
DeNetDM: Debiasing by Network Depth Modulation

Silpa Vadakkeveetil Sreelatha^{*1} Adarsh Kappiyath^{*1} Abhra Chaudhuri^{1,2,3} Anjan Dutta¹

¹ University of Surrey ² University of Exeter ³ Fujitsu Research of Europe

{s.vadakkeveetilsreelatha, a.kappiyath, anjan.dutta}@surrey.ac.uk, abhra.chaudhuri@fujitsu.com

Abstract

Neural networks trained on biased datasets tend to inadvertently learn spurious correlations, hindering generalization. We formally prove that (1) samples that exhibit spurious correlations lie on a lower rank manifold relative to the ones that do not; and (2) the depth of a network acts as an implicit regularizer on the rank of the attribute subspace that is encoded in its representations. Leveraging these insights, we present DeNetDM, a novel debiasing method that uses network depth modulation as a way of developing robustness to spurious correlations. Using a training paradigm derived from Product of Experts, we create both biased and debiased branches with deep and shallow architectures and then distill knowledge to produce the target debiased model. Our method requires no bias annotations or explicit data augmentation while performing on par with approaches that require either or both. We demonstrate that DeNetDM outperforms existing debiasing techniques on both synthetic and real-world datasets by 5%. The project page is available at <https://vssilpa.github.io/denetdm/>.

1 Introduction

Deep neural networks (DNNs) have made remarkable progress across various domains by delivering superior performance on large-scale datasets. However, while the benefits of training DNNs on large-scale datasets are undeniable, these algorithms also tend to inadvertently acquire unwanted biases Shah et al. (2020), hampering their generalization. For instance, a classifier predominantly trained to recognize camels in desert landscapes could encounter difficulties when attempting to identify a camel situated on a road Kim et al. (2021). While a certain degree of bias can enhance model performance, as exemplified by the assumption that cars usually travel on roads Choi et al. (2020), it remains critical to identify and address unwanted biases. Previous methods to address this problem rely on bias annotations as suggested in Majumdar et al. (2021); Kim et al. (2019); Sagawa et al. (2020); Wang et al. (2020), and may involve predefined bias types, such as texture bias mitigation approach in Geirhos et al. (2019). However, acquiring bias labels with human resources is expensive and time-consuming. Recent studies, including Nam et al. (2020) and Lee et al. (2021), have shifted towards debiasing methods without bias labels, with approaches like Nam et al. (2020) emphasizing bias-aligned samples and reweighting bias-conflicting samples, while others like Lee et al. (2021); Kim et al. (2021) introduce augmentation strategies to diversify bias-conflicting points.

We propose DeNetDM (**De**biasing by **Ne**twork **De**pth **Mo**dulation), a novel approach to automatically identify and mitigate spurious correlations in image classifiers without relying on explicit data augmentation or reweighting. We start by showing that a sample set that exhibits bias through spurious correlation of attributes lies on a manifold with an effective dimensionality (rank) lower than its bias-free counterpart. We then leverage this finding to formally derive a relationship between the depth of a network and the true rank of the attribute (not sample) subspace that it encodes. We find for a set of attributes that are equally likely to minimize the empirical risk, a deeper network prefers

^{*}Equal contribution.

to retain those with a lower rank, with a higher probability. This implies that the depth of a network acts as an implicit regularizer in the rank space of the attributes. We find that deeper networks tend to generalize based on bias attributes and shallower networks tend to generalize based on core attributes. This finding is in line with a number of works that show that deeper networks tend to learn low rank solutions in general (Roy and Vetterli, 2007; Huh et al., 2023; Wang and Jacot, 2024). Note, however that prior works do not establish the relationship between network depth and the rank of the attribute subspace, a link we establish in our work for the first time, to the best of our knowledge.

