
Mitigating Spurious Correlations with Memorization-Guided Dataset De-Biasing

Arda Fazla¹

Abolfazl Hashemi¹

¹Electrical and Computer Engineering Dept., Purdue University, West Lafayette, Indiana, USA

Abstract

Real-world datasets often contain spurious correlations that are not causally related to the target label. When such correlations dominate the majority of training samples, models tend to rely on them, leading to misclassification of minority samples that do not exhibit the same spurious patterns. While a potential approach is to select subsets of data to better represent the minority samples, this may require access to group labels, which are typically unknown. Furthermore, as we demonstrate, widely used sample scoring functions in the invariant subset or coreset selection literature largely depend on spurious features and therefore fail to accurately capture the importance or difficulty of core, causally relevant features. Accordingly, we propose to mitigate spurious correlations by developing a two-stage sample scoring function that disentangles the learning dynamics of core and spurious features and evaluates their difficulty separately. Based on our proposed metric, we introduce a new algorithm to find and prioritize informative samples both with and without spurious correlations. Extensive experiments demonstrate that a standard ERM model trained on our selected samples achieves superior performance compared to state-of-the-art debiasing techniques, while requiring as little as 10% of the original training data. Code is available at <https://github.com/ardafazla/TCSL>.

1 INTRODUCTION

Real-world datasets often contain a large number of samples with spurious correlations that are highly consistent within a class but not predictive of the true class label. Deep models learn to base their predictions on these simpler spurious fea-

tures rather than the more complex core features [Yang et al., 2024, Wang and Wang, 2024], leading to poor worst-group accuracy on test data where the spurious correlations may not hold. Thus, a large body of recent work has focused on developing specialized algorithms to mitigate the bias arising from spurious correlations. However, these approaches deviate from standard ERM training, which remains widely used in practice. This raises the question: can we reduce a model’s reliance on spurious correlations while retaining standard ERM training, without introducing complex, specially designed optimization techniques?

One intuitive approach that could aid with addressing the above question is to leverage existing invariant data or sample selection algorithms designed to construct high-quality subsets of data capable of representing the full training distribution. Leveraging this idea, one may form an informative sample collection, or *coreset*, that represents all groups within the dataset well. Training a standard ERM model on such a coreset may then achieve competitive performance across all groups. However, we argue that commonly utilized coreset selection algorithms fail to construct strong coresets on spurious datasets, as they are not explicitly designed to ensure high worst-group accuracy, but rather to ensure high average test accuracy. As a result, certain groups within the dataset may be underrepresented in the selected coreset, even when the overall average accuracy remains high. Thus, these coreset selection methods are not directly applicable to spurious datasets. Recent works [Dharasiri et al., 2025, Vysogorets et al., 2025] demonstrate that coreset selection strategies built on commonly used sample scores, such as EL2N [Paul et al., 2021] and SelfSup [Sorscher et al., 2022], fail to consistently achieve high worst-group accuracy across datasets with known spurious correlations.

Commonly used sample scoring functions and the coreset selection algorithms built upon them typically assign scores to individual samples to reflect their difficulty and derive selection strategies accordingly. We argue that, due to the “simplicity bias” phenomenon widely observed in deep learn-

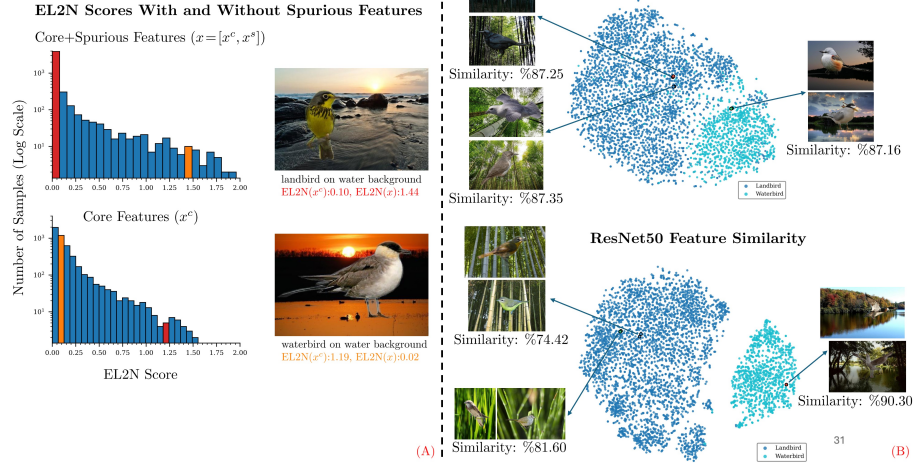


Figure 1: *Comparison of standard sample scores on Waterbirds.* (A) We visualize the EL2N scores for two example images from the Waterbirds dataset, computed on a model trained on the datasets with and without the background. The results demonstrate that the presence of background significantly changes the EL2N score. (B) We present representative image pairs with high similarity based on feature embeddings extracted from a ResNet50 model trained on Waterbirds and from CLIP. In both cases, high similarity is primarily driven by shared background characteristics, even when the bird attributes differ substantially.

ing [Valle-Perez et al., 2019, Shah et al., 2020, Kalimeris et al., 2019, Wang and Wang, 2024], models tend to learn simple spurious features before capturing more complex core features. Consequently, the behavior of sample scoring functions that rely on model outputs or loss values becomes dominated by spurious features. As a result, samples without spurious correlations are often assigned high scores and classified as “hard”, whereas samples with spurious correlations receive low scores and are treated as “easy”. This bias causes commonly used scoring functions in coreset selection to inadequately capture the strength of the underlying core features. We theoretically analyze the impact of simplicity bias on the learning speeds of core and spurious features, as well as on loss-based sample scores, in Section 2.1 and Appendix B.

We illustrate the effects of simplicity bias in Figure 1. In Figure 1(A), we show that the EL2N score changes significantly when the background feature is removed from the dataset, indicating that a common bird image may receive a higher score than a unique bird image solely due to background characteristics. Beyond loss-based sample scoring functions, certain coreset selection algorithms also incorporate sample similarity based on the extracted feature embeddings. Accordingly, we show representative image pairs with high similarity from the Waterbirds dataset in Figure 1(B), identified using cosine similarity of feature embeddings extracted from a ResNet model and the foundation model CLIP. In both cases, similarity is primarily driven by shared background features, even when bird attributes differ substantially. Furthermore, a recent study [Mulchandani and

Kim, 2025] demonstrates that even a small number of samples with simple spurious features and complex core features can cause the model to rely predominantly on the spurious feature for prediction. Motivated by these observations, we argue that commonly used sample scoring functions are dominated by spurious features; consequently, coreset selection algorithms built upon them may fail to distinguish between easy and hard core features, resulting in suboptimal coresets and poor worst-group accuracy.

As a remedy, we propose the Two-Stage Cumulative Sample Loss (TCSL) and the TCSL-guided Coreset Selection (TCSL-CS) algorithm. As illustrated in Figure 2(A), our approach builds on widely used two-stage training methods from the literature on learning under spurious correlations [Liu et al., 2023, Nam et al., 2020, Tsirigotis et al., 2023, Bayat et al., 2025] and distinguishes the score computation for core and spurious features without requiring access to the spurious attribute (group labels). Accordingly, TCSL consists of two scores, TCSL_s and TCSL_c , representing the computed difficulty of the spurious and core features, respectively. As shown in Figure 2(B), while conventional sample scoring functions assign a single score to each image, which is dominated by the spurious feature (e.g., background), our TCSL framework separately evaluates the difficulty of the core (bird) and spurious (background) components. We then design our TCSL-CS algorithm based on TCSL to effectively select coresets that achieve both (1) high average accuracy and (2) high worst-group accuracy. As shown in Figure 2(C), a coreset selected by TCSL-CS using only 10% of the training data improves the worst-group accuracy

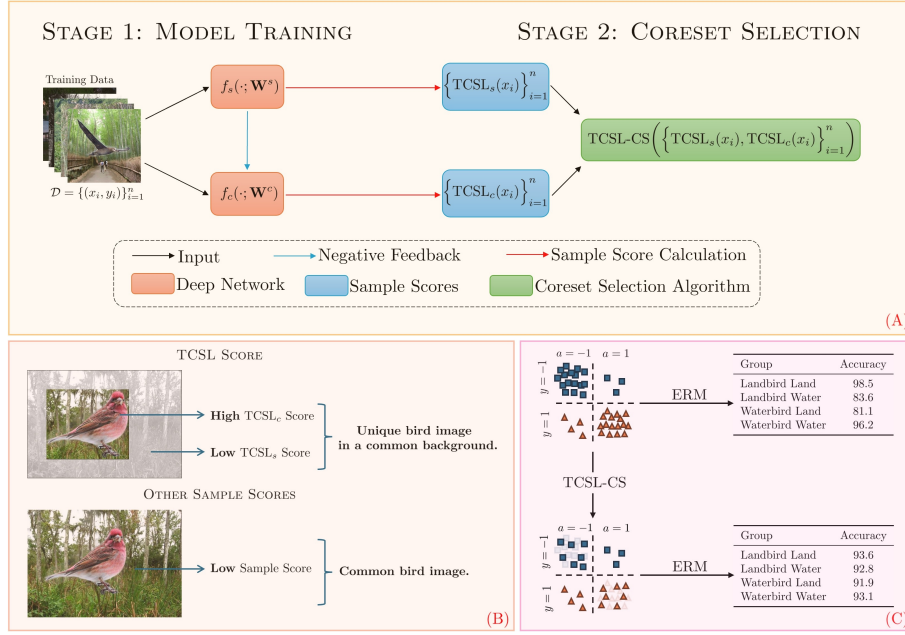


Figure 2: Overview of our proposed coreset selection algorithm. (A) We first train a two-stage model to accurately disentangle the learning processes of spurious and core features and compute sample scores for each component separately (TCSL_s and TCSL_c). We then construct our coreset selection algorithm, the Two-Stage Cumulative Sample Loss (TCSL)-guided Coreset Selection (TCSL-CS), based on the computed scores. (B) We illustrate the advantage of our TCSL score over existing sample scoring functions in the literature, showing its ability to disentangle the difficulty of different feature components within the image. (C) Training an ERM model on the coreset selected by TCSL-CS achieves state-of-the-art WGA on the Waterbirds dataset using only 10% of the total training data.

of a standard ERM model by 11.33% on the Waterbirds dataset, outperforming baselines that require group labels or complex optimization procedures.

Our contributions and scope:

- We propose the TCSL score, which separately quantifies the learning difficulty of core and spurious features in datasets with spurious correlations.
- Our proposed TCSL score enables us to leverage coreset selection as a principled tool to introduce TCSL-CS, an algorithm that selects effective coresets for datasets with spurious correlations, achieving both high average accuracy and high worst-group accuracy.
- We provide a strong theoretical analysis of the distinct learning dynamics of core and spurious features.
- Through extensive experiments on datasets with spurious correlations, we show that TCSL-CS outperforms existing debiasing and sample scoring baselines without requiring access to group labels.

2 PROBLEM FORMULATION

2.1 PROBLEM SETTING

Let $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ denote the training dataset of size n , where for each data sample we observe an input feature vector $x_i \in \mathbb{R}^d$ and its corresponding label y_i . For simplicity and without loss of generality, throughout the analysis, we focus on the binary classification setting where $y_i \in \{\pm 1\}$. Each data sample is also associated with an unobserved spurious attribute $a_i \in \{\pm 1\}$. We assume that within each class, the training data is partitioned into a majority group and a minority group, where the majority group contains all the samples with $a_i = y_i$, and the minority group contains all the samples with $a_i = -y_i$. The fraction of samples in the majority group within each class is denoted by α_y , where $\alpha_y > 0.5$ is commonly observed in practice. We denote the fraction of majority samples in the entire dataset by α . We consider the setting in which the spurious attribute, and thus the group labels, is unknown to us. We assume each sample x_i consists of core and spurious components, $x_i = [x_i^c, x_i^s]$, where x_i^c is correlated with y_i and x_i^s is correlated with a_i . For example, in the Waterbirds [Sagawa et al., 2020b] dataset, the bird region serves as the core feature x_i^c , while the background serves as the spurious feature x_i^s .

Deep learning models are typically trained via empirical risk minimization (ERM), where given a model with probability outputs $f(x; \mathbf{W})$ and weights $\mathbf{W}$, we minimize $\mathcal{L}(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^N \ell(y_i, f(x_i; \mathbf{W}))$ where ℓ can be any suitable loss function, e.g., cross entropy loss or sigmoid loss. Commonly, gradient-based optimization techniques are utilized, such as stochastic gradient descent (SGD), where the parameters of the model are updated at each iteration as $\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_t \nabla \mathcal{L}(\mathcal{B}_t, \mathbf{W}_t)$, where η_t denotes the learning rate at iteration t , and $\mathcal{B}_t$ is the mini-batch of samples from $\mathcal{D}$ used at step t . Depending on the context, we use t interchangeably to denote both the update and epoch step, with T denoting the total number of epochs. Here, $\nabla \mathcal{L}(\cdot, \cdot)$ represents the stochastic gradient of batch loss $\mathcal{L}(\mathcal{B}_t, \mathbf{W}_t)$, which is defined as the average weighted loss over batch:

$$\mathcal{L}(\mathcal{B}_t, \mathbf{W}_t) = \sum_{i=1}^{|\mathcal{B}_t|} \lambda_i \ell(y_i, f(x_i; \mathbf{W}_t)), \quad (1)$$

where λ_i denotes the weight associated with the i -th sample in the batch. Typically, λ_i is set to $1/|\mathcal{B}_t|$ to ensure uniform averaging. In the presence of class imbalanced data, λ_i is often chosen as the inverse number of samples belonging to the same class, which helps prevent mode collapse, i.e., the model degenerating to predict only the majority class. To maintain consistency across batches, an additional normalization step is applied such that the sum of sample weights within each batch equals one, ensuring that the overall optimization problem remains unchanged.

To evaluate algorithm performance, we use average test accuracy (ACC), defined as

$$\text{ACC}(\mathbf{W}) = \mathbb{P}_{(x,y) \sim \mathcal{D}_{\text{test}}} [\arg \max(f(x; \mathbf{W})) = y].$$

With spurious correlations, researchers are often more interested in the worst-group accuracy (WGA):

$$\text{WGA}(\mathbf{W}) = \min_{\substack{y \in \{\pm 1\} \\ a \in \{\pm 1\}}} \mathbb{P}_{(x,y,a) \sim \mathcal{D}_{\text{test}}} [\arg \max(f(x; \mathbf{W})) = y],$$

which measures the worst accuracy of the model among all groups defined by combinations of y and a .

Our overall goal is, given a coreset selection ratio r , to select $r|\mathcal{D}|$ samples from the training dataset $\mathcal{D}$ that maximize WGA on an unobserved test set. We consider a coreset successful if a standard ERM model trained on it achieves optimal WGA. Our objective is therefore to eliminate the dataset bias induced by spurious correlations and hence improve WGA, without compromising overall generalization performance. Equivalently, the task can be viewed as selecting a coreset of size $c|\mathcal{D}|$ that effectively and equally represents all groups in the training dataset, without access to the group labels.

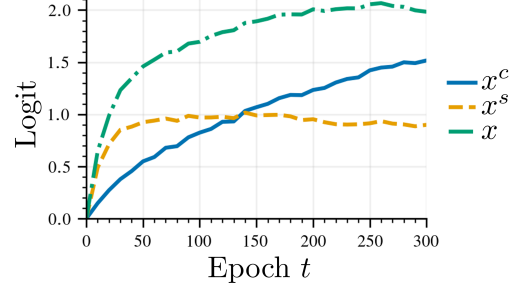


Figure 3: Average logits computed over the entire dataset. x , x^c , and x^s denote the logits from full images, core features only and spurious features only, respectively.

2.2 FEATURE LEARNING ANALYSIS

We begin by demonstrating the logit outputs of a ResNet50 model trained with SGD on the Waterbirds dataset, as shown in Figure 3. Every few epochs, we record the average model predictions for training samples belonging to the largest majority group: landbirds on land backgrounds. The results are collected for the full image, as well as for its core and spurious components, which are manually extracted and denoted as x , x^c , and x^s , respectively. As illustrated in Figure 3, the logits induced by the spurious features converge much faster than those induced by the core features. This commonly observed behavior in deep learning is called the simplicity bias phenomenon, where models exhibit a strong preference for simpler features over more complex ones [Valle-Perez et al., 2019, Shah et al., 2020, Kalimeris et al., 2019]. In our context, when the spurious feature is frequently observed in the dataset and has a higher feature strength than its core counterpart, the model tends to ignore the core feature during the early stages of training, rapidly fitting to the spurious correlations instead [Yang et al., 2024, Qiu et al., 2023a]. While prior work quantitatively defines feature strength in various ways [Yang et al., 2024, Bombari and Mondelli, 2024], we avoid setting a strict definition and instead treat the strengths of the spurious and core features as unknown, independent, non-negative quantities denoted by β^s and β^c , respectively.

In Theorem 1 (proved in Appendix B.1), we analyze the learning dynamics of a spiked covariance model [Donoho et al., 2018] under the Neural Tangent Kernel (NTK) regime [Jacot et al., 2018] and provide theoretical justification showing that the learning speed of spurious features dominates that of core features during the early stages of training. We focus on the NTK regime (see Appendix B for details), following similar theoretical studies [Bombari and Mondelli, 2024, Yang et al., 2024] as it offers a mathematically tractable and strong approximation of gradient-based training, which has been shown to capture the learning behavior of a broad class of deep learning architectures [Jacot et al., 2018, Lee et al., 2019]. We adopt the spiked covari-

ance model, as it enables a clean separation between core and spurious features while also reflecting key properties of real-world datasets [Wang et al., 2024a].

Theorem 1. *Define the simplicity bias condition as $\beta^s(2\alpha - 1) > \beta^c$ and assume this holds. Let $\bar{m}_t(x_i) = \mathbb{E}_{\mathbf{W}_0}[y_i h(x_i; \mathbf{W}_t)]$ be the expected margin of the model. Then, there exists a time $T' > 0$ such that for all $t \in (0, T')$:*

1. *For any sample from the majority group with $a_i = y_i$, the expected margin $\bar{m}_t(x_i)$ is positive, and the loss $\ell(\bar{m}_t(x_i))$ is less than $\log(2)$.*
2. *For any sample from the minority group with $a_i = -y_i$, the expected margin $\bar{m}_t(x_i)$ is negative, and the loss $\ell(\bar{m}_t(x_i))$ is greater than $\log(2)$.*

We analyze the expected initial growth rates of the true core margin and the true spurious margin, and show that under the simplicity bias condition $\beta^s(2\alpha - 1) > \beta^c$, the spurious subnetwork is learned faster than the core subnetwork during the initial stages of training. Theorem 1 and Theorem 2 on initial acceleration (stated in Appendix B.1 due to space limitations) which help explain the phenomenon demonstrated in Figure 3 are further generalized in Appendix B.2.

Takeaway. In the proof of Theorem 1, we show that the learning of the spurious feature will be faster than the core feature during the early stages of training. Hence, the model output will be dominated by the spurious subnetwork. Consequently, there exists $T' > 0$ such that, for all $t \in (0, T')$, the expected loss of the majority group is upper bounded by $\log(2)$, while the expected loss of the minority group is lower bounded by $\log(2)$ over the same interval. Together, these results indicate that sample losses are shifted to smaller or larger values depending on the spurious attribute, independent of the strength of the core features.

Building on Theorem 1, we next show that widely used sample scoring functions in the coreset selection literature cannot be directly employed in the presence of spurious correlations, and motivate the need to separately quantify the difficulty of the core and spurious components of each sample.

