum_{m=0}^{j} n_{Q_{k-m}} = n_{[i]}$ the total number of instances of the group. And for any $m \neq 0$ and $m \neq j$, $n_{Q_{k-m}} = n$ should be the number of instances in each quantile. Even though knowing the real split of the group across quantiles is NP-hard, $n_{Q_k}, \ldots, n_{Q_{k-j}}$ is known since the size of the group and the number of instances in a quantile is known. (e.g., In Figure 3, if we know the number of instances in a quantile $n = 1000$ and further assume the group size of $g_{[N]}$ is $n_{[N]} = 1050$, then there should be 1000 instances in the black part of $g_{[N]}$ and 50 instances in the yellow part of $g_{[N]}$ and so on for all the groups.) Then, the new objective function can be written as:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \left(\sum_{i=1}^{N} w'_i n_{[i]} Y_{[i]} \right) \tag{5}$$

We can prove that the reweighted loss of a group under objective 3 is the same as the reweighted loss of a group under objective 5. Without loss of generality, we can assume that a group $g_{[i]}$ should be split into two quantiles under the ideal objective 3 as $g_{[N]}$ in Figure 3. Assume the sets of instances in two quantiles are $\mathcal{X}$ and $\mathcal{T}$ correspondingly. The losses of instances in $\mathcal{X}$ are $\{X_1, ..., X_\alpha\}$ and the losses of instances in $\mathcal{T}$ are $\{T_1, ..., T_\beta\}$ where α and β are the sizes of two parts and $\alpha + \beta = n_{[i]}$. Then under the ideal split so that each part has the same average loss, we have $\frac{\sum_{i=1}^{\alpha} X_i}{\alpha} = \frac{\sum_{j=1}^{\beta} T_j}{\beta}$. The average loss of the group $Y_{[i]} = \frac{(\sum_{i=1}^{\alpha} X_i + \sum_{j=1}^{\beta} T_j)}{n_{[i]}}$. Under the objective 3, the reweighted loss of this group

is then

$$
\begin{aligned}
& w_\alpha \sum_{i=1}^{\alpha} X_i + w_\beta \sum_{j=1}^{\beta} T_j \\
=& \frac{1}{\alpha}(\alpha w_\alpha \sum_{i=1}^{\alpha} X_i + \alpha w_\beta \sum_{j=1}^{\beta} T_j) \\
=& \frac{1}{\alpha}(\alpha w_\alpha \sum_{i=1}^{\alpha} X_i + \beta w_\beta \sum_{i=1}^{\alpha} X_i) \\
=& \frac{\sum_{i=1}^{\alpha} X_i}{\alpha}(\alpha w_\alpha + \beta w_\beta) \\
=& \frac{(\alpha+\beta)\sum_{i=1}^{\alpha} X_i}{(\alpha+\beta)\alpha}(\alpha w_\alpha + \beta w_\beta) \\
=& \frac{(\alpha w_\alpha + \beta w_\beta)}{\alpha+\beta}(1+\frac{\beta}{\alpha})\sum_{i=1}^{\alpha} X_i \\
=& w'_g(1+\frac{\beta}{\alpha})\sum_{i=1}^{\alpha} X_i \text{ according to Equation 4} \\
=& w'_g(\sum_{i=1}^{\alpha} X_i + \sum_{j=1}^{\beta} T_j) \text{ according to the even loss split} \\
=& w'_g n_{[i]} Y_{[i]}
\end{aligned}
\tag{6}
$$

We have shown that the reweighted loss of a group in objective 3 is the same as the reweighted loss of this group in objective 5 using the weighted reweighting factors in Equation 4. When a group is split into more than two quantiles, the proof can be applied recursively to adjacent quantiles as illustrated above. Hence, one can implement Objective 12 in practice without solving the NP-hard partition problem. Again, as mentioned earlier, the convergence of objective 3 has been proven by [31] under ADMM optimization.

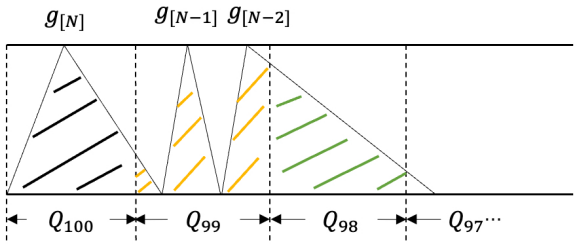


Fig. 3: Figure showing the division of instances into quantiles in the scenario in which groups differ in size. Suppose groups are ranked based on their loss, with $g_{[N]}$ representing the group with the worst performance and $g_{[N-1]}$ as the second worst, continuing in this order. Further, each triangle represents all instances belonging to its respective group. Given that each quantile (Q_i) contains an equal number of instances and the groups vary in size, instances from the same group may be distributed across different quantiles. This is illustrated by the division of $g_{[N]}$ into both black and yellow segments. Instances within the same quantile, even if they originate from different groups, are assigned the same upweighting factor which is determined by the equation 1.