Our theoretical claims are confirmed by our preliminary empirical study on linear feature decodability, which quantifies the extent to which specific data attributes can be accurately and reliably extracted from a given dataset or signal. Our study focuses on the feature decodability of bias and core attributes in the neural networks of varying depths, following the approach outlined in Hermann and Lampinen (2020). Our observations in untrained neural networks reveal that the feature decodability tends to diminish as the networks become deeper. We also investigate how attribute decodability varies with Empirical Risk Minimization (ERM) based training on networks of varying depths.

Our hypothesis posits that in a task requiring deep and shallow branches to acquire distinct information, the deep branch consistently prioritizes bias attributes, while the shallow branch favors core attributes. We utilize a technique inspired by the Product of Experts (Hinton, 2002), where one expert is deeper than the other. Empirical analysis shows that the deep branch becomes perfectly biased and the shallow branch becomes relatively debiased by focusing solely on the core attributes by the end of the training. Since the shallow branch may lack the capacity to capture the nuances of the core attributes adequately due to less depth, we propose a strategy where we train a deep debiased model utilizing the information acquired from both deep (perfectly biased) and shallow (weak debiased) network in the previous phase. Our training paradigm efficiently facilitates the learning of core attributes from bias-conflicting data points to the debiased model of any desired architecture.

In summary, we make the following contributions: (1) We theoretically prove that the deep models prefer to learn spurious correlations compared to shallower ones, supported by empirical analysis of the decodability of bias and core attributes across neural networks of varying depths. (2) Building upon the insights from our decodability experiments, we present a novel debiasing approach that involves training both deep and shallow networks to obtain a desired debiased model. (3) We perform extensive experiments and ablation studies on a diverse set of datasets, including synthetic datasets like Colored MNIST and Corrupted CIFAR-10, as well as real-world datasets, Biased FFHQ, BAR and CelebA, demonstrating an approximate 5% improvement over existing methods.

2 Related Works

Several works, such as Hermann and Lampinen (2020); Mehrabi et al. (2021), have highlighted neural networks' vulnerability to spurious correlations during empirical risk minimization training. Recently, various debiasing techniques have emerged, which can be categorized as follows.

Supervision on bias: A variety of approaches (e.g., Majumdar et al. (2021); Kim et al. (2019); Sagawa et al. (2020); Wang et al. (2020)) assume readily accessible bias labels for bias mitigation. Some approaches assume prior knowledge on specific bias types without using explicit annotations, like texture bias in Wang et al. (2019); Ge et al. (2021); Geirhos et al. (2019). Recent works such as Karimi Mahabadi et al. (2020); Clark et al. (2019) apply the Product of Experts method to mitigate bias in natural language processing, assuming a biased expert's availability. However, obtaining bias labels can be resource-intensive. In contrast, DeNetDM, our proposed method, does not require pre-access to bias labels or types. Instead, it leverages diverse network architecture depths within the Product of Experts framework to implicitly capture relevant bias and core attributes.

Utilization of pseudo bias-labels: Recent approaches avoid explicit bias annotations by obtaining pseudo-labels through heuristics to identify biased samples. One heuristic suggests that biases easy to learn are captured early in training, as seen in Nam et al. (2020); Lee et al. (2021); Liu et al. (2023); Kim et al. (2021); Tiwari and Shenoy (2023); Lee et al. (2023). Nam et al. (2020) employ generalized cross-entropy loss to identify and reweight bias-conflicting points. On the other hand, Lee et al. (2021) augment features of bias-conflicting points for debiasing, while Liu et al. (2023) employ logit correction and group mixup techniques to diversify bias-conflicting samples. Other methods like Sohoni et al. (2020) and Seo et al. (2022) acquire pseudo-bias labels through clustering in biased