Takeaway 2. Consider any sample scoring function that is non-decreasing in the per-step loss, which captures a wide variety of sample scores employed in the coreset selection literature, e.g., instantaneous loss, average/cumulative loss, area under margin, EL2N, GraNd. Since the spurious feature is learned faster than the core feature, the effect of the spurious component will be evident throughout training. Consequently, by Theorem 1, the loss (therefore the assigned sample score) will be dominated by the spurious feature and the (unknown) spurious attribute a . Since widely used sample scoring functions [Paul et al., 2021, Sorscher et al., 2022, Feldman, 2020] assign a single score to each sample, they are inherently unable to assess the feature strengths of the

core and spurious components simultaneously. We extend and validate our theoretical findings under the NTK regime to more general deep learning settings through experiments in Section 4.

3 TCSL-CS

Our goal is to construct two separate subnetworks to disentangle the learning dynamics of spurious and core components. These subnetworks enable us to independently compute difficulty scores associated with spurious and core features. In Section 3.1, we introduce a two-stage network designed to separate the learning processes of the spurious and core components. In Section 3.2, we propose a two-stage sample scoring function based on the learned subnetworks. Finally, in Section 3.3, we present **Algorithm 2** that integrates information from both core and spurious components to achieve group-robust coreset selection.

3.1 MODEL TRAINING

To separate the sample scores into spurious and core components, we propose a two-stage model training scheme described in **Algorithm 1**. For notational simplicity, we write $f_s(x_i) = f_s(x_i; \mathbf{W}_t^s)$ and $f_c(x_i) = f_c(x_i; \mathbf{W}_t^c)$ and omit the explicit dependence on the parameters whenever it is clear from the context. In the first stage, we train a biased model designed to learn only the spurious features. As supported by our theoretical analysis in Appendix B and by prior work on simplicity bias [Yang et al., 2024, Deng et al., 2023, Qiu et al., 2023a], spurious features are learned earlier in training compared to core features. Consequently, samples with strong and easy spurious features are learned the fastest [Deng et al., 2023]. Building on this intuition, at the end of each epoch, we upweight samples with the lowest loss so that they contribute more to the learning process during the next epoch. Hence, we effectively upweight samples for which the spurious attribute agrees with the label ($a = y$), thereby amplifying the learning process of the bias in the data. The proposed architecture is an adapted version of the FLOW algorithm [Sanyal et al., 2025], originally developed for multi-task learning, which we modify for our task. The median term in the denominator is used to control the range of sample weights across iterations.

After training the spurious (biased) model f_s for T_s epochs, we freeze its weights and reset the sample weights λ_i . We then initialize the core model f_c and train it for T_c epochs using SGD, where the cross-entropy loss over a random mini-batch $\mathcal{B}_t$ is adjusted to

$$\mathcal{L}(\mathcal{B}_t, \mathbf{W}_t^c) = \sum_{i=1}^{|\mathcal{B}_t|} \lambda_i \ell \left(y_i, f_c(x_i; \mathbf{W}_t^c) + \log(f_s(x_i)) \right). \quad (2)$$

Algorithm 1 Two-Stage Model Training

Input: Dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, training epochs T_c and T_s . Let n_{y_i} denote the number of samples in class y_i .

Initialize:

- Classifiers f_c, f_s
- Sample weights $\lambda_i = \frac{1}{n_{y_i}}$ for all $i \in \{1, \dots, n\}$ (uniform within each class)

Output: Trained models f_c, f_s .

- 1: **Stage 1: Train spurious model** f_s
- 2: **for** $t = 1$ to T_s **do**
- 3: Train f_s on $\mathcal{D}$ using SGD according to Equation (1)
- 4: Update sample weights:

$$\lambda_i \leftarrow \exp\left(-\frac{\ell(y_i, f_s(x_i))}{\text{median}_j (\ell(y_j, f_s(x_j)))}\right) \quad \forall i$$

- 5: **end for**
 - 6: Freeze f_s and reset $\lambda_i = \frac{1}{n_{y_i}}$ for all i
 - 7: **Stage 2: Train core model** f_c
 - 8: **for** $t = 1$ to T_c **do**
 - 9: Train f_c on $\mathcal{D}$ using SGD according to Equation (2)
 - 10: **end for**
-

Only the core model f_c is updated through Equation (2), as f_s is frozen. Since f_s outputs probabilities, we have $f_s(x) \in [0, 1]$. Intuitively, applying the log operator yields negative values that reflect the strength of the learned spurious relationships captured by f_s . Consequently, f_s acts as a negative feedback mechanism and f_c is pushed towards learning the core features.

Similar two-stage debiasing architectures have been proposed in the literature under the name “logit correction” [Liu et al., 2023, Tsirigotis et al., 2023, Nam et al., 2020, Bayat et al., 2025]. In contrast to prior work, we do not employ a debiasing architecture to optimize model performance, but rather to compute sample scores associated exclusively with the spurious and core features, which are then used for coreset selection.

3.2 TWO-STAGE CUMULATIVE SAMPLE LOSS

For our two-stage per-sample score, we build on the literature on the memorization score, initially proposed by Feldman [2020]. Since computing the memorization score is computationally inefficient, we instead adopt its recently proposed proxy, the Cumulative Sample Loss (CSL) [Ravikumar et al., 2025], where the authors show that the originally proposed memorization score is bounded by the accumulated sample losses over epochs. The original definitions of these scores are provided in Appendix C. We compute the CSL scores for the two learned networks sepa-

Algorithm 2 TCSL-guided Coreset Selection

Input: TCSL scores for $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, selection ratio r , loss curves of f_s for every sample, number of bins B , threshold τ .

Output: Coreset $\mathcal{D}'$ of size $r|\mathcal{D}|$.

- 1: **Stage 1: Identify groups (by TCSL_s)**
 - 2: Set $w_i = \text{TCSL}_s(x_i)$, $L_{i,t} = \ell(y_i, f_s(x_i; \mathbf{W}_t^s))$
 - 3: $\{G_{\text{high}}, G_{\text{low}}\} \leftarrow \text{WKMEANS}(\mathbf{w}, \mathbf{L})$
 - 4: $\mathcal{D}' \leftarrow \text{UNIFORMSELECT}(G_{\text{high}}, \min(|G_{\text{high}}|, r|\mathcal{D}|))$
 - 5: **Stage 2: Fill remaining quota (by TCSL_c)**
 - 6: **if** $|\mathcal{D}'| < r|\mathcal{D}|$ **then**
 - 7: $G_{\text{low}} \leftarrow \text{SORT}(G_{\text{low}}; \text{ascending by TCSL}_c)$
 - 8: $\tilde{n} \leftarrow r|\mathcal{D}| - |\mathcal{D}'|$
 - 9: **if** $r \geq \tau$ **then**
 - 10: $S_{\text{low}} \leftarrow \text{SELECTBOT}(G_{\text{low}}, \tilde{n})$
 - 11: **else**
 - 12: $S_{\text{low}} \leftarrow \text{SELECTHIST}(G_{\text{low}}, \tilde{n}, B)$
 - 13: **end if**
 - 14: $\mathcal{D}' \leftarrow \mathcal{D}' \cup S_{\text{low}}$
 - 15: **end if**
 - 16: **Return** $\mathcal{D}'$
-

ately. Our Two-Stage Cumulative Sample Loss (TCSL) is defined as:

$$\text{TCSL}(x) = [\text{TCSL}_s(x), \text{TCSL}_c(x)], \quad (3)$$

where $\text{TCSL}_s(x) = \frac{1}{T_s} \sum_{t=1}^{T_s} \ell(y, f_s(x; \mathbf{W}_t^s))$ and $\text{TCSL}_c(x) = \frac{1}{T_c} \sum_{t=1}^{T_c} \ell(y, f_c(x; \mathbf{W}_t^c))$. As TCSL requires sample losses after every epoch, we compute and store the loss of each samples during subnetwork training in **Algorithm 1**. Intuitively, for samples where the spurious attribute disagrees with the label ($a = -y$), the TCSL_s score should be high, as the spurious model f_s is likely to misclassify such samples. In contrast, the TCSL_c score should depend solely on the difficulty of the core features. We verify these observations experimentally in Section 4.

3.3 GROUP ROBUST CORESET SELECTION

We propose a group-robust coreset selection algorithm based on the TCSL score, termed TCSL-guided Coreset Selection (TCSL-CS), described in **Algorithm 2**. We first identify samples with high TCSL_s scores (samples without strong spurious features) using the WKMEANS algorithm on the TCSL_s scores and the loss curves of f_s . WKMEANS applies a weighted variant of the k-means algorithm [Likas et al., 2003], where each sample is represented by its losses computed by f_s over T_s epochs and each sample is assigned a weight given by its TCSL_s score. Since the sample losses are already obtained during the computation of TCSL_s , this step introduces no additional computational cost but provides the WKMEANS algorithm with additional information

for clustering. Details of WKMEANS are provided in Appendix D. Samples with high TCSL_s scores are grouped into G_{high} , from which we randomly select samples until the quota $\min(|G_{\text{high}}|, r|\mathcal{D}|)$ is filled. As $\alpha > 0.5$ by definition, the majority of samples are assigned to group G_{low} .

Next, we fill the remaining quota $(r|\mathcal{D}| - |\mathcal{D}'|)$ with samples from G_{low} . Following recent consensus in the coreset selection literature, we follow two different selection strategies depending on the selection ratio r . Since the simplicity bias arises primarily from samples with simple spurious but complex core features, our goal is to eliminate such samples. Therefore, for $r \geq \tau$, we use SELECTBOT to select samples with low TCSL_c scores. However, when the coreset selection ratio is small ($r < \tau$), selecting only samples with low TCSL_c scores reduces diversity and is thus suboptimal. In this case, we use SELECTHIST, which partitions the samples into B bins based on the TCSL_c scores such that each bin contains an equal number of samples and randomly samples from each bin until the quota is filled. Finally, we combine the samples selected from both stages to form the final coreset $\mathcal{D}'$. Details of SELECTBOT and SELECTHIST are provided in Appendix D. We provide the runtime, computational complexity, and memory usage analysis of our algorithm in Appendix J.

4 EXPERIMENTS

4.1 PERFORMANCE COMPARISON

In this section, we evaluate the effectiveness of the proposed TCSL-CS algorithm on four computer vision benchmark datasets known to exhibit spurious correlations: Waterbirds [Sagawa et al., 2020b], cMNIST [Arjovsky et al., 2019], MetaShift [Liang and Zou, 2022], and UrbanCars-B [Li et al., 2023]. Additional details on the datasets are provided in Appendix E and training details for each dataset are provided in Appendix G.

First, we evaluate TCSL-CS as a debiasing algorithm by comparing the performance of an ERM model trained on the coreset identified by TCSL-CS with coreset selection ratio $r=0.1$ against state-of-the-art debiasing methods from the literature. We choose this ratio to assess the effectiveness of our approach, as it highlights the two key capabilities of our proposed method: (1) clustering samples into high and low spuriousity regions as identified by the TCSL_s score and (2) ranking high spuriousity samples based on their core feature difficulty, as captured by the TCSL_c score. Results for different coreset selection ratios are presented in Appendix H. Additional information on the baseline methods is provided in Appendix F.

Table 1 and Table 2 present the worst-group accuracy (WGA) and average accuracy (AVG) on the test datasets for a standard ERM model trained on the coreset identified

by our TCSL-CS algorithm, compared with other baseline debiasing methods. Training a standard ERM model on the TCSL-CS coreset substantially improves WGA, achieving performance on par with state-of-the-art baselines across multiple datasets, even outperforming models that have access to group labels (spurious attribute) in some cases. For instance, TCSL-CS improves the WGA of ERM by 10.44% on the Waterbirds dataset, outperforming GroupDRO, a strong baseline that requires access to the spurious attribute.

Next, we compare the performance of TCSL-CS against other coreset selection algorithms from the literature. To the best of our knowledge, TCSL-CS is the first coreset selection method specifically designed to operate under spurious correlations and achieve group robustness. For completeness, we follow the experimental setup of recent work [Dharmasiri et al., 2025] and use EL2N, Random, RGBal, and SelfSup as our baselines for computing sample scores. We then construct coreset selection strategies under three different settings: (Bot) selects samples with the lowest scores, (Top) selects samples with the highest scores and (Hist) selects samples using a histogram-based approach. We also provide four versions of the state-of-the-art D2 coreset selection algorithm [Maharana et al., 2024], which effectively combines sample scores with feature similarity for enhanced data selection and coverage. Further details for these algorithms are provided in Appendix F.

The results are shown in Table 3. Overall, TCSL-CS outperforms all other methods across all datasets, except on Waterbirds, where RGBal achieves slightly better performance with access to group labels. Moreover, on the cMNIST dataset, where spurious correlations are particularly strong ($\alpha = 0.995$), TCSL-CS surpasses the next best method by more than 11%. The poor performance of baseline sample scores supports our theoretical analysis in Section 2.2 and aligns with recent findings [Dharmasiri et al., 2025] showing that widely used coreset selection algorithms perform poorly when spurious correlations are dominant in the data. Additional results for different coreset selection ratios are provided in Appendix H.

4.2 ABLATION STUDY

We conduct an ablation study on the Waterbirds dataset, as it is synthetically constructed and allows for an easy separation of core and spurious features. Here, we evaluate the accuracy of the TCSL score in disentangling the computation of spurious and core feature difficulties. Additional ablation studies on TCSL_s and TCSL_c , as well as cross-architecture evaluations and robustness of the hyperparameters of TCSL-CS, are provided in Appendix I.

In Figure 4, we compute the cosine similarity of the CSL scores under different training schemes. $\text{CSL}(x)$ denotes the class-balanced ERM score on the full dataset, $\text{CSL}(x^c)$ and

Method	Group Info		Waterbirds	cMNIST
	Train	Val	WGA (AVG)	WGA (AVG)
CB ERM	x	x	81.15±2.26 (97.97±0.03)	56.61±3.41 (76.01±2.54)
GB ERM	✓	✓	89.98±0.77 (98.20±0.03)	<u>81.46</u> ±3.33 (91.83±1.10)
GroupDRO*	✓	✓	91.40±1.10 (93.50±0.30)	58.66±5.46 (76.56±2.26)
LC*	x	✓	90.5±1.1	71.25±3.17
DFR*	x	✓	92.1 (96.7)	74.2 (93.7)
CNC*	x	✓	88.5±0.3 (90.9±0.1)	77.4±3.0 (90.9±0.6)
LfF*	x	✓	75.2 (97.5)	77.0
JTT*	x	✓	86.0 (93.6)	74.04±1.33
ULA*	x	✓	86.1±1.5 (91.5±0.7)	75.13±0.78
EIL*	x	x	77.2±1.0 (96.5±0.2)	72.8±6.8 (90.7±0.9)
GEORGE*	x	x	76.2±2.0 (95.7±0.5)	76.4±2.3 (89.5±0.3)
TCSL-CS (ERM)	x	x	<u>91.91</u> ±0.35 (92.83±0.55)	83.37 ±1.33 (91.76±0.44)

Table 1: We compare the WGA of different methods on the Waterbirds and cMNIST datasets. We provide the AVG in parentheses, if available. * indicates the original scores reported by the authors or subsequent work. The best WGA for each dataset is shown in **bold** and the second best value is underlined.

Method	Group Info		MetaShift	UrbanCars-B
	Train	Val	WGA (AVG)	WGA (AVG)
CB ERM	x	x	73.15±2.63 (87.27±0.27)	66.00±2.80 (83.30±0.36)
GB ERM	✓	✓	75.69±2.50 (88.48±0.50)	74.53±3.63 (87.23±1.17)
GroupDRO	✓	✓	74.07±2.89 (87.62±0.20)	71.33±1.29 (85.63±0.68)
LC	x	✓	<u>76.16</u> ±1.75 (88.71±0.63)	<u>78.67</u> ±1.29 (87.40±0.4)
DFR	x	✓	75.23±0.80 (95.17±0.40)	77.50±1.10 (81.00±0.40)
TCSL-CS (ERM)	x	x	79.40 ±2.23 (84.95±1.52)	84.27 ±1.67 (86.93±0.78)

Table 2: We compare the WGA of different methods on the MetaShift and UrbanCars-B datasets. We also provide the AVG in parentheses. Results are averaged over 3 seeds. The best WGA for each dataset is shown in **bold** and the second best value is underlined.

Method	Waterbirds WGA (AVG)	cMNIST WGA (AVG)	MetaShift WGA (AVG)	UrbanCars-B WGA (AVG)
EL2N (Bot)	39.88±0.93 (93.90±0.16)	0.00±0.00 (15.90±1.08)	53.47±4.81 (78.76±2.01)	35.60±4.33 (68.07±3.59)
EL2N (Top)	88.13±3.32 (89.53±2.61)	52.67±5.84 (76.58±2.34)	35.19±19.60 (59.61±9.74)	28.53±4.74 (56.63±2.43)
EL2N (Hist)	64.94±9.30 (95.97±0.33)	1.72±1.12 (24.10±5.07)	64.12±4.93 (81.48±2.27)	54.00±1.74 (75.33±0.61)
SelfSup (Bot)	59.81±0.52 (91.01±0.59)	0.00±0.00 (18.48±0.26)	38.19±6.01 (77.14±1.90)	38.80±1.20 (74.90±0.46)
SelfSup (Top)	75.39±0.82 (96.25±0.28)	58.15±3.09 (77.45±1.44)	37.96±5.31 (75.35±3.01)	27.47±5.72 (64.10±1.87)
SelfSup (Hist)	74.97±0.64 (96.11±0.21)	3.94±3.72 (24.57±2.83)	68.06±3.67 (84.26±1.28)	57.87±10.51 (79.00±3.22)
D2 (EL2N+ResNet)	89.92±2.14 (91.44±2.55)	31.77±11.91 (58.74±4.54)	26.85±5.57 (55.61±4.76)	53.20±4.00 (66.27±0.78)
D2 (SelfSup+ResNet)	75.49±1.67 (96.37±0.08)	22.89±10.66 (55.98±5.44)	37.50±11.47 (75.17±2.73)	30.13±3.80 (68.77±1.14)
D2 (EL2N+CLIP)	90.39±0.74 (96.82±0.37)	28.18±10.38 (58.19±1.76)	<u>78.01</u> ±0.40 (82.47±1.67)	58.40±0.40 (65.13±3.17)
D2 (SelfSup+CLIP)	75.49±1.41 (97.31±0.07)	32.39±6.14 (61.57±2.31)	46.53±7.89 (77.89±2.09)	28.67±7.64 (65.80±1.91)
Random	62.15±1.84 (96.69±0.12)	1.82±2.96 (24.76±1.15)	65.97±1.84 (81.77±0.90)	51.07±3.49 (76.07±1.67)
RGbal	92.67 ±0.58 (93.68±0.41)	<u>71.68</u> ±13.83 (87.59±2.24)	70.14±2.78 (81.13±0.44)	<u>84.00</u> ±1.20 (86.57±0.06)
TCSL-CS	<u>91.91</u> ±0.35 (92.83±0.55)	83.37 ±1.33 (91.76±0.44)	79.40 ±2.23 (84.95±1.52)	84.27 ±1.67 (86.93±0.78)

Table 3: We compare the WGA and AVG of different coresets selection methods at $r = 0.1$ across datasets. Results are averaged over 3 seeds. We use CB ERM for retraining on all of the identified coresets. The best WGA for each dataset is shown in **bold** and the second best value is underlined. Only RGbal has access to group labels.

$CSL(x^s)$ are scores from ERM models trained on core-only and spurious-only features, respectively. Figure 4 demonstrates that $TCSL_s$ and $TCSL_c$ achieve high cosine similarity with $CSL(x^s)$ and $CSL(x^c)$, respectively. These results

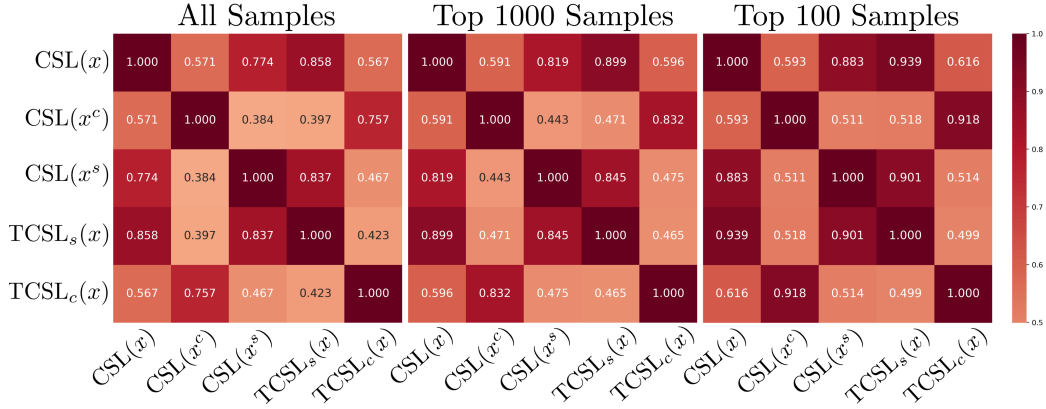


Figure 4: Cosine similarity between the TCSL score components and the CSL scores computed on the Waterbirds dataset for ERM models trained with and without the core and spurious parts of images. Samples with the highest core feature difficulty, as identified by the TCSL_c score, are specifically evaluated.

confirm that TCSL successfully disentangles the learning processes of core and spurious features. We further evaluate the similarity for the Top 1,000 and Top 100 samples with the highest TCSL_c scores. For the Top 100 samples, CSL(x^c) and TCSL_c are nearly identical, with a cosine similarity of 0.918. This further demonstrates that TCSL_c accurately captures core feature difficulty, effectively identifying samples with the most challenging core features, regardless of the spurious attribute. We additionally include density visualizations of EL2N and TCSL scores across different groups in Appendix I.

5 CONCLUSION

We present TCSL-CS, a debiasing framework based on core-set selection, specifically built for datasets with strong spurious correlations. TCSL-CS is built on top of our proposed TCSL score, which disentangles the sample score computation for the core and spurious features of images and provides difficulty estimates for each part separately. Across a variety of datasets, we show that a standard ERM model trained on the coresets identified by TCSL-CS achieves performance on par with or exceeding state-of-the-art baselines across multiple datasets, even outperforming models that have access to group labels.

We highlight that our debiasing method based on coreset selection has a significant advantage over other debiasing methods: standard ERM trained on only 10% of the data achieves state-of-the-art performance on highly spurious datasets. Hence, the identified coreset is broadly applicable, for example, it can be used as part of larger training pipelines. The expensive training step is performed once, after which the resulting coreset can be reused as a standalone product. Thus, TCSL-CS is significantly more scalable than other debiasing methods.

Acknowledgements

This work is supported in part by NSF DMS-2502560 and CNS-2313109.

References

- Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.
- Reza Bayat, Mohammad Pezeshki, Elvis Dohmatob, David Lopez-Paz, and Pascal Vincent. The pitfalls of memorization: When memorization hurts generalization. In *International Conference on Learning Representations*, volume 2025, pages 86017–86043, 2025.
- Simone Bombari and Marco Mondelli. How spurious features are memorized: Precise analysis for random and ntk features. In *Forty-first International Conference on Machine Learning*, 2024.
- Elliot Creager, Jörn-Henrik Jacobsen, and Richard Zemel. Environment inference for invariant learning. In *International Conference on Machine Learning*, pages 2189–2200. PMLR, 2021.
- Yihe Deng, Yu Yang, Baharan Mirzasoleiman, and Quanquan Gu. Robust learning with progressive data expansion against spurious correlation. *Advances in neural information processing systems*, 36:1390–1402, 2023.
- Amaya Dharmasiri, William Yang, Polina Kirichenko, Lydia T Liu, and Olga Russakovsky. The impact of coreset selection on spurious correlations and group robustness. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

- David L Donoho, Matan Gavish, and Iain M Johnstone. Optimal shrinkage of eigenvalues in the spiked covariance model. *Annals of statistics*, 46(4):1742, 2018.
- Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.
- Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.
- Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32, 2019.
- Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. In *The Eleventh International Conference on Learning Representations*, 2023.
- Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.
- Zhiheng Li, Ivan Evtimov, Albert Gordo, Caner Hazirbas, Tal Hassner, Cristian Canton Ferrer, Chenliang Xu, and Mark Ibrahim. A whac-a-mole dilemma: Shortcuts come in multiples where mitigating one amplifies others. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20071–20082, 2023.
- Weixin Liang and James Zou. Metashift: A dataset of datasets for evaluating contextual distribution shifts and training conflicts. In *International Conference on Learning Representations*, 2022.
- Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.
- 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.
- Sheng Liu, Xu Zhang, Nitesh Sekhar, Yue Wu, Prateek Singhal, and Carlos Fernandez-Granda. Avoiding spurious correlations via logit correction. In *The Eleventh International Conference on Learning Representations*, 2023.
- Adyasha Maharana, Prateek Yadav, and Mohit Bansal. D2 pruning: Message passing for balancing diversity & difficulty in data pruning. In *International Conference on Learning Representations*, volume 2024, pages 55740–55759, 2024.
- Varun Mulchandani and Jung-Eun Kim. Severing spurious correlations with data pruning. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: De-biasing classifier from biased classifier. *Advances in Neural Information Processing Systems*, 33:20673–20684, 2020.
- Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Mądry. Trak: attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, pages 27074–27113, 2023.
- Debashis Paul. Asymptotics of sample eigenstructure for a large dimensional spiked covariance model. *Statistica Sinica*, pages 1617–1642, 2007.
- Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34:20596–20607, 2021.
- GuanWen Qiu, Da Kuang, and Surbhi Goel. Complexity matters: Dynamics of feature learning in the presence of spurious correlations. In *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*, 2023a.
- Shikai Qiu, Andres Potapczynski, Pavel Izmailov, and Andrew Gordon Wilson. Simple and fast group robustness by automatic feature reweighting. In *International Conference on Machine Learning*, pages 28448–28467. PMLR, 2023b.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.
- Deepak Ravikumar, Efstathia Soufleri, Abolfazl Hashemi, and Kaushik Roy. Towards memorization estimation: Fast, formal and free. In *Forty-second International Conference on Machine Learning*, 2025.
- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

- Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020a.
- Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020b.
- Sunny Sanyal, Hayden Prairie, Rudrajit Das, Ali Kavis, and Sujay Sanghavi. Upweighting easy samples in fine-tuning mitigates forgetting. In *Forty-second International Conference on Machine Learning*, 2025.
- Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585, 2020.
- Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. *Advances in Neural Information Processing Systems*, 33:19339–19352, 2020.
- Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022.
- Christos Tsirigotis, Joao Monteiro, Pau Rodriguez, David Vazquez, and Aaron C Courville. Group robust classification without any group information. *Advances in Neural Information Processing Systems*, 36:56553–56575, 2023.
- Guillermo Valle-Perez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*, 2019.
- Artem M Vysogorets, Kartik Ahuja, and Julia Kempe. Drop: Distributionally robust data pruning. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.
- Yipei Wang and Xiaoqian Wang. On the effect of key factors in spurious correlation: A theoretical perspective. In *International Conference on Artificial Intelligence and Statistics*, pages 3745–3753. PMLR, 2024.
- Zhichao Wang, Denny Wu, and Zhou Fan. Nonlinear spiked covariance matrices and signal propagation in deep neural networks. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 4891–4957. PMLR, 2024a.
- Zhichao Wang, Denny Wu, and Zhou Fan. Nonlinear spiked covariance matrices and signal propagation in deep neural networks. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 4891–4957. PMLR, 2024b.
- Yu Yang, Eric Gan, Gintare Karolina Dziugaite, and Baharan Mirzasoleiman. Identifying spurious biases early in training through the lens of simplicity bias. In *International conference on artificial intelligence and statistics*, pages 2953–2961. PMLR, 2024.
- Michael Zhang, Nimit S Sohoni, Hongyang R Zhang, Chelsea Finn, and Christopher Re. Correct-n-contrast: a contrastive approach for improving robustness to spurious correlations. In *International Conference on Machine Learning*, pages 26484–26516. PMLR, 2022.
- Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.
- Haizhong Zheng, Rui Liu, Fan Lai, and Atul Prakash. Coverage-centric coresets selection for high pruning rates. In *The Eleventh International Conference on Learning Representations*, 2023.
- Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva. Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*, 2016.

Mitigating Spurious Correlations with Memorization-Guided Dataset De-Biasing (Supplementary Material)

Arda Fazla¹

Abolfazl Hashemi¹

¹Electrical and Computer Engineering Dept., Purdue University, West Lafayette, Indiana, USA

APPENDIX

A Related Work	13
B Theoretical Analysis	14
B.1 Homogeneous Spiked Model	15
B.2 Heterogeneous Spiked Model	18
B.3 Validation of Our Theoretical Setting and Assumptions	20
B.3.1 Analysis on the Linearization in Theorem 2	20
B.3.2 Verification of Simplicity Bias Through a Toy Dataset	23
B.3.3 Relaxation to Multiple Spurious Correlations	23
B.4 Background on the NTK and Spiked Covariance Model	25
B.4.1 Background on the NTK	25
B.4.2 Background on the Spiked Covariance Model	26
C Memorization and CSL Definitions	27
C.1 Memorization Score	27
C.2 Cumulative Sample Loss (CSL)	27
D Formal Definitions of Helper Algorithms	27
E Dataset Details	27
F Baseline Methods and Sample Scores	28
G Hyperparameters for Model Training and Coreset Selection	29
G.1 Model Training	29
G.2 Coreset Selection	30

H Additional Experiments	30
I Additional Ablation Studies	32
J Computational Efficiency	34
J.1 Complexity Analysis	34
J.2 Runtime Analysis	34
J.3 Memory Usage Analysis	34

A RELATED WORK

Simplicity Bias. The simplicity bias phenomenon of deep learning models trained on SGD, where the model learns simpler features before more complex ones even when the latter are more predictive, has been both theoretically analyzed and empirically observed across various settings and architectures [Valle-Perez et al., 2019, Shah et al., 2020, Kalimeris et al., 2019]. Recent studies have further extended the notion of simplicity bias to learning under spurious correlations, showing that deep models tend to learn spurious features before capturing the more complex core features [Yang et al., 2024, Qiu et al., 2023a, Deng et al., 2023, Wang and Wang, 2024].

Mitigating Spurious Correlations. A variety of methods have been proposed to mitigate the learning of spurious correlations during training. SPARE [Yang et al., 2024] leverages the simplicity bias phenomenon by upsampling underrepresented groups that are identified early in training. Feature reweighting based methods [Kirichenko et al., 2023, Qiu et al., 2023b] finetune the final layer of an ERM model using a group balanced validation set. Two-stage approaches [Liu et al., 2021, Nam et al., 2020] train two models, where the second model is designed to debias the first, typically by upweighting the “hard” samples identified by the initial model. Among these, the methods most similar to our sample score computation scheme are the logit-correction approaches [Liu et al., 2023, Tsirigotis et al., 2023, Bayat et al., 2025], in which the second model is built directly on top of the first model’s predictions, treating them as prior knowledge. The authors of MAT [Bayat et al., 2025] further show that their logit-corrected model assigns similar self-influence scores to majority and minority groups, computed using an external influence estimation method called TRAK [Park et al., 2023]. These prior methods highlight the importance of two-stage modeling, which directly motivates the computation of our Two-Stage Cumulative Sample Loss (TCSL).

Coreset Selection. The goal of coreset selection is to identify a subset of samples that accurately represents the entire dataset, thereby reducing training data size while maintaining model performance [Dharmasiri et al., 2025]. Most coreset selection methods rely on sample scores that quantify the importance or difficulty of individual samples. These scores vary in formulation but are typically functions of the model’s output computed at one or multiple epochs. Commonly used sample scores include EL2N [Paul et al., 2021], Memorization [Feldman, 2020] and SelfSup [Sorscher et al., 2022], all of which have proven effective in various datasets and are widely used as baselines in the literature. Based on these scores, researchers have developed numerous coreset selection strategies, though the optimal selection policy remains an open question. Recent studies [Sorscher et al., 2022, Zheng et al., 2023] have shown that the optimal policy depends on the coreset selection ratio: higher ratios favor pruning “easy” samples, while lower ratios favor pruning “hard” ones. Furthermore, histogram-based selection approaches [Zheng et al., 2023] have gained popularity for their ability to capture the overall data distribution by sampling from both “easy” and “hard” regions. In addition, recent work has explored combining score-based functions with feature similarity metrics to improve coreset coverage. For example, the state-of-the-art D2 pruning method [Maharana et al., 2024] employs a message-passing algorithm to integrate both scoring- and similarity-based importance metrics.

Coreset Selection Meets Spurious Correlations. A growing body of recent work has explored the intersection between coreset selection and learning under spurious correlations. [Mulchandani and Kim, 2025] investigates the setting where group labels (spurious attributes) are available and experimentally demonstrates that removing a small fraction of samples with spurious correlations, particularly those with complex (hard) core features, significantly improves worst-group accuracy. [Vysogorets et al., 2025] proposes a coreset selection algorithm aimed at achieving consistently high accuracy across all classes and further shows that their method can be extended to the group-level setting when group labels are available. Most recently and most closely related to our work, [Dharmasiri et al., 2025] conducts a comprehensive study analyzing how commonly used EL2N and SelfSup scores, and the coreset selection strategies built on them, behave under spurious correlations.

To the best of our knowledge, our method is the first coreset selection algorithm specifically designed to achieve both high average accuracy and high worst-group accuracy, without requiring access to group labels. Our dual objectives are (1) mitigating spurious correlations and (2) reducing training data size. Next, we formalize the problem setting and illustrate why commonly used coreset selection algorithms fail to achieve these objectives.

B THEORETICAL ANALYSIS

In this section, we provide theoretical justifications for the simplicity bias phenomenon by resorting to the Neural Tangent Kernel (NTK) regime [Jacot et al., 2018] and a series of spiked data covariance models [Paul, 2007, Wang et al., 2024b]. This analysis provides a rigorous, dynamic foundation for the claims made in the main paper. We provide a brief introduction to NTK and spiked covariance models at the end of this section to make our theoretical analysis more accessible.

Recall from [Jacot et al., 2018] (here we simplify and adopt the notation to our setup) that the NTK regime emerges in the infinite-width limit of a neural network. The key consequences of this limiting setting are:

1. The network’s output function at initialization $h(x; \mathbf{W}_0)$ (which we refer to as the logit) becomes a draw from a Gaussian Process (Proposition 1 in [Jacot et al., 2018]).
2. The Neural Tangent Kernel $K(x, x'; \mathbf{W}) := \nabla_{\mathbf{W}} h(x; \mathbf{W}) \cdot \nabla_{\mathbf{W}} h(x'; \mathbf{W})$ converges to a deterministic, positive semi-definite kernel $K(x, x')$ that is constant in time (Theorem 1 in [Jacot et al., 2018]).
3. The evolution of the logit outputs $h(x_i; \mathbf{W}_t)$ for the n training samples under gradient flow for the empirical loss $\mathcal{L}(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^n \ell(y_i, h(x_i; \mathbf{W}_t))$ is governed by an exact, deterministic, non-linear Ordinary Differential Equation (ODE) in function space (Theorem 2 in [Jacot et al., 2018]). For a specific logit $h_j(t) \equiv h(x_j; \mathbf{W}_t)$, the dynamic is

$$\frac{\partial h(x_j; \mathbf{W}_t)}{\partial t} = -\frac{1}{n} \sum_{i=1}^n K(x_j, x_i) \frac{\partial \ell(y_i, h(x_i; \mathbf{W}_t))}{\partial h(x_i; \mathbf{W}_t)} \quad (4)$$

To analyze the dynamics, we require a set of foundational assumptions. These are standard in the theoretical analysis of deep learning and are necessary to make the problem analytically tractable.

The following is a standard assumption in NTK, as it is a direct consequence of the Central Limit Theorem applied to wide networks initialized with zero-mean symmetric weights (e.g., Gaussian). It allows us to analyze the deterministic expected dynamics rather than a single stochastic trajectory.

Assumption 1 (Symmetric Initialization). *The initialization distribution $p_{\mathbf{W}}$ is symmetric such that the resulting Gaussian Process $h(x; \mathbf{W}_0)$ has a zero mean function, i.e., $\mathbb{E}_{\mathbf{W}_0}[h(x; \mathbf{W}_0)] = 0$ for all x .*

The following proposition shows that under symmetric initialization, the expected initial gradient of the logistic loss is non-zero and points exactly in the direction of the negative label with a constant factor of $1/2$.

Proposition 1. *Suppose Assumption 1 holds. Let $\ell(y, h) = \log(1 + e^{-yh})$ be the logistic loss. Let $g_i(t) = \frac{\partial \ell(y_i, h(x_i; \mathbf{W}_t))}{\partial h(x_i; \mathbf{W}_t)} = -y_i \sigma(-y_i h(x_i; \mathbf{W}_t))$, where $\sigma(z) = (1 + e^{-z})^{-1}$ is the sigmoid function. Then,*

$$\mathbb{E}_{\mathbf{W}_0}[g_i(0)] = -\frac{1}{2}y_i \quad (5)$$

Proof. Let $Z_i = -y_i h(x_i; \mathbf{W}_0)$. By Assumption 1, $h(x_i; \mathbf{W}_0)$ has a distribution $p(h)$ that is symmetric about zero (i.e., $p(h) = p(-h)$). Thus, the distribution of Z_i , $q(z)$, is likewise symmetric about zero. Recall that the sigmoid function satisfies the identity $\sigma(-z) = 1 - \sigma(z)$. Thus, by linearity of expectation $\mathbb{E}[\sigma(-Z_i)] = 1 - \mathbb{E}[\sigma(Z_i)]$. Since the distribution $q(z)$ is symmetric, the random variables Z_i and $-Z_i$ are identically distributed. Thus, $\mathbb{E}[\sigma(Z_i)] = \mathbb{E}[\sigma(-Z_i)]$. Substituting this into the previous identity yields $\mathbb{E}[\sigma(Z_i)] = 1/2$. Therefore, $\mathbb{E}_{\mathbf{W}_0}[g_i(0)] = -y_i \mathbb{E}_{\mathbf{W}_0}[\sigma(-y_i h(x_i; \mathbf{W}_0))] = -y_i(1/2)$. $\square$

The following assumption is necessary to formally disentangle the learning of core and spurious features. It allows us to analyze their dynamics independently by partitioning the model’s parameters into those that process core features and those that process spurious features.

Assumption 2 (Model Decomposition). *The model architecture additively separates core and spurious features, $h(x; \mathbf{W}) = h_c(x^c; \mathbf{W}^c) + h_s(x^s; \mathbf{W}^s)$, where the parameter sets $\mathbf{W}^c$ and $\mathbf{W}^s$ are disjoint.*

The following proposition formally proves that an additive decomposition of the logit function, combined with disjoint parameters, directly implies an additive decomposition of the Tangent Kernel.

Proposition 2. *Suppose Assumption 2 holds. Then, the NTK is additively decomposable, i.e.,*

$$K(x, x') = K_c(x^c, x'^c) + K_s(x^s, x'^s) \quad (6)$$

where

$$K_c(x, x') = \nabla_{\mathbf{W}^c} h_c(x^c; \mathbf{W}^c) \cdot \nabla_{\mathbf{W}^c} h_c(x'^c; \mathbf{W}^c) \quad (7)$$

is the NTK of the core subnetwork, computed with respect to its own parameters $\mathbf{W}^c$, and

$$K_s(x, x') = \nabla_{\mathbf{W}^s} h_s(x^s; \mathbf{W}^s) \cdot \nabla_{\mathbf{W}^s} h_s(x'^s; \mathbf{W}^s) \quad (8)$$

is the NTK of the spurious subnetwork, computed with respect to its own parameters $\mathbf{W}^s$.