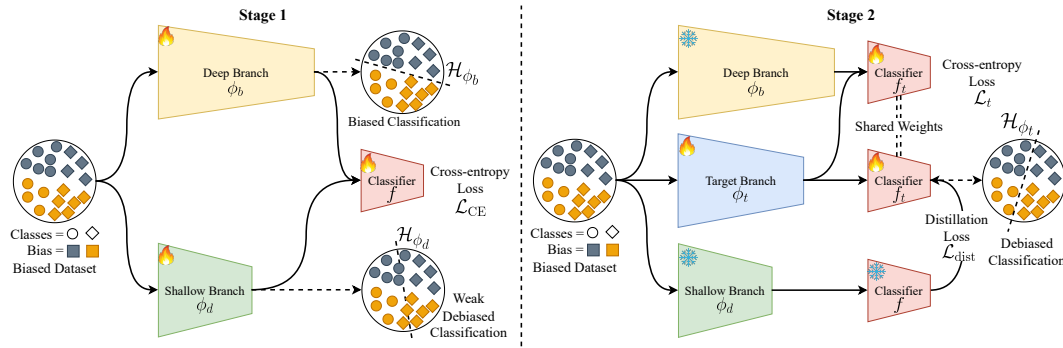


Figure 1: **Illustration of the DeNetDM framework:** In Stage 1, an ensemble of shallow and deep branches produces outputs linearly combined and trained as a product of experts. The cross-entropy loss with depth modulation aids in separating biases and identifying target attributes. In Stage 2, we further introduce a target branch with the desired architecture, which also requires debiasing. This phase exclusively focuses on refining the target branch’s feature extractor (ϕ_t) and classifier head (f_t) while leveraging knowledge from the initial stages.

network feature spaces. Our approach does not explicitly require pseudo-bias labels; it implicitly uses them during training to learn both biased and debiased models.

Dependence on network architectures: Diffenderfer et al. (2021) employ lottery-ticket-style pruning algorithms for compressed robust architectures. Similarly, approaches like Park et al. (2023); Zhang et al. (2021) introduce pruning to extract robust subnetworks. Our method aligns with this category but does not target specific robust subnetwork discovery. Instead, we utilize training dynamics of varied-depth architectures to enhance debiasing. Meanwhile, Shrestha et al. (2022) applies Occam’s razor principle to optimize network depth and visual regions, enhancing overall robustness. Both DeNetDM and OccamNets (Shrestha et al., 2022) aim to simplify learning for better generalization and reduced spurious correlations. DeNetDM uses depth modulation with separate deep and shallow branches to address bias – where the shallow model captures biases and the deep model learns complex, unbiased patterns. In OccamNets, simplicity is a core design principle, with the architecture adaptively minimizing complexity on a per-sample basis. Both methods tackle spurious correlations without extra annotations or data augmentation but through distinct architectural strategies.

3 Debiasing by Network Depth Modulation

First, we theoretically justify that the deeper models are more inclined to learn spurious correlations compared to shallow networks, as discussed in Section 3.1. We then provide empirical evidence to support our theoretical claims by utilizing feature decodability, detailed in Section 3.2. Based on these, we introduce DeNetDM, a debiasing approach centered on network depth modulation. Our training process comprises two stages: initially, a deep and shallow network pair is trained using a training paradigm that originates from Products of Experts (Hinton, 2002), yielding both biased and debiased models, which is detailed in Section 3.3. Subsequently, recognizing the limitations of the shallow debiased model in capturing core feature complexities due to its depth, we proceed to train a target debiased model, ensuring it possesses the same or higher depth compared to the deep biased model. This phase leverages information acquired from the biased and debiased models in the previous step, as elaborated in Section 3.4. An illustration of DeNetDM is provided in Figure 1.

Notations: We operate on a dataset X , where a fraction of the data points, denoted with X_a , are bias-aligned and the remaining points, denoted with X_c , are bias conflicting. Let $\phi : X \rightarrow \mathbb{R}^n$ be an encoder that produces an embedding $z \in \mathbb{R}^n$ for an input $x \in X$. We denote the effective rank (Roy and Vetterli, 2007) of a matrix A as $\rho(A)$, which gives us a continuous notion of the size of the span (rank) of A , a quantity that is maximized under equally distributed singular values, and minimized when a single singular value dominates over the rest (Huh et al., 2023). Let B and C be the set of bias and core attributes respectively, both with strictly positive ranks, defining bases that are orthogonal to each other, i.e., $B \perp C$. A summary of notations is provided in Section 7.1.