Proof. By definition, $K(x, x'; \mathbf{W}) = \nabla_{\mathbf{W}} h(x; \mathbf{W}) \cdot \nabla_{\mathbf{W}} h(x'; \mathbf{W})$. The total gradient $\nabla_{\mathbf{W}} h(x; \mathbf{W})$ is the concatenation $[\nabla_{\mathbf{W}^c} h(x; \mathbf{W}); \nabla_{\mathbf{W}^s} h(x; \mathbf{W})]$. By Assumption 2, the sub-functions are functionally independent of the other's parameters: $\nabla_{\mathbf{W}^s} h_c(x^c; \mathbf{W}^c) = \mathbf{0}$ and $\nabla_{\mathbf{W}^c} h_s(x^s; \mathbf{W}^s) = \mathbf{0}$. Thus, $\nabla_{\mathbf{W}^c} h(x; \mathbf{W}) = \nabla_{\mathbf{W}^c} (h_c + h_s) = \nabla_{\mathbf{W}^c} h_c(x^c; \mathbf{W}^c)$. Similarly, $\nabla_{\mathbf{W}^s} h(x; \mathbf{W}) = \nabla_{\mathbf{W}^s} h_s(x^s; \mathbf{W}^s)$. The total gradient is $\nabla_{\mathbf{W}} h(x; \mathbf{W}) = [\nabla_{\mathbf{W}^c} h_c; \nabla_{\mathbf{W}^s} h_s]$. Let $g_c(x) = \nabla_{\mathbf{W}^c} h_c(x)$ and $g_s(x) = \nabla_{\mathbf{W}^s} h_s(x)$. The inner product is therefore

$$\begin{aligned} K(x, x') &= [g_c(x); g_s(x)] \cdot [g_c(x'); g_s(x')] \\ &= g_c(x) \cdot g_c(x') + g_s(x) \cdot g_s(x') \\ &= K_c(x, x') + K_s(x, x'), \end{aligned}$$

which completes the proof. $\square$

The following assumption formally defines the setting of spurious correlation, where the dataset is imbalanced such that the spurious attribute a_i is predictive of the true label y_i for a fraction $\alpha > 1/2$ of the data.

Assumption 3 (Data Structure). *The dataset $\mathcal{D}$ of size n is partitioned into $G_1 = \{i : y_i = a_i\}$ of size $n_1 = \alpha n$ and $G_2 = \{i : y_i = -a_i\}$ of size $n_2 = (1 - \alpha)n$, with $\alpha \in (1/2, 1]$.*

B.1 HOMOGENEOUS SPIKED MODEL

We begin with a simple model that assumes all samples have uniform feature strength. The strength of this setting is in its simple statements, which we find valuable for pedagogical reasons.

The following assumption is inspired by the Spiked Covariance Model from high-dimensional statistics. It provides an idealized and analytically simple model to isolate the competition between the core signal (strength β^c) and the spurious signal (strength β^s).

Assumption 4 (Homogeneous Spiked Model). *The NTK components are perfectly aligned with the latent data structure and have a uniform, rank-1 spiked structure:*

$$K_c(x_j^c, x_i^c) = \beta^c y_j y_i, \quad \beta^c > 0 \quad (9)$$

and

$$K_s(x_j^s, x_i^s) = \beta^s a_j a_i, \quad \beta^s > 0 \quad (10)$$

The following theorem provides the exact initial velocities of the expected logits. It shows that the core subnetwork always learns in the direction of the true label, while the spurious subnetwork learns in the direction of the spurious attribute, with a velocity amplified by the data imbalance $2\alpha - 1$.

Theorem 0 (Initial Velocity in Homogeneous Spiked Model). *Define the expected logit $\bar{h}_t(x) = \mathbb{E}_{\mathbf{W}_0}[h(x; \mathbf{W}_t)]$ and let Assumptions 1-4 hold. Then, the initial velocities of the expected subnetworks are*

$$\left. \frac{\partial \bar{h}_t^c(x_j)}{\partial t} \right|_{t=0} = \frac{\beta^c}{2} y_j, \quad (11)$$

and

$$\left. \frac{\partial \bar{h}_t^s(x_j)}{\partial t} \right|_{t=0} = \frac{\beta^s(2\alpha - 1)}{2} a_j \quad (12)$$

where $\bar{h}_t^c(x) = \mathbb{E}_{\mathbf{W}_0}[h_c(x; \mathbf{W}_t)]$ and $\bar{h}_t^s(x) = \mathbb{E}_{\mathbf{W}_0}[h_s(x; \mathbf{W}_t)]$.

Proof. Let $\bar{h}_{t,j} = \bar{h}_t(x_j)$. The dynamics of $\bar{h}_{t,j}$ are $\frac{\partial \bar{h}_{t,j}}{\partial t} = \mathbb{E}_{\mathbf{W}_0} \left[\frac{\partial h(x_j; \mathbf{W}_t)}{\partial t} \right]$. At $t = 0$

$$\left. \frac{\partial \bar{h}_{t,j}}{\partial t} \right|_{t=0} = \mathbb{E}_{\mathbf{W}_0} \left[-\frac{1}{n} \sum_{i=1}^n K(x_j, x_i) g_i(0) \right] \quad (13)$$

By linearity of expectation and the fact that K is deterministic

$$\left. \frac{\partial \bar{h}_{t,j}}{\partial t} \right|_{t=0} = -\frac{1}{n} \sum_{i=1}^n K(x_j, x_i) \mathbb{E}_{\mathbf{W}_0}[g_i(0)] \quad (14)$$

Leveraging the result of Proposition 1

$$\begin{aligned} \left. \frac{\partial \bar{h}_{t,j}}{\partial t} \right|_{t=0} &= -\frac{1}{n} \sum_{i=1}^n K(x_j, x_i) \left(-\frac{y_i}{2} \right) \\ &= \frac{1}{2n} \sum_{i=1}^n K(x_j, x_i) y_i. \end{aligned} \quad (15)$$

Using Proposition 2, we analyze the subnetworks. For the core subnetwork

$$\begin{aligned} \left. \frac{\partial \bar{h}_t^c(x_j)}{\partial t} \right|_{t=0} &= \frac{1}{2n} \sum_{i=1}^n K_c(x_j^c, x_i^c) y_i \\ &= \frac{1}{2n} \sum_{i=1}^n (\beta^c y_j y_i) y_i \\ &= \frac{\beta^c y_j}{2n} \sum_{i=1}^n y_i^2 \\ &= \frac{\beta^c}{2} y_j, \end{aligned} \quad (16)$$

where we used Assumption 4 and $y_i^2 = 1$. Similarly, for the spurious subnetwork

$$\begin{aligned} \left. \frac{\partial \bar{h}_t^s(x_j)}{\partial t} \right|_{t=0} &= \frac{1}{2n} \sum_{i=1}^n K_s(x_j^s, x_i^s) y_i \\ &= \frac{1}{2n} \sum_{i=1}^n (\beta^s a_j a_i) y_i \\ &= \frac{\beta^s a_j}{2n} \sum_{i=1}^n a_i y_i. \end{aligned} \quad (17)$$

By Assumption 3, $\sum_{i=1}^n a_i y_i = n_1 - n_2 = n(2\alpha - 1)$. This yields $\frac{\beta^s a_j}{2n} (n(2\alpha - 1)) = \frac{\beta^s(2\alpha - 1)}{2} a_j$. $\square$

The following corollary provides the precise condition for simplicity bias in the homogeneous model. It is a competition between the core feature strength β^c and the spurious feature strength β^s modulated by the data imbalance $2\alpha - 1$.

Corollary 1. *Let the expected initial growth rate of the true core margin be $R_c := \left. \frac{\partial}{\partial t} \mathbb{E}_{\mathbf{W}_0}[y_j h_c(x_j; \mathbf{W}_t)] \right|_{t=0}$. Let the expected initial growth rate of the true spurious margin be $R_s := \left. \frac{\partial}{\partial t} \mathbb{E}_{\mathbf{W}_0}[a_j h_s(x_j; \mathbf{W}_t)] \right|_{t=0}$. Then $R_c = \beta^c/2$ and $R_s = \frac{\beta^s(2\alpha - 1)}{2}$. The model exhibits simplicity bias ($R_s > R_c$) if and only if $\beta^s(2\alpha - 1) > \beta^c$.*

The following theorem corresponds to Theorem 1 in the main paper. We demonstrate the direct consequence of the velocity imbalance. At the start of training, the expected loss on the majority group immediately decreases, while the expected loss on the minority group immediately increases, demonstrating the simplicity bias.

Theorem 1 (Initial Loss Divergence in Homogeneous Spiked Model). *Let the simplicity bias condition $\beta^s(2\alpha - 1) > \beta^c$ from Corollary 1 hold. Let $\bar{m}_t(x_j) = \mathbb{E}_{\mathbf{W}_0}[y_j h(x_j; \mathbf{W}_t)]$ be the expected margin. Then, there exists a time $T > 0$ such that for all $t \in (0, T)$:*

1. *For $j \in G_1$ (majority group, $y_j = a_j$), the expected margin $\bar{m}_t(x_j)$ is positive, and the loss $\ell(\bar{m}_t(x_j))$ is less than $\log(2)$.*
2. *For $j \in G_2$ (minority group, $y_j = -a_j$), the expected margin $\bar{m}_t(x_j)$ is negative, and the loss $\ell(\bar{m}_t(x_j))$ is greater than $\log(2)$.*

Proof. By Assumption 1, the initial expected logit is $\bar{h}_0(x_j) = 0$, so the initial expected margin is $\bar{m}_0(x_j) = 0$. We compute the initial time-derivative of the expected margin

$$\left. \frac{\partial \bar{m}_t(x_j)}{\partial t} \right|_{t=0} = y_j \left(\left. \frac{\partial \bar{h}_t^c(x_j)}{\partial t} \right|_{t=0} + \left. \frac{\partial \bar{h}_t^s(x_j)}{\partial t} \right|_{t=0} \right)$$

Using Theorem 0 and $y_j^2 = 1$

$$\begin{aligned} \left. \frac{\partial \bar{m}_t(x_j)}{\partial t} \right|_{t=0} &= y_j \left(\frac{\beta^c}{2} y_j + \frac{\beta^s(2\alpha - 1)}{2} a_j \right) \\ &= \frac{\beta^c}{2} y_j^2 + \frac{\beta^s(2\alpha - 1)}{2} (y_j a_j) \\ &= \frac{\beta^c}{2} + \frac{\beta^s(2\alpha - 1)}{2} (y_j a_j). \end{aligned} \tag{18}$$

Note that the loss function $\ell(\bar{m}) = \log(1 + e^{-\bar{m}})$ is strictly monotonically decreasing in $\bar{m}$.

1. For $j \in G_1$, $y_j a_j = 1$. The initial velocity is $R_c + R_s = \frac{\beta^c}{2} + \frac{\beta^s(2\alpha - 1)}{2} > 0$. Since $\bar{m}_0(x_j) = 0$ and $\left. \frac{\partial \bar{m}_t(x_j)}{\partial t} \right|_{t=0} > 0$, there exists $T_1 > 0$ such that $\bar{m}_t(x_j) > 0$ for $t \in (0, T_1)$. Thus, $\ell(\bar{m}_t(x_j)) < \ell(0) = \log(2)$.
2. For $j \in G_2$, $y_j a_j = -1$. The initial velocity is $R_c - R_s = \frac{\beta^c}{2} - \frac{\beta^s(2\alpha - 1)}{2} < 0$ by the simplicity bias condition. Since $\bar{m}_0(x_j) = 0$ and $\left. \frac{\partial \bar{m}_t(x_j)}{\partial t} \right|_{t=0} < 0$, there exists $T_2 > 0$ such that $\bar{m}_t(x_j) < 0$ for $t \in (0, T_2)$. Thus, $\ell(\bar{m}_t(x_j)) > \ell(0) = \log(2)$.

Let $T = \min(T_1, T_2)$. For $t \in (0, T)$, both statements hold. $\square$

The following theorem characterizes the initial curvature of the learning path. It shows how the model begins to decelerate, or saturate, as a function of the feature strengths. We need to define some notation and make an approximation.

Let $\mathbf{h}_t = [h(x_1; \mathbf{W}_t), \dots, h(x_n; \mathbf{W}_t)]^\top$ be the vector of logits, and let $\mathbf{g}_t = [g_1(t), \dots, g_n(t)]^\top$. The flow ODE in vector form is $\dot{\mathbf{h}}_t = -\frac{1}{n} K \mathbf{g}_t$. Let $\bar{\mathbf{h}}_t = \mathbb{E}[\mathbf{h}_t]$. The expected dynamics are $\dot{\bar{\mathbf{h}}}_t = -\frac{1}{n} K \mathbb{E}[\mathbf{g}_t]$. From Theorem 0, we have $\dot{\bar{\mathbf{h}}}_0 = \frac{1}{2n} K \mathbf{y}$. The expected acceleration is $\ddot{\bar{\mathbf{h}}}_t = \frac{\partial}{\partial t} \dot{\bar{\mathbf{h}}}_t = -\frac{1}{n} K \mathbb{E}[\dot{\mathbf{g}}_t]$. Element-wise, $\dot{g}_i(t) = \frac{\partial g_i(t)}{\partial h_i(t)} \dot{h}_i(t) = \sigma'(-y_i h_i(t)) \dot{h}_i(t)$. We will approximate by linearization $\mathbb{E}[\sigma'(-y_i h_i(0))] \approx \sigma'(-y_i \bar{h}_i(0)) = \sigma'(0) = 1/4$, leading to $\mathbb{E}[\dot{\mathbf{g}}_0] \approx \frac{1}{4} \dot{\bar{\mathbf{h}}}_0$.

Theorem 2 (Initial Acceleration in Homogeneous Spiked Model). *Let $\bar{a}_{0,j} := \left. \frac{\partial^2 \bar{h}_t(x_j)}{\partial t^2} \right|_{t=0}$ denote the initial acceleration of the expected logit for sample j . Under Assumptions 1-4, the total acceleration is*

$$\bar{a}_{0,j} = -\frac{1}{8} \left[((\beta^c)^2 + \beta^c \beta^s (2\alpha - 1)^2) y_j + (2\alpha - 1) (\beta^c \beta^s + (\beta^s)^2) a_j \right] \tag{19}$$

The subnetwork accelerations are

$$\bar{a}_{0,j}^c = -\frac{1}{8} [(\beta^c)^2 + \beta^c \beta^s (2\alpha - 1)^2] y_j \tag{20}$$

$$\bar{a}_{0,j}^s = -\frac{1}{8} (2\alpha - 1) [\beta^c \beta^s + (\beta^s)^2] a_j \tag{21}$$

Proof. Adopting the approximation described in the preceding discussion,

$$\ddot{\mathbf{h}}_0 \approx -\frac{1}{4n} K \dot{\mathbf{h}}_0 \quad (22)$$

Substituting $\dot{\mathbf{h}}_0 = \frac{1}{2n} K \mathbf{y}$ we have

$$\ddot{\mathbf{h}}_0 \approx -\frac{1}{8n^2} K^2 \mathbf{y} \quad (23)$$

We compute $K^2 \mathbf{y}$ using $K = \beta^c \mathbf{y} \mathbf{y}^\top + \beta^s \mathbf{a} \mathbf{a}^\top$ and the inner products $\mathbf{y}^\top \mathbf{y} = n$, $\mathbf{a}^\top \mathbf{a} = n$, and $\mathbf{y}^\top \mathbf{a} = n(2\alpha - 1)$. We have $K \mathbf{y} = \beta^c \mathbf{y} (\mathbf{y}^\top \mathbf{y}) + \beta^s \mathbf{a} (\mathbf{a}^\top \mathbf{y}) = n\beta^c \mathbf{y} + n\beta^s (2\alpha - 1) \mathbf{a}$. Also, $K \mathbf{a} = \beta^c \mathbf{y} (\mathbf{y}^\top \mathbf{a}) + \beta^s \mathbf{a} (\mathbf{a}^\top \mathbf{a}) = n\beta^c (2\alpha - 1) \mathbf{y} + n\beta^s \mathbf{a}$.

$$\begin{aligned} K^2 \mathbf{y} &= K (n\beta^c \mathbf{y} + n\beta^s (2\alpha - 1) \mathbf{a}) \\ &= n\beta^c (K \mathbf{y}) + n\beta^s (2\alpha - 1) (K \mathbf{a}) \\ &= n\beta^c [n\beta^c \mathbf{y} + n\beta^s (2\alpha - 1) \mathbf{a}] + n\beta^s (2\alpha - 1) [n\beta^c (2\alpha - 1) \mathbf{y} + n\beta^s \mathbf{a}] \\ &= [n^2 (\beta^c)^2 + n^2 \beta^c \beta^s (2\alpha - 1)^2] \mathbf{y} + [n^2 \beta^c \beta^s (2\alpha - 1) + n^2 (\beta^s)^2 (2\alpha - 1)] \mathbf{a}. \end{aligned}$$

Dividing by $-8n^2$ gives the total acceleration $\bar{a}_{0,j}$. The subnetwork accelerations $\bar{a}_{0,j}^c = -\frac{1}{4n} (K_c \dot{\mathbf{h}}_0)_j$ and $\bar{a}_{0,j}^s = -\frac{1}{4n} (K_s \dot{\mathbf{h}}_0)_j$ follow by isolating the respective kernel terms. $\square$

The following corollary breaks down the acceleration by group. The majority group, with aligned signals, always decelerates. The minority group, with conflicting signals, can accelerate if the spurious signal is strong. When both decelerate, the majority group decelerates faster.

Corollary 2 (Group-Specific Acceleration Dynamics). *Let A_{maj} and A_{min} be the scalar acceleration of the margin ($A_j = y_j \bar{a}_{0,j}$) for the majority ($j \in G_1$) and minority ($j \in G_2$) groups, respectively.*

1. **Universal Majority Deceleration:** *The majority group always decelerates (saturates):*

$$A_{maj} = -\frac{1}{8} \left[(\beta^c)^2 + \beta^c \beta^s (2\alpha - 1)^2 + (2\alpha - 1) (\beta^c \beta^s + (\beta^s)^2) \right] < 0$$

2. **Minority Acceleration Regime:** *The minority group accelerates ($A_{min} > 0$) if the spurious feature is sufficiently strong:*

$$(2\alpha - 1) (\beta^s)^2 + (2\alpha - 1) \beta^c \beta^s > (\beta^c)^2 + \beta^c \beta^s (2\alpha - 1)^2 \quad (24)$$

3. **Relative Saturation Strength:** *In the regime where $A_{min} < 0$ (both groups decelerate), the majority group decelerates with a strictly greater magnitude:*

$$|A_{maj}| > |A_{min}| \quad (25)$$

B.2 HETEROGENEOUS SPIKED MODEL

We now relax the assumption of uniform feature strength, allowing each sample to have its own strength. This model is more realistic as it allows for “hard” (low β) and “easy” (high β) samples within each group, but retains a rank-1 structure that permits exact closed-form solutions.

Assumption 5 (Heterogeneous Spiked Model). *The NTK components decompose into rank-1 matrices based on sample-specific strengths:*

$$\begin{aligned} K_c(x_j, x_i) &= (\beta_i^c \beta_j^c) y_i y_j \\ K_s(x_j, x_i) &= (\beta_i^s \beta_j^s) a_i a_j, \end{aligned} \quad (26)$$

where $\beta_i^c > 0$ and $\beta_i^s > 0$ are the core and spurious strengths of sample i . We assume these are bounded, $b^c \leq \beta_i^c \leq B^c$ and $b^s \leq \beta_i^s \leq B^s$.