3.1 Simplicity Bias and Spurious Correlations

Debiasing with network depth modulation requires understanding how the depth of a neural network affects its learning of bias-aligned or bias-conflicting subsets of X with lower generalization error. These results finally let us build up to our finding that deeper networks are more susceptible to learning spurious features over their shallower counterparts. All proofs are deferred to Section 7.2.

Definition 1 (Stability). A partitioning $X = X_1 \cup X_2 \dots \cup X_m$ of a sample set X is stable wrt. an attribute ω when:

$$P(X_i^\omega) = P(X^\omega); \forall i \in [1, m],$$

where X^ω and X_i^ω are the respective subspaces of X and X_i corresponding to the attribute ω , and $P(\cdot)$ is the associated probability distribution.

For example, if ω follows a uniform distribution in X , a stable partitioning would ensure that each of the partitions X_i also have ω distributed uniformly. Stability ensures that a partitioning does not introduce sampling bias into any of the partitions wrt. a particular attribute.

Theorem 1 (Partition Rank). When the partitioning $X = X_a \cup X_c$ is stable wrt. C , the rank of the bias-aligned partition is upper-bounded by the rank of the bias-conflicting partition, i.e.,

$$\text{rank}(X_a) \leq \text{rank}(X_c)$$

Intuition: The theorem assumes a stable partitioning of the sample set X . It implies that, in both the bias-aligned and conflicting subsets, the distribution of the core attributes are equal to that of the original sample set, i.e., $P(X_a^C) = P(X_c^C) = P(X^C)$. Under this condition, the only component in either of the subsets that determines the subset's rank should be the bias attributes, assuming (without loss of generality) that the attribute space is made up of only the core and the bias attributes. The proof proceeds by establishing that the rank of the bias attributes is lower in the bias-aligned points (resulting from the lack of intra-class variation due to spurious correlation with the class label) than in the bias-conflicting points.

Theorem 2 (Depth-Rank Duality). Let $\mathcal{A} = [A_0, A_1, \dots, A_n]$ be the attribute subspace of X with increasing ranks, i.e., $\text{rank}(A_0) < \text{rank}(A_1) < \dots < \text{rank}(A_n)$, such that every $A \in \mathcal{A}$ is maximally and equally informative of the label Y , i.e., $I(A_0, Y) = I(A_1, Y) = \dots = I(A_n, Y)$. Then, across the depth of the encoder ϕ , SGD yields a parameterization that optimizes the following objective:

$$\underbrace{\min_{\phi, f} \mathcal{L}(f(\phi(X)), Y)}_{\text{ERM}} + \min_{\phi} \sum_d \left\| \phi[d](\tilde{X}) - \Omega^d \odot \mathcal{A} \right\|_2, \quad (1)$$

where $\mathcal{L}(\cdot, \cdot)$ is the empirical risk, $f(\cdot)$ is a classifier head, $\phi[i](\cdot)$ is the output of the encoder ϕ (optimized end-to-end) at depth d , $\|\cdot\|_2$ is the l^2 -norm, $\odot$ is the element-wise product, $\tilde{X}$ is the l_2 -normalized version of X , $\Omega^d = [\mathbb{1}_{\pi_1(d)}; \mathbb{1}_{\pi_2(d)}; \dots; \mathbb{1}_{\pi_n(d)}]$, $\mathbb{1}_\pi$ is a random binary function that outputs 1 with a probability π , and $\pi_i(d)$ is the propagation probability of A_i at depth d bounded as:

$$\pi_i(d) = \mathcal{O}(\rho(\phi[d]) r_i^{-d}), \quad (2)$$

where $\rho(\phi[d])$ is the effective rank of the $\phi[d]$ representation space, and $r_i = \text{rank}(A_i)$.

Intuition: For a set of attributes, all of which equally minimize the training loss, Theorem 2 describes the strategy adopted by SGD to parameterize a neural encoder, for capturing the above set of attributes. At a given depth d of the encoder ϕ (represented as $\phi[d]$), each attribute $A_i \in \mathcal{A}$ gets encoded in the representation space of $\phi[d]$ according to its corresponding probability mass $\pi_i(d)$. According to Equation (2), the probability of survival of all attributes decrease with increasing depth. However, the probability of survival of an attribute with a higher rank drops faster with increasing depth than that of one with a lower rank, prioritizing the usage of lower rank attributes at greater depths. In other words, the depth of a network acts as an implicit regularizer in the attribute rank space.

As an example, say, a neural network ϕ of depth d (denoted as $\phi[d]$) has $3K$ available dimensions, and of depth $D > d$, $\phi[D]$ has K available dimensions (the rank reduction with increasing depth stemming from the simplicity bias (Huh et al., 2023; Wang and Jacot, 2024)). Say the attribute space it has to learn from is composed of two attributes: (1) A_0 , with a rank of K , and (2) A_1 , with a rank of $K + i$, where $1 \leq i \leq K$, where both A_0 and A_1 are equal minimizers of the empirical risk.

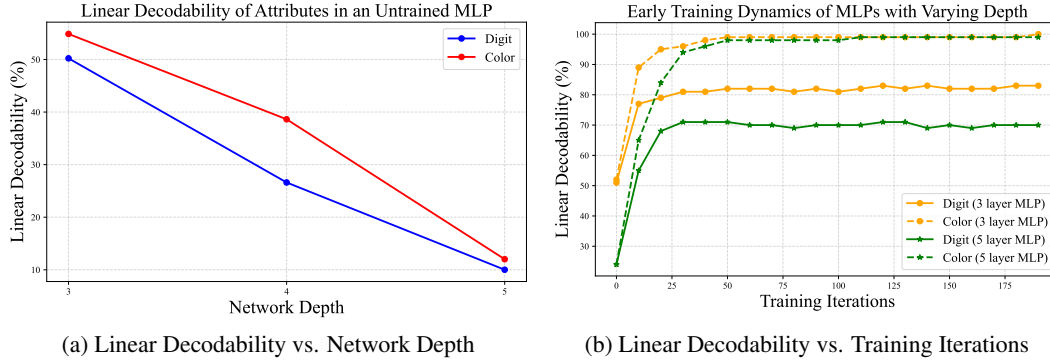


Figure 2: **Exploring the effect of depth modulation:** (a) illustrates how the linear decodability of features decreases as neural network depth increases, while (b) dives into the training dynamics of MLPs with varying depths under ERM.

So, according to Theorem 2, at $\phi[d]$, the encoder has no constraint over the number of attributes it can accommodate, since $3K \geq 2K + i$. However, at depth D , $\phi[D]$ can only choose an attribute with K dimensions. Since both A_0 and A_1 result in the same solution for ERM (Empirical Risk Minimization), SGD would parameterize $\phi[D]$ to capture A_0 with a higher probability.