Definition 1 (Global Dataset Statistics). *We define the following scalar summaries of the dataset's feature structure:*

$$S_{cc} = \sum_{i=1}^n \beta_i^c \quad (27)$$

$$S_{sc} = \sum_{i=1}^n \beta_i^s a_i y_i = \sum_{i \in G_1} \beta_i^s - \sum_{i \in G_2} \beta_i^s \quad (28)$$

$$Q_{cc} = \sum_{i=1}^n (\beta_i^c)^2 \quad (29)$$

$$Q_{ss} = \sum_{i=1}^n (\beta_i^s)^2 \quad (30)$$

$$Q_{cs} = \sum_{i=1}^n \beta_i^c \beta_i^s a_i y_i = \sum_{i \in G_1} \beta_i^c \beta_i^s - \sum_{i \in G_2} \beta_i^c \beta_i^s \quad (31)$$

Further, we define the global energy constants

$$\begin{aligned} C_{core} &= S_{cc}Q_{cc} + S_{sc}Q_{cs}, \\ C_{spur} &= S_{cc}Q_{cs} + S_{sc}Q_{ss}. \end{aligned} \quad (32)$$

These scalar quantities summarize the aggregate “global” properties of the dataset under the heterogeneous Spiked model. Note that S_{sc} and Q_{cs} are positive by Assumption 3.

The following theorem shows that the initial velocity for a sample is a product of its local feature strength (β_k) and the global alignment of the entire dataset (S_{cc} or S_{sc}).

Theorem 3 (Initial Velocities in Heterogeneous Spiked Model). *Let $\bar{h}_t(x_k)$ be the expected logit for sample k . Under Assumptions 1-3 and 5, the initial velocities of the subnetwork logits are*

$$\begin{aligned} \left. \frac{\partial \bar{h}_t^c(x_k)}{\partial t} \right|_{t=0} &= \frac{S_{cc}}{2n} \beta_k^c y_k, \\ \left. \frac{\partial \bar{h}_t^s(x_k)}{\partial t} \right|_{t=0} &= \frac{S_{sc}}{2n} \beta_k^s a_k. \end{aligned} \quad (33)$$

Proof. Let $\mathbf{u}_c \in \mathbb{R}^n$ have entries $(\mathbf{u}_c)_i = \beta_i^c y_i$ and $\mathbf{u}_s \in \mathbb{R}^n$ have entries $(\mathbf{u}_s)_i = \beta_i^s a_i$. Then $K_c = \mathbf{u}_c \mathbf{u}_c^\top$ and $K_s = \mathbf{u}_s \mathbf{u}_s^\top$. From the proof of Theorem 0, $\dot{\mathbf{h}}_0 = \frac{1}{2n} K \mathbf{y}$. $\dot{\mathbf{h}}_0^c = \frac{1}{2n} (\mathbf{u}_c \mathbf{u}_c^\top) \mathbf{y} = \frac{1}{2n} \mathbf{u}_c (\mathbf{u}_c^\top \mathbf{y}) = \frac{1}{2n} \mathbf{u}_c (\sum_i \beta_i^c y_i^2) = \frac{S_{cc}}{2n} \mathbf{u}_c$. $\dot{\mathbf{h}}_0^s = \frac{1}{2n} (\mathbf{u}_s \mathbf{u}_s^\top) \mathbf{y} = \frac{1}{2n} \mathbf{u}_s (\mathbf{u}_s^\top \mathbf{y}) = \frac{1}{2n} \mathbf{u}_s (\sum_i \beta_i^s a_i y_i) = \frac{S_{sc}}{2n} \mathbf{u}_s$. The k -th entry of $\dot{\mathbf{h}}_0^c$ is $\frac{S_{cc}}{2n} \beta_k^c y_k$, and for $\dot{\mathbf{h}}_0^s$ it is $\frac{S_{sc}}{2n} \beta_k^s a_k$. $\square$

The following corollary refines the simplicity bias condition. It is now a sample-specific condition that depends on the ratio of the sample's local strengths (β_k^s/β_k^c) versus the ratio of the dataset's global alignments (S_{cc}/S_{sc}). Interestingly, this explains why some minority samples might be learned while others (with high β_k^s) are not.

Corollary 3 (Sample-Wise Simplicity Bias). *The margin velocity for sample k is $\dot{m}_k(0) = \frac{1}{2n} (S_{cc} \beta_k^c + S_{sc} \beta_k^s (y_k a_k))$. For a minority sample ($k \in G_2$, $y_k a_k = -1$), the loss increases at initialization if:*

$$\beta_k^s S_{sc} > \beta_k^c S_{cc} \quad (34)$$

Theorem 4 (Initial Acceleration in Heterogeneous Spiked Model). *Recall the global energy constants from Definition 1*

$$\begin{aligned} C_{core} &= S_{cc}Q_{cc} + S_{sc}Q_{cs}, \\ C_{spur} &= S_{cc}Q_{cs} + S_{sc}Q_{ss}. \end{aligned} \quad (35)$$

The initial acceleration of the expected logit for sample k is

$$\bar{a}_{0,k} = -\frac{1}{8n^2} [\beta_k^c C_{core} y_k + \beta_k^s C_{spur} a_k] \quad (36)$$

Proof. We start with the approximation $\ddot{\mathbf{h}}_0 \approx -\frac{1}{4n}K\dot{\mathbf{h}}_0$. From Theorem 3, $\dot{\mathbf{h}}_0 = \frac{1}{2n}(S_{cc}\mathbf{u}_c + S_{sc}\mathbf{u}_s)$.

$$\begin{aligned}\ddot{\mathbf{h}}_0 &\approx -\frac{1}{8n^2}(\mathbf{u}_c\mathbf{u}_c^\top + \mathbf{u}_s\mathbf{u}_s^\top)(S_{cc}\mathbf{u}_c + S_{sc}\mathbf{u}_s) \\ &= -\frac{1}{8n^2}\left[\mathbf{u}_c(S_{cc}(\mathbf{u}_c^\top\mathbf{u}_c) + S_{sc}(\mathbf{u}_c^\top\mathbf{u}_s)) + \mathbf{u}_s(S_{cc}(\mathbf{u}_s^\top\mathbf{u}_c) + S_{sc}(\mathbf{u}_s^\top\mathbf{u}_s))\right].\end{aligned}$$

Using the definition of inner product and the quantities defined in Definition 1 and the proof of Theorem 3 we have $\mathbf{u}_c^\top\mathbf{u}_c = Q_{cc}$, $\mathbf{u}_s^\top\mathbf{u}_s = Q_{ss}$, and $\mathbf{u}_c^\top\mathbf{u}_s = \mathbf{u}_s^\top\mathbf{u}_c = Q_{cs}$ leading to

$$\ddot{\mathbf{h}}_0 \approx -\frac{1}{8n^2}\left[\mathbf{u}_c(S_{cc}Q_{cc} + S_{sc}Q_{cs}) + \mathbf{u}_s(S_{cc}Q_{cs} + S_{sc}Q_{ss})\right]. \quad (37)$$

Substituting the definitions of C_{core} , C_{spur} , $\mathbf{u}_c$, and $\mathbf{u}_s$ yields the element-wise result for $\bar{a}_{0,k}$ stated in the Theorem. $\square$

Note that the initial acceleration is also a product of local sample strengths (β_k) and the global energy constants (C_{core} , C_{spur}) that depend on the aggregate statistics of the dataset.

The following corollary demonstrates that the qualitative dynamics hold. Majority samples always decelerate. A minority sample's acceleration depends on its specific β_k^s vs β_k^c profile, explaining intra-group variance. When all samples decelerate, majority samples do so more rapidly.

Corollary 4 (Group-Specific Acceleration in Heterogeneous Spiked Model). *Let $A_k = y_k\bar{a}_{0,k}$ be the margin acceleration for sample k .*

1. **Universal Majority Deceleration:** For any majority sample $k \in G_1$ ($y_k = a_k$):

$$A_k = -\frac{1}{8n^2}(\beta_k^c C_{core} + \beta_k^s C_{spur}) < 0 \quad (38)$$

(Assuming $S_{sc}, Q_{cs} > 0$).

2. **Minority Acceleration Regime:** For a minority sample $k \in G_2$ ($y_k = -a_k$), the acceleration is:

$$A_k = -\frac{1}{8n^2}(\beta_k^c C_{core} - \beta_k^s C_{spur}) \quad (39)$$

The sample accelerates ($A_k > 0$) if its spurious-weighted energy contribution exceeds its core-weighted one:

$$\beta_k^s C_{spur} > \beta_k^c C_{core} \quad (40)$$

3. **Relative Saturation Strength:** In the regime where a minority sample also decelerates ($A_k < 0$, i.e., $\beta_k^c C_{core} > \beta_k^s C_{spur}$), its deceleration is strictly weaker than that of a majority sample with the same local strengths β_k^c, β_k^s :

$$|A_{k \in G_1}| - |A_{k \in G_2}| = \frac{1}{4n^2}\beta_k^s C_{spur} > 0 \quad (41)$$

B.3 VALIDATION OF OUR THEORETICAL SETTING AND ASSUMPTIONS

B.3.1 Analysis on the Linearization in Theorem 2

We establish guarantees for the linearization approximation $\mathbb{E}_{\mathbf{W}_0}[\sigma'(-y_i h(x_i; \mathbf{W}_0))] \approx \sigma'(0) = 1/4$ utilized in the derivation of the initial expected acceleration. In the infinite-width NTK limit under symmetric initialization, the network output $h(x_i; \mathbf{W}_0)$ converges in distribution to a centered Gaussian process. Consequently, the random variable $Z_i = -y_i h(x_i; \mathbf{W}_0)$ follows a Gaussian distribution $\mathcal{N}(0, \sigma_h^2)$, where $\sigma_h^2 = K(x_i, x_i)$ denotes the initial function space variance.

Proposition 3. *Let $Z \sim \mathcal{N}(0, \sigma_h^2)$. The absolute error induced by the linearization approximation evaluated at initialization is bounded by*

$$0 \leq \frac{1}{4} - \mathbb{E}[\sigma'(Z)] \leq \frac{\sigma_h^2}{16} \quad (42)$$

Proof. The derivative of the logistic sigmoid function $\sigma'(z) = (e^z + e^{-z} + 2)^{-1}$ achieves its global maximum at $z = 0$ with $\sigma'(0) = 1/4$. Thus, the lower bound $\frac{1}{4} - \mathbb{E}[\sigma'(Z)] \geq 0$ holds trivially. To establish the upper bound, we observe that the function $g(z) = \frac{1}{4} - \sigma'(z)$ satisfies $g(0) = 0$, $g'(0) = 0$, and its second derivative is globally bounded above by $1/8$. Integrating the second derivative yields the global parabolic upper bound

$$\frac{1}{4} - \sigma'(z) \leq \frac{z^2}{16} \quad (43)$$

Taking the expectation over the Gaussian measure of Z directly provides the upper bound $\frac{1}{16} \mathbb{E}[Z^2] = \frac{\sigma_h^2}{16}$. $\square$

Let us now quantify the initial function space variance σ_h^2 under standard parameterizations. Consider a fully connected neural network architecture of depth L with input dimension d_0 and hidden widths d_l for $l \in \{1, \dots, L\}$.

Let $x \in \mathbb{R}^{d_0}$ represent the input vector. The forward pass is defined recursively

$$z^{(l)} = W^{(l)} a^{(l-1)} \quad (44)$$

$$a^{(l)} = \phi(z^{(l)}) \quad (45)$$

where $a^{(0)} = x$, $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ is the weight matrix at layer l , and $\phi(\cdot)$ is the element-wise activation function. The final scalar output is given by the linear transformation

$$h(x; \mathbf{W}_0) = W^{(L+1)} a^{(L)} \quad (46)$$

with $W^{(L+1)} \in \mathbb{R}^{1 \times d_L}$.

Proposition 4. Assume the network weights are initialized according to the He initialization protocol for the ReLU activation function $\phi(z) = \max(0, z)$. Specifically $W_{ij}^{(l)} \sim \mathcal{N}\left(0, \frac{2}{d_{l-1}}\right)$ for $l \in \{1, \dots, L\}$ and the final linear layer is initialized as $W_{1j}^{(L+1)} \sim \mathcal{N}\left(0, \frac{1}{d_L}\right)$. The initial variance of the network output is exactly

$$\sigma_h^2 = \frac{\|x\|_2^2}{d_0} \quad (47)$$

Proof. We proceed by induction on the second moment of the activations. For the first hidden layer the pre-activation components are $z_i^{(1)} = \sum_{j=1}^{d_0} W_{ij}^{(1)} x_j$. Since $W_{ij}^{(1)}$ are independent zero-mean Gaussian random variables we have

$$\mathbb{E} \left[\left(z_i^{(1)} \right)^2 \right] = \sum_{j=1}^{d_0} \mathbb{E} \left[\left(W_{ij}^{(1)} \right)^2 \right] x_j^2 = \frac{2}{d_0} \sum_{j=1}^{d_0} x_j^2 = \frac{2\|x\|_2^2}{d_0} \quad (48)$$

Given that $z_i^{(1)}$ follows a symmetric zero-mean normal distribution the ReLU activation halves the expected squared magnitude

$$\mathbb{E} \left[\left(a_i^{(1)} \right)^2 \right] = \frac{1}{2} \mathbb{E} \left[\left(z_i^{(1)} \right)^2 \right] = \frac{\|x\|_2^2}{d_0} \quad (49)$$

Assume the inductive hypothesis holds for layer $l-1$ such that $\mathbb{E} \left[\left(a_j^{(l-1)} \right)^2 \right] = \frac{\|x\|_2^2}{d_0}$. For layer l the weights and previous activations are independent

$$\mathbb{E} \left[\left(z_i^{(l)} \right)^2 \right] = \sum_{j=1}^{d_{l-1}} \mathbb{E} \left[\left(W_{ij}^{(l)} \right)^2 \right] \mathbb{E} \left[\left(a_j^{(l-1)} \right)^2 \right] = d_{l-1} \left(\frac{2}{d_{l-1}} \right) \left(\frac{\|x\|_2^2}{d_0} \right) = \frac{2\|x\|_2^2}{d_0} \quad (50)$$

Applying the ReLU properties again yields $\mathbb{E} \left[\left(a_i^{(l)} \right)^2 \right] = \frac{\|x\|_2^2}{d_0}$. By induction this holds for the final hidden layer L . The scalar output evaluates the linear combination with the final weight vector

$$\sigma_h^2 = \mathbb{E} \left[\left(h(x; \mathbf{W}_0) \right)^2 \right] = \sum_{j=1}^{d_L} \mathbb{E} \left[\left(W_{1j}^{(L+1)} \right)^2 \right] \mathbb{E} \left[\left(a_j^{(L)} \right)^2 \right] \quad (51)$$

Substituting the variance of the final layer $W_{1j}^{(L+1)} \sim \mathcal{N}\left(0, \frac{1}{d_L}\right)$ directly yields

$$\sigma_h^2 = d_L \left(\frac{1}{d_L}\right) \left(\frac{\|x\|_2^2}{d_0}\right) = \frac{\|x\|_2^2}{d_0} \quad (52)$$

□

A similar result holds for the Xavier initialization under which $W_{ij}^{(l)} \sim \mathcal{N}\left(0, \frac{1}{d_{l-1}}\right)$ and $\phi(z) = z$ corresponding to the linear regime of symmetric activations near the origin. In this case, a simpler argument leveraging that the variance propagation trivially preserves the second moment at each layer yields $\sigma_h^2 = \frac{\|x\|_2^2}{d_0}$.

We can now substitute this exact variance derivation into the theoretical bound established previously to obtain

$$0 \leq \frac{1}{4} - \mathbb{E}_{\mathbf{W}_0}[\sigma'(-y_i h(x_i; \mathbf{W}_0))] \leq \frac{\|x_i\|_2^2}{16d_0}. \quad (53)$$

This bound thus demonstrates that the quality of the linearization approximation is dictated by the input dimension. In standard computer vision datasets the input space dimension d_0 is adequately large (around 50K for the ones used in our paper). Assuming a data preprocessing that standardizes the input vectors, i.e. $\|x_i\|_2^2 = 1$, the absolute error scales as $\mathcal{O}(d_0^{-1})$. The error term thus converges to zero as $d_0 \rightarrow \infty$ certifying the soundness of evaluating the expected acceleration dynamics using $\sigma'(0) = 1/4$. This calculation then justifies the approximation at $t = 0$ we leveraged in our original theoretical result.

We proceed to address the degradation of this approximation as the optimization progresses for $t > 0$ by characterizing the Jensen's gap (which is an intuitive approach to compare $\mathbb{E}[\sigma'(Z_t)]$ vs. $\sigma'(\mathbb{E}[Z_t])$ when applying the Jensen's inequality). As the model fits the training data, the expected margin diverges from zero. Let $\mu_t = \mathbb{E}[-y_i h(x_i; \mathbf{W}_t)]$. For samples successfully classified by the network, μ_t diverges towards $-\infty$.

Proposition 5. *Assume the logit $Z_t = -y_i h(x_i; \mathbf{W}_t)$ maintains a Gaussian distribution $\mathcal{N}(\mu_t, \sigma_h^2)$ during gradient flow. As the expected margin grows such that $\mu_t \rightarrow -\infty$, the Jensen gap between the expected acceleration and the point estimate converges to a constant multiplicative factor dictated by the variance*

$$\lim_{\mu_t \rightarrow -\infty} \frac{\mathbb{E}[\sigma'(Z_t)]}{\sigma'(\mathbb{E}[Z_t])} = \exp\left(\frac{\sigma_h^2}{2}\right) \quad (54)$$

Proof. The derivative of the logistic sigmoid function is $\sigma'(z) = \frac{e^z}{(1+e^z)^2}$. We express the random variable as $Z_t = \mu_t + \sigma_h X$ where $X \sim \mathcal{N}(0, 1)$. Then

$$\frac{\mathbb{E}[\sigma'(Z_t)]}{\sigma'(\mu_t)} = \frac{\mathbb{E}\left[\frac{e^{\mu_t + \sigma_h X}}{(1+e^{\mu_t + \sigma_h X})^2}\right]}{\frac{e^{\mu_t}}{(1+e^{\mu_t})^2}} = \mathbb{E}\left[e^{\sigma_h X} \frac{(1+e^{\mu_t})^2}{(1+e^{\mu_t + \sigma_h X})^2}\right] \quad (55)$$

Let $g(\mu_t, X) = e^{\sigma_h X} \frac{(1+e^{\mu_t})^2}{(1+e^{\mu_t + \sigma_h X})^2}$. For any fixed $X \in \mathbb{R}$, taking the limit yields

$$\lim_{\mu_t \rightarrow -\infty} g(\mu_t, X) = e^{\sigma_h X} \frac{(1+0)^2}{(1+0)^2} = e^{\sigma_h X} \quad (56)$$

We need to pass the limit inside the expectation. For all $\mu_t \leq 0$ and $X \in \mathbb{R}$, we have $(1+e^{\mu_t})^2 \leq 4$ and $(1+e^{\mu_t + \sigma_h X})^2 \geq 1$. Consequently,

$$|g(\mu_t, X)| \leq 4e^{\sigma_h X} \quad (57)$$

The dominating function $4e^{\sigma_h X}$ is integrable with respect to the standard Gaussian measure, as its expectation corresponds to a scaled log-normal moment

$$\mathbb{E}[4e^{\sigma_h X}] = 4 \exp\left(\frac{\sigma_h^2}{2}\right) < \infty \quad (58)$$

By Lebesgue's Dominated Convergence Theorem, the limit of the expectation is

$$\lim_{\mu_t \rightarrow -\infty} \mathbb{E}[g(\mu_t, X)] = \mathbb{E}\left[\lim_{\mu_t \rightarrow -\infty} g(\mu_t, X)\right] = \mathbb{E}[e^{\sigma_h X}] \quad (59)$$

Evaluating the moment generating function of the standard normal distribution at σ_h completes the proof:

$$\mathbb{E}[e^{\sigma_h X}] = \exp\left(\frac{\sigma_h^2}{2}\right). \quad (60)$$

□

Given that we argued $\sigma_h^2 = \mathcal{O}(\frac{\|x\|_2^2}{d_0})$ previously, we expect the approximation does not degrade significantly as training progresses.

The requirements for the above proposition are further justified by the following result.