3.2 Effect of Depth Modulation

Theorem 2 establishes a relationship between the depth of a network and the nature of the features it learns in terms of its rank. To empirically validate this, we probe MLPs of depths 3, 4, and 5, using the feature decodability technique proposed by Hermann and Lampinen (2020), to uncover the types of features that get encoded in them. We use the Colored MNIST dataset Nam et al. (2020) (CMNIST), where digit identity (core attribute) is spuriously correlated with color (bias attribute). We experiment with the decodability of the digit identity and color attributes in the CMNIST dataset. Additional information on the computation of feature decodability can be found in Section 7.5. We regard digit identity to have a higher rank than that of color, due to its higher representational complexity / information content in terms of the number of bits required for storage, a notion also confirmed in the experiments of Hermann and Lampinen (2020). We start by looking at the decodabilities at random initialization of the networks, and interestingly observe in Figure 2a that the decodabilities of both attributes decrease with increasing depth, but that of digit identity drops faster than color. Since at random initialization, there is no notion of empirical risk, the $\min \mathcal{L}(\cdot, \cdot)$ term in Theorem 2 is cancelled out. Thus, the observation aligns with our prediction of the second term in $\|\cdot\|_2$ of Theorem 2 that the higher the rank of a feature, the less likely it is to get encoded in the later layers, the theoretically predicted behavior specifically for random networks being discussed in Corollary 2.1. We then proceed to investigating how feature decodability evolves during the early stages of Empirical Risk Minimization (ERM) training across the networks of varying depths, *i.e.*, under the presence of $\min \mathcal{L}(\cdot, \cdot)$, the results of which are summarized in Figure 2b. We perform similar linear decodability analysis on C-CIFAR10 dataset and the observations are presented in Section 7.6.1.

As observed in Figure 2b, the initial phases of training for both networks emphasize color attribute (since bias is easy to learn), leading to notable improvements in color decodability for both models. Also, as training progresses, the 3-layer model exhibits higher digit decodability compared to the 5-layer model. Hence, the difference in decodability between color and digit attributes becomes more pronounced in the 5-layer compared to the 3-layer MLP. This again confirms the prediction of our Theorem 2 that when two attributes equally minimize the empirical risk, a deeper network is more likely to select the one with a lower rank, while a shallower network will try to accommodate as much of both as possible. Based on these observations, the deep models may prefer bias attributes, while shallow models focus on core attributes when tasked with capturing distinct information.

This prompts us to explore whether similar behavior can be induced in models of equal depth. In this scenario, both models, undergoing ERM training, may exhibit a similar trend, with the disparity in decodability between biased and core attributes becoming nearly identical in both models due to same depth. Consequently, when compelling each model to learn distinct information, they may

capture biased or core attributes, or even divide attribute information between them, leading to a loss of control over the bias identification process. We also present empirical evidence in Table 5 to support these claims. Therefore, using models of different depths introduces an inductive bias suitable for the bias identification process.

3.3 Stage1: Segregation of Bias & Core Attributes

Theorem 1 predicts that bias-aligned points lie on a lower-rank manifold than bias conflicting points. Theorem 2 predicts that as we go deeper into a neural network, the likelihood that a higher rank feature, that equally minimizes the empirical risk as that of other lower rank features, is retained, decays exponentially with depth. Based on this, we present a training procedure to obtain the biased and debiased classifier for an M class classification problem. Let ϕ_b and ϕ_d denote the parameters of the feature extractors associated with the deep and shallow branches, where $\text{depth}(\phi_b) > \text{depth}(\phi_d)$. We use f to represent the classifier head shared by ϕ_b and ϕ_d . Here, f , ϕ_b and ϕ_d are trainable parameters. Considering an image-label pair (x, y) , the objective function is expressed as:

$$\mathcal{L}_{\text{CE}}(\hat{p}, y) = - \sum_{c=1}^M y_c \log(\hat{p}_c) \quad (3)$$

where $\hat{p} = \text{softmax}(f(\alpha_b \phi_b(x) + \alpha_d \phi_d(x)))$. If we set $\alpha_b = \alpha_d = 1$ throughout the training process, we get:

$$\hat{p} = \text{softmax}(f(\phi_b(x) + \phi_d(x))) \quad (4)$$

To evaluate the performance of an individual expert, we assign a value of 1 to the corresponding α while setting the other α equal to 0.