Proposition 6. *The random variable $\sigma'(Z)$ exhibits sub-Gaussian concentration around its mean*

$$\mathbb{P}(|\sigma'(Z) - \mathbb{E}[\sigma'(Z)]| \geq \delta) \leq 2 \exp\left(-\frac{54\delta^2}{\sigma_h^2}\right) \quad (61)$$

Proof. The second derivative of the sigmoid function $\sigma''(z) = \sigma(z)(1 - \sigma(z))(1 - 2\sigma(z))$ admits a global supremum norm $L = \sup_{z \in \mathbb{R}} |\sigma''(z)| = \frac{1}{6\sqrt{3}}$. Therefore, the function $\sigma' : \mathbb{R} \rightarrow \mathbb{R}$ is uniformly L -Lipschitz continuous. Applying the Gaussian Lipschitz concentration inequality (see Chapter 2.3 and Theorem 2.26 in the book “High-Dimensional Statistics A Non-Asymptotic Viewpoint” by Martin J. Wainwright) for the function σ' evaluated on the Gaussian random variable Z yields the stated exponential tail bound with constant $(2L^2)^{-1} = 54$. □

B.3.2 Verification of Simplicity Bias Through a Toy Dataset

Our theoretical analysis on the simplicity bias condition relies on the simplicity bias condition $\beta^s(2\alpha - 1) > \beta^c$ which conflates feature strength with data imbalance. To demonstrate how minority group learning dynamics change under varying β^s/β^c and α , we conduct experiments on a toy setup using a synthetic dataset that satisfies our theoretical setting and assumptions. We visualize the learning behavior of a linear model trained with gradient descent under the same setting used in our theoretical analysis.

We use a synthetic dataset $\mathcal{D}$ to visualize minority losses while varying β_s/β_c and α . We adopt the setting in our theoretical study: each sample has label $y \in \{-1, +1\}$ and spurious attribute $a \in \{-1, +1\}$ satisfying $\alpha = P(a = y)$. The input is

$$x = \begin{bmatrix} \beta_c y \\ \beta_s a \end{bmatrix},$$

where β_c is the core feature strength and β_s is the spurious feature strength. This creates majority groups with $a = y$ and minority groups with $a \neq y$. We train a linear model $\hat{y} = x^\top w = w_c \beta_c y + w_s \beta_s a$ with full-batch gradient descent while varying β_s/β_c and α . After training, we measure the peak loss among all minority samples. Figure 5 shows a consistent transition in learned behavior depending on whether $\beta_s(2\alpha - 1) > \beta_c$ holds, and it also captures the boundary cases, verifying the simplicity bias condition.

B.3.3 Relaxation to Multiple Spurious Correlations

Here we extend the theoretical analysis of the homogeneous spiked model to accommodate a dataset where each sample is associated with multiple distinct spurious attributes. Let the training dataset $\mathcal{D}$ of size n consist of samples (x_i, y_i) where $y_i \in \{\pm 1\}$ represents the target label. Each sample x_i contains a core feature vector x_i^c and M independent spurious feature vectors $x_i^{s,m}$ for $m \in \{1, \dots, M\}$. We denote the corresponding unobserved binary spurious attributes by the vector $\mathbf{a}_i \in \{\pm 1\}^M$ where the m -th element is $a_{i,m}$. We quantify the consistency of each spurious attribute within the dataset by defining the scalar $\alpha_m \in (0.5, 1]$ as the fraction of samples satisfying $a_{i,m} = y_i$.

Assumption 6. *The neural network architecture permits an additive decomposition of the output logit into a core subnetwork and M distinct spurious subnetworks*

$$h(x; \mathbf{W}) = h_c(x^c; \mathbf{W}^c) + \sum_{m=1}^M h_{s,m}(x^{s,m}; \mathbf{W}^{s,m}) \quad (62)$$

where the parameter sets $\mathbf{W}^c$ and $\mathbf{W}^{s,m}$ for $m \in \{1, \dots, M\}$ are mutually disjoint.

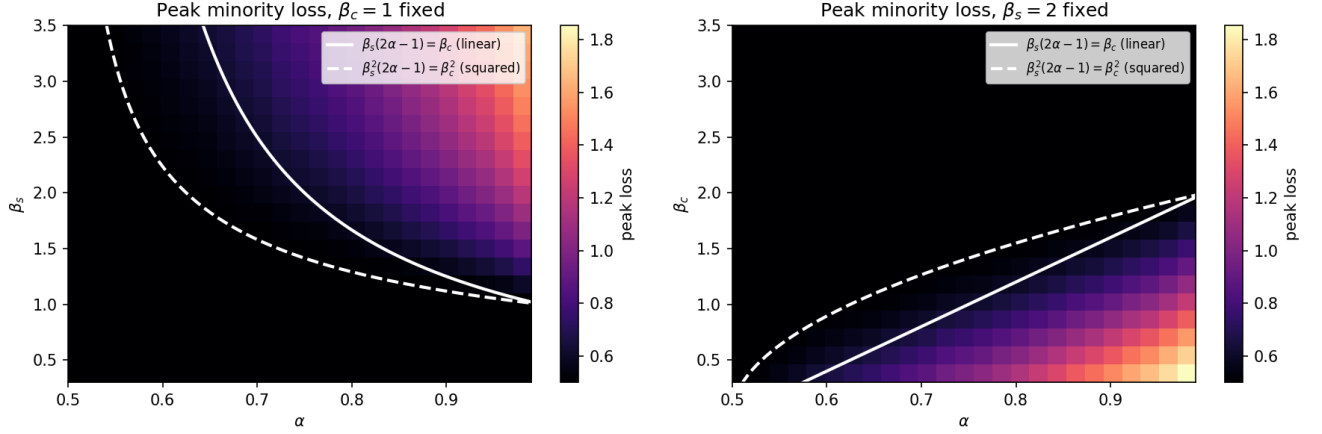


Figure 5: Learning behavior and simplicity bias analysis on a synthetic toy dataset.

By the linearity of the gradient with respect to disjoint parameter subsets, the NTK decomposes additively into $M + 1$ distinct components evaluated on their respective feature subspaces

$$K(x, x') = K_c(x^c, x'^c) + \sum_{m=1}^M K_{s,m}(x^{s,m}, x'^{s,m}) \quad (63)$$

where K_c and $K_{s,m}$ represent the tangent kernels of the core and spurious subnetworks.

Assumption 7. *The kernel components exhibit a uniform rank-1 spiked structure corresponding to the latent data generation process*

$$K_c(x_j^c, x_i^c) = \beta^c y_j y_i \quad (64)$$

and

$$K_{s,m}(x_j^{s,m}, x_i^{s,m}) = \beta^{s,m} a_{j,m} a_{i,m} \quad (65)$$

where the scalars $\beta^c > 0$ and $\beta^{s,m} > 0$ capture the inherent feature strengths of the core and spurious components respectively.

We assume symmetric initialization such that the expected initial output of the network is zero. Under gradient flow with the logistic loss function, the exact initial dynamics of the expected logits are governed by the dataset statistics and the isolated kernel strengths.

Theorem 5. *Let $\bar{h}_t(x_j) = \mathbb{E}_{\mathbf{W}_0}[h(x_j; \mathbf{W}_t)]$ denote the expected logit for sample j at time t . The initial expected velocities of the subnetworks are given by*

$$\left. \frac{\partial \bar{h}_t^c(x_j)}{\partial t} \right|_{t=0} = \frac{\beta^c}{2} y_j \quad (66)$$

and

$$\left. \frac{\partial \bar{h}_t^{s,m}(x_j)}{\partial t} \right|_{t=0} = \frac{\beta^{s,m}(2\alpha_m - 1)}{2} a_{j,m} \quad (67)$$

for all $m \in \{1, \dots, M\}$.

Proof. The time derivative of the expected logit evaluated at $t = 0$ under the specified flow ordinary differential equation is $\dot{\bar{\mathbf{h}}}_0 = \frac{1}{2n} K \mathbf{y}$. We project the vector $\mathbf{y} \in \mathbb{R}^n$ onto the respective decomposed kernel matrices. For the core subnetwork we have

$$\left. \frac{\partial \bar{\mathbf{h}}_t^c}{\partial t} \right|_{t=0} = \frac{1}{2n} (\beta^c \mathbf{y} \mathbf{y}^\top) \mathbf{y} = \frac{\beta^c}{2n} \mathbf{y} (\mathbf{y}^\top \mathbf{y}) = \frac{\beta^c}{2} \mathbf{y} \quad (68)$$

since $\mathbf{y}^\top \mathbf{y} = \sum_{i=1}^n y_i^2 = n$. For the m -th spurious subnetwork we project $\mathbf{y}$ onto the rank-1 spurious matrix parameterized by the vector $\mathbf{a}_m \in \mathbb{R}^n$

$$\left. \frac{\partial \bar{\mathbf{h}}_t^{s,m}}{\partial t} \right|_{t=0} = \frac{1}{2n} (\beta^{s,m} \mathbf{a}_m \mathbf{a}_m^\top) \mathbf{y} = \frac{\beta^{s,m}}{2n} \mathbf{a}_m (\mathbf{a}_m^\top \mathbf{y}) \quad (69)$$

The inner product evaluates to $\mathbf{a}_m^\top \mathbf{y} = \sum_{i=1}^n a_{i,m} y_i$. By the definition of α_m , the number of samples where $a_{i,m} = y_i$ is $\alpha_m n$ and the number where $a_{i,m} = -y_i$ is $(1 - \alpha_m)n$. Hence the summation equals $n(2\alpha_m - 1)$. Substitution yields the exact velocity element-wise. $\square$

We formalize the generalized simplicity bias phenomenon by evaluating the initial derivative of the true expected margin $\bar{m}_t(x_j) = \mathbb{E}_{\mathbf{W}_0}[y_j h(x_j; \mathbf{W}_t)]$. The evolution of the margin demonstrates how intersecting group alignments either accelerate or hinder the correct classification of any sample j .

Theorem 6. *Define the disjoint index sets $\mathcal{M}_{align}(j) = \{m \mid y_j a_{j,m} = 1\}$ and $\mathcal{M}_{conflict}(j) = \{m \mid y_j a_{j,m} = -1\}$. The expected margin for sample j initially decreases and the corresponding cross-entropy loss increases if and only if*

$$\sum_{m \in \mathcal{M}_{conflict}(j)} \beta^{s,m} (2\alpha_m - 1) > \beta^c + \sum_{m \in \mathcal{M}_{align}(j)} \beta^{s,m} (2\alpha_m - 1) \quad (70)$$

Proof. We compute the initial time derivative of the expected margin by multiplying the total initial logit velocity by y_j

$$\left. \frac{\partial \bar{m}_t(x_j)}{\partial t} \right|_{t=0} = y_j \left(\frac{\beta^c}{2} y_j + \sum_{m=1}^M \frac{\beta^{s,m} (2\alpha_m - 1)}{2} a_{j,m} \right) \quad (71)$$

Applying $y_j^2 = 1$ and distributing y_j into the summation yields

$$\left. \frac{\partial \bar{m}_t(x_j)}{\partial t} \right|_{t=0} = \frac{\beta^c}{2} + \sum_{m=1}^M \frac{\beta^{s,m} (2\alpha_m - 1)}{2} y_j a_{j,m} \quad (72)$$

We partition the summation over the sets $\mathcal{M}_{align}(j)$ and $\mathcal{M}_{conflict}(j)$ where $y_j a_{j,m}$ takes values of 1 and -1 respectively

$$\left. \frac{\partial \bar{m}_t(x_j)}{\partial t} \right|_{t=0} = \frac{\beta^c}{2} + \sum_{m \in \mathcal{M}_{align}(j)} \frac{\beta^{s,m} (2\alpha_m - 1)}{2} - \sum_{m \in \mathcal{M}_{conflict}(j)} \frac{\beta^{s,m} (2\alpha_m - 1)}{2} \quad (73)$$

The loss function increases at initialization if and only if the margin derivative is negative. Setting the right-hand side to be less than zero directly yields the stated condition. $\square$

B.4 BACKGROUND ON THE NTK AND SPIKED COVARIANCE MODEL

Below, we provide a brief introduction to NTK and spiked covariance models to make our theoretical analysis more accessible.

B.4.1 Background on the NTK

Definition 2. Let $h(x; \mathbf{W})$ denote the scalar output of a neural network parameterized by the weight vector $\mathbf{W} \in \mathbb{R}^P$. The NTK evaluates the inner product of the gradients of the network output with respect to its parameters evaluated at two inputs x and x'

$$K(x, x'; \mathbf{W}) = \nabla_{\mathbf{W}} h(x; \mathbf{W})^\top \nabla_{\mathbf{W}} h(x'; \mathbf{W}) \quad (74)$$

The key idea of the NTK framework is that as the width of the hidden layers approaches infinity, this empirical kernel converges to a fixed limit.

Theorem 7. *In the infinite-width limit, under symmetric initialization of the parameters $\mathbf{W}_0$, the empirical kernel $K(x, x'; \mathbf{W}_t)$ converges in probability to a deterministic, positive semi-definite kernel $K(x, x')$. Furthermore, this kernel remains time-invariant throughout the optimization process.*

This time-invariance property implies that the highly non-linear parameter space optimization maps to a linear functional space optimization. To formalize the learning dynamics, one typically approximates discrete stochastic gradient descent via continuous-time gradient flow, as done in our paper as well.

Definition 3. Let $\mathcal{L}(\mathbf{W}_t) = \frac{1}{n} \sum_{i=1}^n \ell(y_i, h(x_i; \mathbf{W}_t))$ define the empirical risk over a dataset of size n , where $\ell(\cdot, \cdot)$ is a differentiable loss function. Continuous-time gradient flow dictates the parameter evolution according to the ordinary differential equation

$$\frac{d\mathbf{W}_t}{dt} = -\nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}_t) \quad (75)$$

By applying the chain rule, we map the evolution of the network parameters to the evolution of the network outputs (logits) for any given input x .

Theorem 8. Under gradient flow in the NTK regime, the exact evolution of the expected network output $\bar{h}_t(x) = \mathbb{E}_{\mathbf{W}_0}[h(x; \mathbf{W}_t)]$ is governed by the deterministic equation

$$\frac{\partial \bar{h}_t(x)}{\partial t} = -\frac{1}{n} \sum_{i=1}^n K(x, x_i) \mathbb{E}_{\mathbf{W}_0} \left[\frac{\partial \ell(y_i, h(x_i; \mathbf{W}_t))}{\partial h(x_i; \mathbf{W}_t)} \right] \quad (76)$$

This differential equation constitutes the core machinery of our theoretical analysis. It demonstrates that the instantaneous change in the model’s prediction for a sample x is a linear combination of the gradients of the loss evaluated on all training samples x_i , weighted exactly by the similarity measure defined by the kernel $K(x, x_i)$.

To intuitively understand why this framework is necessary for our paper, consider the separation of features. If an architecture processes a core feature x^c and a spurious feature x^s through disjoint parameter subsets, the definition of the NTK ensures that the global kernel additively decomposes into a core kernel and a spurious kernel.

Consequently, the differential equation governing the learning dynamics linearly separates into independent velocity components driven by these respective kernels. By substituting the spiked covariance model (discussed in the next section) into this ODE, we can extract the learning velocities of different features. Features that frequently co-occur with the target label yield large cumulative sums in the differential equation, forcing the network to minimize the loss along those feature dimensions at a faster rate. This formulation thus helps us avoid heuristic explanations of simplicity bias and instead, to our knowledge for the first time, concretely quantify the conditions under which a neural network prioritizes spurious correlations.

B.4.2 Background on the Spiked Covariance Model

Characterizing the exact optimization dynamics of overparameterized neural networks requires analyzing the spectrum of the data covariance or the induced Gram matrix. When input data lacks specific latent structures, the sample covariance spectrum is bounded and continuously distributed, rendering the isolation of individual feature learning velocities impossible. Accordingly, we propose relying on the widely adopted notion of spiked models.

Definition 4. The spiked covariance model posits that the population covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$ decomposes into an isotropic background noise component perturbed by a low-rank structural matrix of rank $K \ll d$

$$\Sigma = \sigma^2 I_d + \sum_{k=1}^K \lambda_k v_k v_k^\top \quad (77)$$

where $\sigma^2 > 0$ denotes the uniform noise variance, $\lambda_k > 0$ are the discrete spike eigenvalues representing signal strengths, and $v_k \in \mathbb{R}^d$ are orthonormal vectors defining the principal latent directions of the data distribution.

In the context of machine learning, these principal directions encode the dominant predictive features embedded within the input space. When evaluating the NTK across the training dataset, the inner products of the network gradients are overwhelmingly governed by these underlying latent factors.

In our paper, we map this classical statistical model directly to the kernel matrix induced by the NTK. We assume the dataset generation is governed by dominant latent variables corresponding to the core and spurious features. We study the idealized low-rank structure of the kernel matrix $K \in \mathbb{R}^{n \times n}$ by isolating the spectral spikes

$$K = \beta^c \mathbf{y} \mathbf{y}^\top + \beta^s \mathbf{a} \mathbf{a}^\top \quad (78)$$

The orthonormal vectors v_k from the classical statistical formulation are replaced by the dataset-level structural vectors $\mathbf{y}$ and $\mathbf{a}$, while the spike eigenvalues λ_k map to the isolated feature strengths β^c and β^s .

This framework thus helps bridge random matrix theory and deep learning optimization, which we found essential for our ensuing theoretical analysis: It guarantees that the gradient flow dynamics are constrained to a low-dimensional subspace spanned precisely by the core and spurious features. Without adopting the spiked covariance formulation, the continuous-time ordinary differential equations governing the expected logits would entangle across all n data dimensions. By restricting the kernel to have discrete structural spikes, we decouple the dataset complexity and derive exact, closed-form velocities for the competing feature components.

C MEMORIZATION AND CSL DEFINITIONS

C.1 MEMORIZATION SCORE

The memorization score introduced by Feldman [2020] is quantitatively defined and calculated for each training sample as follows:

$$\text{mem}(f, \mathcal{D}, i) = \mathbb{P}_{\hat{f}=f(\mathcal{D})}[\arg \max(\hat{f}(x_i)) = y_i] - \mathbb{P}_{\hat{f}=f(\mathcal{D} \setminus i)}[\arg \max(\hat{f}(x_i)) = y_i], \quad (79)$$

Here, f denotes the learning algorithm trained on the dataset, and $\hat{f}$ represents the trained model. $\mathcal{D} \setminus i$ denotes the subportion of the dataset $\mathcal{D}$ with the sample i removed. While memorization score has proven effective for various tasks including coreset selection [Sorscher et al., 2022], its computation is prohibitively expensive, requiring the model to be retrained from scratch after removing each individual sample. Consequently, researchers have proposed several computationally efficient proxies.

C.2 CUMULATIVE SAMPLE LOSS (CSL)

The Cumulative Sample Loss (CSL) was recently proposed by Ravikumar et al. [2025] as an efficient proxy for the memorization score and is computed as

$$\text{CSL}(x_i) = \frac{1}{T} \sum_{t=1}^T \ell(y_i, f(x_i, \mathbf{W}_t)), \quad (80)$$

where the cross-entropy loss $\ell(\cdot)$ is calculated for each training sample x_i . Sample losses are recorded at the end of every training epoch and the final CSL is obtained by averaging the losses across all training epochs.

D FORMAL DEFINITIONS OF HELPER ALGORITHMS

Helper algorithms used in the main paper are given in **Algorithms 3, 4 and 5**.

E DATASET DETAILS

We present detailed explanations of the datasets used in our experiments. Further information regarding the groups is illustrated in Table 4.

Waterbirds [Sagawa et al., 2020b] The labels are landbird and waterbird, where the spurious attribute corresponds to the background type: land or water. The dataset is synthetically constructed by placing bird images from the Caltech-UCSD Birds-200-2011 dataset [Wah et al., 2011] onto background images from the Places dataset [Zhou et al., 2016].

cMNIST [Arjovsky et al., 2019] A synthetic variant of the MNIST dataset consisting of 10 digit classes, where each digit is assigned a distinct color that serves as the spurious attribute.

Algorithm 3 WKMEANS

Input: weights $\mathbf{w} = \{w_i\}_{i=1}^n$, sample representations $\mathbf{L} = \{\mathbf{L}_i\}_{i=1}^n$, number of clusters $K=2$.

Output: Cluster assignment sets $\{G_k\}_{k=1}^K$.

- 1: **Initialization:** Initialize centers $\{\mu_k\}_{k=1}^K$ randomly from $\mathbf{L}$. Set maximum iterations to 200.
- 2: **while** cluster assignments $\{c_i\}$ change and the maximum iterations has not reached **do**
- 3: **Assignment step:**
- 4: **for** $i = 1$ **to** n **do**
- 5: Assign x_i to the closest center

$$c_i \leftarrow \arg \min_{k \in \{1, \dots, K\}} \|\mathbf{L}_i - \mu_k\|_2^2$$

- 6: **end for**
- 7: Define clusters $G_k \leftarrow \{i : c_i = k\}$ for $k = 1, \dots, K$.
- 8: **Update step:**
- 9: **for** $k = 1$ **to** K **do**
- 10: Update center using weighted mean:

$$\mu_k \leftarrow \frac{\sum_{i \in G_k} w_i \mathbf{L}_i}{\sum_{i \in G_k} w_i}.$$

- 11: **end for**
 - 12: **end while**
 - 13: **Return** clusters $\{G_k\}_{k=1}^K$.
-

Algorithm 4 SELECTBOT

Input: Sorted group $\tilde{G}$ by TCSL_c (ascending), number of samples to select $\tilde{n}$.

Output: Selected set $\tilde{\mathcal{D}}$.

- 1: **Return** $\tilde{\mathcal{D}} \leftarrow \tilde{G}[1:\tilde{n}]$
-

MetaShift [Liang and Zou, 2022] This dataset contains cat and dog images, where the spurious attribute corresponds to the environment of the animal, given as indoor or outdoor.

UrbanCars-B [Li et al., 2023] The goal is to classify samples as urban or country cars, where the background serves as the spurious attribute. The dataset is synthetically constructed by placing car images onto background images from the Places dataset [Zhou et al., 2016].

F BASELINE METHODS AND SAMPLE SCORES

In this section, we describe the baseline debiasing methods used in our experiments, as well as the sample scoring functions employed in the coresnet selection experiments.

CB ERM and GB ERM denote the standard ERM model trained with samples reweighted by the inverse of their class sizes and group sizes, respectively. GroupDRO [Sagawa et al., 2020a] uses group labels during training to upweight samples from the worst performing group and directly minimizes the worst group loss. LC [Liu et al., 2023] is a two-stage training algorithm that uses information from a first model as prior knowledge about the groups to shift the second model towards a more group-balanced solution. DFR [Kirichenko et al., 2023] retrains the last layer of a standard ERM model, initially trained on the entire dataset, using a group-balanced validation set. CNC [Zhang et al., 2022] is a two-stage training algorithm in which the second model is trained with a contrastive loss that aligns representations of samples within a class while mitigating spurious correlations. LfF [Nam et al., 2020] is a two-stage training algorithm where the debiased model is trained to upweight samples on which the biased model fails to predict accurately. JTT [Liu et al., 2021] first trains an ERM model and marks the samples it misclassifies, then retrains another model from scratch while upweighting these misclassified samples. ULA [Tsirigotis et al., 2023] is a two-stage training algorithm where the biased model is a self-supervised pretrained network and the debiased model is trained with a logit adjustment similar to LC. EIIL [Creager

Algorithm 5 SELECTHIST

Input: Sorted group $\tilde{G}$ by TCSL_c, number of samples to select $\tilde{n}$, number of bins B .

Output: Selected set $\tilde{D}$.

```
1: Partition  $\tilde{G}$  into  $B$  equal-size bins  $\{\tilde{G}_1, \dots, \tilde{G}_B\}$ 
2:  $\tilde{D} \leftarrow \emptyset$ 
3: while  $|\tilde{D}| < \tilde{n}$  do
4:   for each bin  $\tilde{G}_b$  do
5:     if  $|\tilde{D}| = \tilde{n}$  then
6:       break
7:     end if
8:     if  $\tilde{G}_b \neq \emptyset$  then
9:       pick a random  $\tilde{x} \in \tilde{G}_b$ 
10:       $\tilde{D} \leftarrow \tilde{D} \cup \{\tilde{x}\}$ 
11:       $\tilde{G}_b \leftarrow \tilde{G}_b \setminus \{\tilde{x}\}$ 
12:    end if
13:  end for
14: end while
15: Return  $\tilde{D}$ 
```

Dataset	y	$a = y$	$a = -y$	α
Waterbirds [Sagawa et al., 2020b]	2	4555	240	0.950
cMNIST [Arjovsky et al., 2019]	10	52551	257	0.995
MetaShift [Liang and Zou, 2022]	2	1500	300	0.882
UrbanCars-B [Li et al., 2023]	2	7600	400	0.950

Table 4: Dataset details including the number of classes, number of samples where the spurious attribute agrees with the label, number of samples where the spurious attribute does not agree with the label and the α parameter.

et al., 2021] learns an invariant model based on groups (environments) identified by a reference model. GEORGE [Sohoni et al., 2020] clusters the feature space of a standard ERM model within each class to identify groups and then uses the inferred groups to train a new model with an objective similar to GroupDRO.

For our coreset selection experiments, following the setting in [Dharmasiri et al., 2025], we employ four baseline scoring functions to compute sample scores. EL2N [Paul et al., 2021] assigns a difficulty score to each sample based on the norm of the difference between the model’s predicted probability vector and the one-hot label vector. SelfSup [Sorscher et al., 2022] uses an embedding-based scoring method, defined as the norm of the difference between the feature vector of a sample and the mean feature vector of its assigned cluster. Random [Dharmasiri et al., 2025] randomly selects samples according to the coreset selection ratio r . RGBal [Dharmasiri et al., 2025] uses group labels to always select samples from the minority group and randomly selects from the majority group to satisfy the remaining quota. We additionally include the state-of-the-art coreset selection algorithm D2 Maharana et al. [2024] in our experiments. D2 combines a chosen sample scoring function with feature embeddings to capture feature similarity and employs a message-passing scheme to select coresets with high distributional coverage. As the sample scoring function of D2, we use the EL2N and SelfSup scores. For feature embeddings, we consider (i) representations extracted from a ResNet model trained with ERM on the corresponding training dataset and (ii) representations from the pretrained CLIP model Radford et al. [2021]. This results in four variants of the D2 algorithm.

G HYPERPARAMETERS FOR MODEL TRAINING AND CORESET SELECTION

G.1 MODEL TRAINING

To maintain consistency with prior work, we use SGD as our optimization algorithm. Following the literature, we adopt the ResNet50 architecture for Waterbirds, MetaShift and UrbanCars-B, and ResNet18 for cMNIST. All architectures are initialized with ImageNet-1K [Russakovsky et al., 2015] pretrained weights. The hyperparameters used for the ERM model in our experiments are listed in Table 5. Both the model architectures and hyperparameters are chosen to match the most

Parameter	Waterbirds	cMNIST	MetaShift	UrbanCars-B
Learning rate	1e-4	1e-3	1e-3	1e-4
Weight decay	1e-1	1e-3	1e-3	1e-1
Momentum	0.9	0.9	0.9	0.9
Batch size	128	32	32	128
Training epochs	300	50	200	300

Table 5: Hyperparameters for the datasets used in our experiments.

commonly employed configurations in baseline studies; therefore, we do not perform hyperparameter tuning or early stopping. For both f_s and f_c , we use the same hyperparameters as the ERM model, changing only the total training epochs to $T_s = T/10$ for the spurious network. Our goal is to demonstrate that the performance gains achieved by retraining the ERM model on the coreset selected by TCSL-CS arise solely from the effectiveness of our proposed coreset selection method, rather than from any hyperparameter adjustments.

G.2 CORESET SELECTION

Our coreset selection algorithm TCSL-CS has two hyperparameters: the number of bins B used for histogram-based selection and the threshold τ , which determines when to switch from bottom-based selection (SELECTBOT) to histogram-based selection (SELECTHIST). Staying consistent with the coreset selection literature, we treat the coreset selection ratio r as a user-specified input to the algorithm. Following prior histogram-based coreset selection methods [Zheng et al., 2023], we set $B = 50$ by default and do not tune it. For all datasets except cMNIST, we set $\tau = 0.4$. For cMNIST, due to the lower complexity of digit images, the TCSL_c scores do not form a sufficiently diverse distribution. Therefore, we disable SELECTHIST by setting $\tau = 0$.

For retraining the CB ERM model on the selected coresets, we fix the total number of training iterations by setting the number of training epochs to T/r , where r is the coreset selection ratio and T denotes the total training epochs of the CB ERM model trained on the full dataset. We compute the EL2N [Paul et al., 2021] scores using the CB ERM model after $T/10$ epochs of training on the full dataset.

H ADDITIONAL EXPERIMENTS

We provide additional coreset selection results for different selection ratios. We demonstrate results for ratios 0.2, 0.4, 0.6 and 0.8 in Tables 6, 7, 8 and 9, respectively.

Method	Waterbirds WGA (AVG)	cMNIST WGA (AVG)	MetaShift WGA (AVG)	UrbanCars-B WGA (AVG)
EL2N (Bot)	44.89±2.14 (93.98±0.42)	0.00±0.00 (13.92±0.67)	52.78±5.93 (77.95±1.71)	34.00±5.23 (66.73±4.40)
EL2N (Top)	<u>89.56</u> ±0.93 (98.05±0.16)	49.66±7.07 (75.06±2.09)	77.55±2.23 (87.50±0.60)	62.53±0.46 (74.87±1.34)
EL2N (Hist)	74.28±2.13 (96.57±0.07)	10.40±5.71 (35.18±1.39)	68.98±2.63 (84.14±0.86)	57.07±3.35 (78.30±0.95)
SelfSup (Bot)	61.21±0.26 (90.98±0.15)	0.00±0.00 (19.16±2.00)	47.92±2.08 (81.02±0.70)	43.47±2.95 (77.50±0.80)
SelfSup (Top)	83.85±1.80 (97.80±0.29)	52.70±8.00 (76.05±1.28)	48.38±6.42 (78.88±1.71)	28.13±3.63 (68.63±1.10)
SelfSup (Hist)	76.88±0.34 (96.84±0.39)	5.40±2.81 (34.16±0.91)	67.59±2.44 (83.97±1.16)	60.00±5.64 (80.17±1.69)
D2 (EL2N+ResNet)	<u>89.56</u> ±0.41 (98.26±0.01)	20.79±11.92 (42.64±8.86)	<u>79.17</u> ±0.69 (87.91±0.70)	70.00±2.23 (84.37±1.03)
D2 (SelfSup+ResNet)	83.84±0.40 (96.97±0.13)	25.57±7.09 (58.65±5.23)	48.84±6.85 (78.99±1.84)	41.33±1.97 (75.10±0.66)
D2 (EL2N+CLIP)	86.09±1.00 (98.22±0.03)	27.87±11.64 (56.75±4.07)	73.84±1.06 (86.86±0.61)	62.80±3.49 (80.20±1.35)
D2 (SelfSup+CLIP)	80.32±0.74 (97.72±0.03)	22.05±9.23 (50.58±11.16)	68.29±2.81 (83.97±0.70)	37.47±4.28 (72.07±0.23)
Random	73.44±2.31 (97.15±0.08)	5.40±0.72 (34.60±1.39)	67.36±1.39 (83.97±2.24)	56.00±0.69 (78.97±0.86)
RGlobal	87.49±1.82 (96.98±0.15)	<u>73.83</u> ±3.65 (85.58±1.10)	80.79 ±1.06 (87.85±1.22)	<u>81.73</u> ±1.10 (87.60±0.10)
TCSL-CS	90.60 ±0.48 (96.68±0.08)	85.26 ±1.72 (92.22±0.59)	76.85±4.24 (88.60±1.12)	83.60 ±0.40 (88.57±0.38)

Table 6: We compare the WGA and AVG of different coreset selection methods at $r = 0.2$ across datasets. Results are averaged over 3 seeds. We use CB ERM for retraining on all of the identified coresets. The best WGA for each dataset is shown in **bold** and the second best value is underlined. Only RGlobal has access to group labels.

Method	Waterbirds WGA (AVG)	cMNIST WGA (AVG)	MetaShift WGA (AVG)	UrbanCars-B WGA (AVG)
EL2N (Bot)	35.57 $\pm$ 4.43 (92.71 $\pm$ 0.45)	0.00 $\pm$ 0.00 (14.21 $\pm$ 0.04)	60.42 $\pm$ 7.05 (80.21 $\pm$ 2.13)	29.87 $\pm$ 6.02 (65.63 $\pm$ 4.49)
EL2N (Top)	85.25 $\pm$ 0.39 (98.39 $\pm$ 0.04)	52.18 $\pm$ 14.40 (75.43 $\pm$ 4.29)	75.46 $\pm$ 2.12 (88.54 $\pm$ 0.52)	73.33 $\pm$ 3.21 (85.67 $\pm$ 1.10)
EL2N (Hist)	78.09 $\pm$ 0.01 (97.46 $\pm$ 0.04)	25.76 $\pm$ 8.73 (51.65 $\pm$ 3.97)	71.76 $\pm$ 1.75 (85.19 $\pm$ 0.40)	62.40 $\pm$ 2.80 (80.87 $\pm$ 0.21)
SelfSup (Bot)	50.95 $\pm$ 1.32 (89.60 $\pm$ 0.03)	0.00 $\pm$ 0.00 (17.58 $\pm$ 0.08)	61.34 $\pm$ 4.62 (84.84 $\pm$ 1.61)	50.40 $\pm$ 2.40 (80.00 $\pm$ 0.44)
SelfSup (Top)	83.64 $\pm$ 0.16 (98.22 $\pm$ 0.14)	60.12 $\pm$ 4.10 (76.64 $\pm$ 0.70)	71.30 $\pm$ 3.43 (85.76 $\pm$ 0.90)	42.13 $\pm$ 4.67 (74.27 $\pm$ 0.55)
SelfSup (Hist)	78.20 $\pm$ 2.54 (97.50 $\pm$ 0.07)	24.51 $\pm$ 3.26 (50.54 $\pm$ 8.80)	71.06 $\pm$ 2.12 (85.94 $\pm$ 0.46)	61.47 $\pm$ 4.84 (80.63 $\pm$ 0.91)
D2 (EL2N+ResNet)	84.79 $\pm$ 0.63 (98.35 $\pm$ 0.05)	40.22 $\pm$ 4.86 (67.71 $\pm$ 2.44)	75.46 $\pm$ 2.12 (88.25 $\pm$ 0.20)	70.93 $\pm$ 2.54 (83.50 $\pm$ 0.89)
D2 (SelfSup+ResNet)	83.43 $\pm$ 0.51 (97.61 $\pm$ 0.20)	64.07 $\pm$ 7.23 (79.94 $\pm$ 0.94)	70.60 $\pm$ 3.56 (85.36 $\pm$ 0.96)	60.53 $\pm$ 1.15 (80.23 $\pm$ 0.15)
D2 (EL2N+CLIP)	76.17 $\pm$ 0.71 (97.88 $\pm$ 0.05)	33.30 $\pm$ 5.13 (64.89 $\pm$ 2.97)	70.60 $\pm$ 2.00 (85.88 $\pm$ 0.82)	63.73 $\pm$ 0.61 (81.53 $\pm$ 0.55)
D2 (SelfSup+CLIP)	75.03 $\pm$ 1.92 (97.82 $\pm$ 0.08)	36.48 $\pm$ 4.46 (67.23 $\pm$ 1.68)	71.76 $\pm$ 0.80 (86.05 $\pm$ 0.44)	61.47 $\pm$ 0.92 (79.67 $\pm$ 0.93)
Random	75.47 $\pm$ 0.77 (97.38 $\pm$ 0.11)	28.71 $\pm$ 17.27 (61.84 $\pm$ 1.12)	71.30 $\pm$ 1.75 (85.94 $\pm$ 0.63)	61.20 $\pm$ 2.12 (80.57 $\pm$ 0.46)
RGBal	<u>86.55</u> $\pm$ 2.52 (97.97 $\pm$ 0.01)	<u>66.02</u> $\pm$ 5.78 (81.58 $\pm$ 0.28)	<u>78.70</u> $\pm$ 1.06 (89.35 $\pm$ 0.82)	<u>76.27</u> $\pm$ 0.83 (86.57 $\pm$ 0.25)
TCSL-CS	88.63 $\pm$ 0.61 (92.55 $\pm$ 0.11)	75.75 $\pm$ 8.18 (88.38 $\pm$ 0.67)	82.64 $\pm$ 0.69 (87.27 $\pm$ 0.72)	81.87 $\pm$ 1.40 (84.17 $\pm$ 0.25)

Table 7: We compare the WGA and AVG of different coreset selection methods at $r = 0.4$ across datasets. Results are averaged over 3 seeds. We use CB ERM for retraining on all of the identified coresets. The best WGA for each dataset is shown in **bold** and the second best value is underlined. Only RGBal has access to group labels.

Method	Waterbirds WGA (AVG)	cMNIST WGA (AVG)	MetaShift WGA (AVG)	UrbanCars-B WGA (AVG)
EL2N (Bot)	30.92 $\pm$ 0.62 (93.17 $\pm$ 0.39)	0.00 $\pm$ 0.00 (16.50 $\pm$ 1.12)	60.65 $\pm$ 5.36 (80.21 $\pm$ 2.23)	32.93 $\pm$ 8.34 (67.37 $\pm$ 4.10)
EL2N (Top)	76.40 $\pm$ 8.41 (98.25 $\pm$ 0.11)	51.31 $\pm$ 3.94 (77.48 $\pm$ 0.50)	74.07 $\pm$ 0.65 (87.79 $\pm$ 0.22)	68.67 $\pm$ 2.44 (83.80 $\pm$ 0.20)
EL2N (Hist)	81.31 $\pm$ 1.22 (97.81 $\pm$ 0.04)	36.42 $\pm$ 0.56 (62.40 $\pm$ 0.26)	72.22 $\pm$ 1.70 (86.11 $\pm$ 0.57)	65.73 $\pm$ 3.26 (82.57 $\pm$ 0.45)
SelfSup (Bot)	49.25 $\pm$ 1.49 (93.99 $\pm$ 0.31)	0.00 $\pm$ 0.00 (15.36 $\pm$ 1.20)	67.59 $\pm$ 0.65 (86.11 $\pm$ 0.25)	58.00 $\pm$ 0.69 (82.50 $\pm$ 0.61)
SelfSup (Top)	80.74 $\pm$ 0.80 (98.16 $\pm$ 0.03)	<u>55.62</u> $\pm$ 19.36 (78.88 $\pm$ 2.55)	<u>75.46</u> $\pm$ 1.18 (86.75 $\pm$ 0.43)	53.60 $\pm$ 8.65 (79.43 $\pm$ 1.76)
SelfSup (Hist)	81.09 $\pm$ 0.17 (97.77 $\pm$ 0.02)	39.27 $\pm$ 8.77 (62.18 $\pm$ 0.11)	71.99 $\pm$ 1.73 (86.52 $\pm$ 0.46)	66.00 $\pm$ 4.92 (82.00 $\pm$ 0.66)
D2 (EL2N+ResNet)	81.00 $\pm$ 1.18 (98.27 $\pm$ 0.07)	41.21 $\pm$ 4.88 (68.75 $\pm$ 7.38)	73.61 $\pm$ 0.00 (87.73 $\pm$ 0.27)	66.53 $\pm$ 3.00 (82.93 $\pm$ 1.68)
D2 (SelfSup+ResNet)	82.50 $\pm$ 1.26 (97.99 $\pm$ 0.06)	58.32 $\pm$ 1.96 (77.02 $\pm$ 1.72)	<u>75.46</u> $\pm$ 1.45 (86.63 $\pm$ 0.52)	61.20 $\pm$ 3.67 (80.73 $\pm$ 0.45)
D2 (EL2N+CLIP)	76.32 $\pm$ 2.04 (97.66 $\pm$ 0.04)	30.45 $\pm$ 7.00 (65.55 $\pm$ 4.61)	72.22 $\pm$ 0.69 (85.76 $\pm$ 0.17)	66.27 $\pm$ 0.23 (81.67 $\pm$ 0.21)
D2 (SelfSup+CLIP)	74.04 $\pm$ 0.94 (97.70 $\pm$ 0.03)	37.58 $\pm$ 5.15 (66.85 $\pm$ 3.38)	72.69 $\pm$ 0.80 (86.28 $\pm$ 0.17)	59.47 $\pm$ 0.61 (79.03 $\pm$ 1.29)
Random	78.04 $\pm$ 1.10 (97.69 $\pm$ 0.05)	48.23 $\pm$ 7.14 (68.54 $\pm$ 2.20)	71.53 $\pm$ 0.57 (86.05 $\pm$ 0.46)	64.40 $\pm$ 0.69 (82.00 $\pm$ 0.44)
RGBal	83.54 $\pm$ 1.48 (97.91 $\pm$ 0.11)	54.13 $\pm$ 1.87 (77.46 $\pm$ 1.03)	75.23 $\pm$ 0.87 (88.31 $\pm$ 0.22)	<u>72.80</u> $\pm$ 1.06 (85.03 $\pm$ 0.84)
TCSL-CS	<u>83.52</u> $\pm$ 0.33 (95.08 $\pm$ 0.03)	70.80 $\pm$ 1.61 (86.14 $\pm$ 1.16)	79.40 $\pm$ 1.64 (89.29 $\pm$ 0.30)	80.40 $\pm$ 1.06 (88.03 $\pm$ 0.84)

Table 8: We compare the WGA and AVG of different coreset selection methods at $r = 0.6$ across datasets. Results are averaged over 3 seeds. We use CB ERM for retraining on all of the identified coresets. The best WGA for each dataset is shown in **bold** and the second best value is underlined. Only RGBal has access to group labels.

Method	Waterbirds WGA (AVG)	cMNIST WGA (AVG)	MetaShift WGA (AVG)	UrbanCars-B WGA (AVG)
EL2N (Bot)	28.56 $\pm$ 2.29 (94.81 $\pm$ 0.13)	0.00 $\pm$ 0.00 (18.00 $\pm$ 0.79)	65.97 $\pm$ 3.45 (82.06 $\pm$ 1.66)	32.67 $\pm$ 7.49 (68.83 $\pm$ 4.76)
EL2N (Top)	81.07 $\pm$ 2.09 (98.07 $\pm$ 0.03)	<u>55.48</u> $\pm$ 0.10 (74.35 $\pm$ 0.76)	72.92 $\pm$ 0.57 (87.27 $\pm$ 0.30)	68.80 $\pm$ 0.40 (83.70 $\pm$ 0.36)
EL2N (Hist)	80.49 $\pm$ 1.97 (97.86 $\pm$ 0.02)	40.11 $\pm$ 16.23 (68.80 $\pm$ 2.47)	73.15 $\pm$ 1.31 (86.86 $\pm$ 0.22)	66.93 $\pm$ 1.85 (82.60 $\pm$ 0.75)
SelfSup (Bot)	61.63 $\pm$ 2.65 (96.13 $\pm$ 0.18)	0.00 $\pm$ 0.00 (16.22 $\pm$ 1.95)	70.37 $\pm$ 0.33 (86.69 $\pm$ 0.30)	64.13 $\pm$ 0.83 (83.70 $\pm$ 0.17)
SelfSup (Top)	80.89 $\pm$ 1.06 (98.09 $\pm$ 0.01)	55.35 $\pm$ 4.53 (74.87 $\pm$ 0.74)	74.31 $\pm$ 0.57 (87.27 $\pm$ 0.30)	64.00 $\pm$ 1.60 (81.57 $\pm$ 0.59)
SelfSup (Hist)	<u>82.24</u> $\pm$ 1.10 (97.93 $\pm$ 0.17)	49.94 $\pm$ 10.04 (71.10 $\pm$ 1.54)	71.99 $\pm$ 1.31 (86.75 $\pm$ 0.78)	67.47 $\pm$ 1.80 (82.83 $\pm$ 0.40)
D2 (EL2N+ResNet)	81.26 $\pm$ 1.94 (98.09 $\pm$ 0.04)	45.95 $\pm$ 2.50 (68.54 $\pm$ 0.41)	72.92 $\pm$ 0.69 (87.21 $\pm$ 0.27)	67.60 $\pm$ 3.82 (82.77 $\pm$ 1.11)
D2 (SelfSup+ResNet)	79.75 $\pm$ 2.29 (97.93 $\pm$ 0.03)	47.69 $\pm$ 9.67 (75.62 $\pm$ 2.21)	74.77 $\pm$ 1.06 (87.38 $\pm$ 0.36)	57.60 $\pm$ 2.08 (80.03 $\pm$ 0.21)
D2 (EL2N+CLIP)	77.05 $\pm$ 1.65 (97.75 $\pm$ 0.03)	37.67 $\pm$ 7.74 (63.48 $\pm$ 4.71)	68.98 $\pm$ 2.12 (85.07 $\pm$ 0.80)	64.80 $\pm$ 1.06 (80.90 $\pm$ 0.61)
D2 (SelfSup+CLIP)	73.62 $\pm$ 0.18 (97.76 $\pm$ 0.12)	34.48 $\pm$ 6.20 (64.26 $\pm$ 4.38)	70.60 $\pm$ 1.06 (85.36 $\pm$ 0.44)	59.33 $\pm$ 2.72 (79.43 $\pm$ 1.02)
Random	79.44 $\pm$ 0.66 (97.83 $\pm$ 0.02)	36.02 $\pm$ 1.45 (69.29 $\pm$ 4.13)	72.45 $\pm$ 0.65 (86.92 $\pm$ 0.22)	61.87 $\pm$ 3.00 (82.10 $\pm$ 0.82)
RGBal	81.52 $\pm$ 1.06 (97.88 $\pm$ 0.11)	48.29 $\pm$ 16.12 (76.51 $\pm$ 0.12)	72.45 $\pm$ 1.43 (87.62 $\pm$ 0.08)	<u>70.27</u> $\pm$ 0.46 (84.20 $\pm$ 0.70)
TCSL-CS	83.71 $\pm$ 1.34 (96.87 $\pm$ 0.03)	62.09 $\pm$ 1.71 (82.20 $\pm$ 2.72)	<u>74.54</u> $\pm$ 0.33 (89.00 $\pm$ 0.16)	73.87 $\pm$ 4.69 (86.77 $\pm$ 0.90)

Table 9: We compare the WGA and AVG of different coreset selection methods at $r = 0.8$ across datasets. Results are averaged over 3 seeds. We use CB ERM for retraining on all of the identified coresets. The best WGA for each dataset is shown in **bold** and the second best value is underlined. Only RGBal has access to group labels.

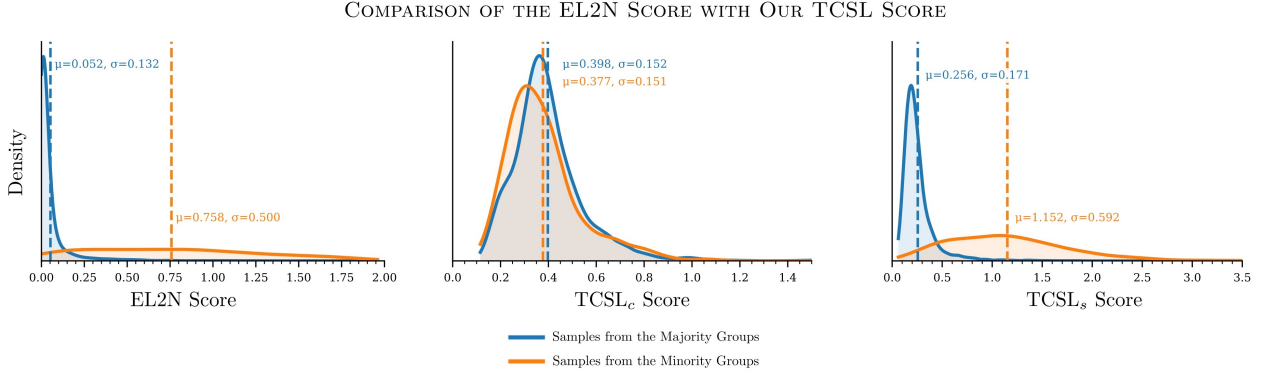


Figure 6: Density Comparison of EL2N and TCSL Scores.

I ADDITIONAL ABLATION STUDIES

In Figure 6, we compare the distribution of EL2N scores with our TCSL_s and TCSL_c scores on Waterbirds. While both EL2N and TCSL_s consistently assign higher scores to minority-group samples (i.e., without spurious background correlations), TCSL_c remains largely invariant to background and assigns comparable scores across groups. Next, we provide additional ablation studies for TCSL-CS. As the TCSL score consists of two components, TCSL_c and TCSL_s, we evaluate the need for each part separately. Finally, we present ablation studies on the choice of B and T_s in our algorithm, to illustrate the robustness of our proposed coreset selection method.

First, we assume that the group identification stage based on the TCSL_s score in **Algorithm 2** of the main paper is replaced with the exact group labels. Hence, our coreset selection algorithm first selects samples from the minority groups and then fills the remaining quota based on the TCSL_c scores of the majority group. In this way, we are able to demonstrate the performance gains achieved by selecting based on the TCSL_c score rather than using random selection as in RGBal. We illustrate the results in Table 10. Since $\alpha > 0.9$ for all datasets except MetaShift, we set $r = 0.1$ as in the main paper. For MetaShift, as $\alpha = 0.882$, we set $r = 0.2$. The results show that TCSL-CS improves performance on all datasets when the group labels are made available. We note that the performance of TCSL-CS could be further improved with group labels, since our TCSL_c score is still computed under a debiased training scheme constructed without access to group labels. With access to group labels, one could build even stronger biased and debiased model training schemes, which would in turn improve the computation of the TCSL_c score.

Next, we compare the TCSL_s scores obtained from our proposed biased model structure given in **Algorithm 1** of the main paper, with the scores obtained from a standard baseline biased model from the literature. Specifically, we train a biased model using the generalized cross entropy (GCE) loss [Zhang and Sabuncu, 2018] for the same number of epochs as our biased model and compute the TCSL_s scores for both cases.

As our goal is to capture information related only to the spurious feature part of the image, we compute the cosine similarity between the computed TCSL_s scores and the CSL scores obtained by manually removing and training on only the spurious or core feature parts of the images. We choose Waterbirds for this study, since the dataset is artificially constructed, allowing for clean separation of the feature parts. The results are presented in Table 11. Since GCE requires tuning the hyperparameter q , we report results for $q \in \{0.7, 0.8, 0.9\}$. As shown, the scores obtained with our biased model achieve higher similarity to $\text{CSL}(x^s)$ and lower similarity to $\text{CSL}(x^c)$. This demonstrates that our biased model architecture is a more suitable choice for our setting than the widely used GCE-based biased training.

To demonstrate that the performance gains obtained from training on our identified coresets are not specific to a particular architecture, we present cross-architectural results in Table 12. Models are trained on coresets selected by TCSL-CS ($r = 0.1$) using a ResNet backbone (ResNet18 for cMNIST and ResNet50 for all other datasets). We report WGA improvements relative to ERM baselines of the same target architectures. The results indicate that the selected coresets are architecture-agnostic, yielding consistent performance gains across different model architectures.

In Tables 13 and 14 we vary B and T_s in our algorithm, respectively, to support the effectiveness of our proposed coreset selection method. The results show that performance remains robust across these choices and consistently outperforms the strongest baselines on all four datasets.

Method	Group Info Train	Waterbirds WGA (AVG)	cMNIST WGA (AVG)	MetaShift WGA (AVG)	UrbanCars-B WGA (AVG)
TCSL-CS	x	91.91 $\pm$ 0.35 (92.83 $\pm$ 0.55)	<u>83.37</u> $\pm$ 1.33 (91.76 $\pm$ 0.44)	76.85 $\pm$ 4.24 (88.60 $\pm$ 1.12)	<u>84.27</u> $\pm$ 1.67 (86.93 $\pm$ 0.78)
TCSL-CS	✓	<u>92.06</u> $\pm$ 0.12 (93.32 $\pm$ 0.43)	88.26 $\pm$ 2.07 (94.01 $\pm$ 1.67)	85.42 $\pm$ 4.42 (88.02 $\pm$ 1.84)	85.20 $\pm$ 0.14 (86.70 $\pm$ 0.07)
RGlobal	✓	92.67 $\pm$ 0.58 (93.68 $\pm$ 0.41)	71.68 $\pm$ 13.83 (87.59 $\pm$ 2.24)	<u>80.79</u> $\pm$ 1.06 (87.85 $\pm$ 1.22)	84.00 $\pm$ 1.20 (86.57 $\pm$ 0.06)

Table 10: We conduct an ablation study on the TCSL_c score. Since $\alpha > 0.9$ for all datasets except MetaShift, we set $r = 0.1$ as in the main paper. For MetaShift, as $\alpha = 0.882$, we set $r = 0.2$. We use CB ERM for retraining on all of the identified coresets. The best WGA for each dataset is shown in **bold** and the second best value is underlined.

Method	Ours	GCE(q=0.7)	GCE(q=0.8)	GCE(q=0.9)
CSL(x^c)	0.568	0.637	0.639	0.641
CSL(x^s)	0.823	0.769	0.763	0.756

Table 11: Cosine similarity between the CSL scores obtained with models trained on only the core and spurious feature parts of images and the TCSL_s scores obtained from biased models trained with different methods. We compare our proposed biased model (f_s) against biased models trained with the generalized cross-entropy (GCE) loss.

Dataset	Original	ResNet101	InceptionV3	DenseNet121
Waterbirds	+10.76	+9.97	+8.10	+15.81
cMNIST	+26.76	+24.09	+28.20	+25.14
MetaShift	+6.25	+4.86	+4.86	+4.08
UrbanCars-B	+18.27	+18.80	+15.60	+13.30

Table 12: We evaluate performance gains relative to ERM baselines across different architectures. Each model is trained on the coresets ($r = 0.1$) identified by TCSL-CS, where the coreset selection is conducted using a ResNet backbone (ResNet18 for cMNIST and ResNet50 for all other datasets). The results show that the selected coresets are architecture-agnostic and transfer effectively across different model architectures.

Method	Waterbirds WGA (AVG)	MetaShift WGA (AVG)	UrbanCars-B WGA (AVG)
$B = 25$	91.22 $\pm$ 0.31 (92.75 $\pm$ 0.76)	79.49 $\pm$ 2.46 (87.85 $\pm$ 0.49)	<u>85.00</u> $\pm$ <u>1.13</u> (87.60 $\pm$ 0.14)
$B = 50$	91.91 $\pm$ 0.35 (92.83 $\pm$ 0.55)	79.40 $\pm$ 2.23 (84.95 $\pm$ 1.52)	84.27 $\pm$ 1.67 (86.93 $\pm$ 0.78)
$B = 100$	<u>91.75</u> $\pm$ <u>0.87</u> (92.53 $\pm$ 1.45)	<u>79.48</u> $\pm$ <u>3.44</u> (85.59 $\pm$ 1.23)	85.60 $\pm$ 0.28 (87.35 $\pm$ 0.07)

Table 13: We compare the WGA and AVG for different B at $r = 0.1$ across datasets. Results are averaged over 3 seeds. The best WGA for each dataset is shown in **bold** and the second-best value is underlined. $B = 50$ represents what was originally reported.

Method	Waterbirds WGA (AVG)	cMNIST WGA (AVG)	MetaShift WGA (AVG)	UrbanCars-B WGA (AVG)
$T_s = T/12$	92.28 $\pm$ 0.50 (93.18 $\pm$ 0.49)	82.88 $\pm$ 2.24 (88.78 $\pm$ 0.06)	80.86 $\pm$ 1.44 (85.97 $\pm$ 1.97)	83.72 $\pm$ 1.62 (87.17 $\pm$ 0.61)
$T_s = T/10$	91.91 $\pm$ 0.35 (92.83 $\pm$ 0.55)	83.37 $\pm$ 1.33 (91.76 $\pm$ 0.44)	79.40 $\pm$ 2.23 (84.95 $\pm$ 1.52)	84.27 $\pm$ 1.67 (86.93 $\pm$ 0.78)
$T_s = T/8$	<u>92.06</u> $\pm$ <u>0.52</u> (92.78 $\pm$ 0.36)	84.07 $\pm$ 1.94 (92.75 $\pm$ 0.77)	<u>80.14</u> $\pm$ <u>2.89</u> (85.60 $\pm$ 1.86)	81.85 $\pm$ 1.80 (85.60 $\pm$ 1.04)

Table 14: We compare the WGA and AVG for different T_s at $r = 0.1$ across datasets. Results are averaged over 3 seeds. The best WGA for each dataset is shown in **bold** and the second-best value is underlined. $T_s = T/10$ represents what was originally reported.

Dataset	Training (1 Epoch)	Coreset Selection (Total)
Waterbirds	6.576s	1.161s
cMNIST	25.30s	1.735s
MetaShift	3.201s	0.732s
UrbanCars-B	9.610s	1.392s

Table 15: Runtime of each stage of the TCSL algorithm. All measurements were obtained on a single NVIDIA A100 GPU.

J COMPUTATIONAL EFFICIENCY

Below, we analyze the runtime, computational complexity, and memory usage of our algorithm.

J.1 COMPLEXITY ANALYSIS

We assume that the cost of a single forward pass through a given deep neural network is F , and that the cost of one backpropagation step is B . Thus, the cost of one forward-backward pass is $F + B$. Let T_s denote the number of epochs used to train the biased model, T_c the number of epochs used to train the debiased model, and n the size of the training dataset.

Under standard empirical risk minimization (ERM), training for T epochs incurs a computational cost of $Tn(F + B)$. In TCSL, we use $T_s = T/10$ and $T_c = T$ by default. The TCSL scores are obtained during training without additional forward passes, since per-sample losses are already computed as part of optimization. Hence, the total computational cost of TCSL is $1.1Tn(F + B)$. Assuming $B \approx 2F$, ERM requires $\approx 3TnF$ operations, whereas TCSL requires $\approx 3.3TnF$ operations.

Common two-stage algorithms, including LC, CNC, LfF, JTT, and ULA, typically train two models for T epochs each. Their total computational cost is therefore $2Tn(F + B) = 6TnF$. By contrast, TCSL trains the biased model for only a fraction of the full training budget, resulting in lower computational overhead.

We note that the debiased model f_c trained within the TCSL framework can itself be used as a final model, since it follows the structure of two-stage and logit-correction based approaches. However, the primary objective of TCSL is to identify a **debiased coreset** such that a standard ERM model trained on this subset achieves state-of-the-art performance. Consequently, evaluating the quality of the selected coreset requires an additional ERM training run.

J.2 RUNTIME ANALYSIS

We report the average runtime per training epoch for TCSL across all four datasets, along with the total runtime of the coreset selection procedure in Table 15.

J.3 MEMORY USAGE ANALYSIS

Our implementation of wKMeans takes the biased model’s loss trajectories as input, which requires storing $T_s \cdot n$ scalar values that are then used to compute the $TCSL_s$ scores. The $TCSL_c$ scores are computed as the average per-sample losses over the debiased model’s training trajectory. These values are accumulated online during training, so only n scalar values are stored. Hence, the total memory overhead is $(T_s + 1) \cdot n$ scalar values. Compared with the datasets used in our experiments, which typically require $224 \cdot 224 \cdot 3 \cdot n$ scalar values for image storage, this introduces a negligible memory overhead of approximately 0.02% for $T = 300$. Furthermore, the losses can be stored in CPU memory to reduce GPU memory pressure.