Our training methodology is derived from the Products of Experts technique (Hinton, 2002) where multiple experts are combined to make a final prediction, and each expert contributes to the prediction with a weight. However, in our approach, the role of the experts is assumed by ϕ_b and ϕ_d , whose features are combined through weighted contributions. The conjunction of features is then passed to the shared classifier to generate predictions. We provide a detailed proof elucidating the derivation of Equation (4) through the Product of Experts in Section 7.3 of the Appendix. Due to the architectural constraints we imposed by modulating their capacities, the deep expert tends to prioritize the learning of bias attribute, while the shallow expert is inclined towards the core attribute. The model leverages the strengths of both experts to effectively learn from their combined knowledge. We investigate the training dynamics in Section 4.3.

3.4 Stage2 : Training the Target Debiased Model

The initial phase effectively separates the bias and core attributes into deep and shallow branches, respectively. However, relying solely on the debiased shallow branch may not be practical, as it might not capture the complex features representing the core attributes, given the less depth of the shallow model. This limitation does not apply to the deep biased model. To tackle this challenge, we introduce a target branch with the desired architecture for debiasing.

Let ϕ_t be the parameters of the feature extractor associated with the target branch and f_t be the classifier head whose weights are initialized using the weights of f . During this phase, our training is exclusively focused on ϕ_t and f_t . We freeze ϕ_b and ϕ_d since we leverage these models to only extract the necessary knowledge for debiasing the target branch. To capture information orthogonal to ϕ_b , we employ the same training approach described in Section 3.3, where ϕ_b and ϕ_t serve as the experts. The objective function can be written as:

$$\mathcal{L}_t(\hat{p}, y) = - \sum_{c=1}^M y_c \log(\hat{p}_c)$$

where

$$\hat{p} = \text{softmax}(f_t(\beta_b \phi_b(x) + \beta_t \phi_t(x))) \quad (5)$$

The training and evaluation of the experts follow the procedure described in Section 3.3, with the key difference being that in this phase, only a single expert, ϕ_t , which is the target branch and classifier f_t , undergoes updates.

We further leverage the knowledge pertaining to the core attributes, which is encapsulated in ϕ_d , by transferring this knowledge to the target branch ϕ_t through knowledge distillation. Here, ϕ_t acts as the student, whereas ϕ_d corresponds to the teacher. We set $\beta_b = 0$ and $\beta_t = 1$ in Equation (5) to obtain the predictions of the student $\hat{\phi}_t$. Therefore, the distillation loss is given by :

$$\mathcal{L}_{\text{dist}}(\hat{p}_t, \hat{p}_s) = - \sum_{c=1}^M \hat{p}_{t_c} \log(\hat{p}_{s_c}) \quad (6)$$

$$\text{where} \quad \hat{p}_s = \text{softmax} \left(\frac{f_t(\phi_t(x))}{\tau} \right) \quad (7) \quad \hat{p}_t = \text{softmax} \left(\frac{f(\phi_d(x))}{\tau} \right) \quad (8)$$

where λ is a hyperparameter chosen from the interval $[0, 1]$. The pseudocode for the entire training process of DeNetDM is provided in Section 7.4.

4 Experiments

In this section, we discuss the experimental results and analysis to demonstrate the effectiveness of DeNetDM training in debiasing. We evaluate the performance of the proposed approach by comparing it with the previous methods in debiasing, utilizing well-known datasets with diverse bias ratios, consistent with the prior works in debiasing. Additionally, we conduct an empirical study to analyze the training dynamics of DeNetDM. We also perform ablation studies to assess the effectiveness of individual components within the proposed approach.

4.1 Experimental Setup

Datasets: We evaluate the performance of DeNetDM across diverse domains using two synthetic datasets (Colored MNIST Ahuja et al. (2020), Corrupted CIFAR10 Hendrycks and Dietterich (2019)) and three real-world datasets (Biased FFHQ Kim et al. (2021), BAR Nam et al. (2020)) and CelebA Liu et al. (2015). In Colored MNIST (CMNIST), the digit identity is spuriously correlated with color, while in Corrupted CIFAR10 (C-CIFAR10), the texture noise corrupts the target attribute. Biased FFHQ (BFFHQ) comprises human face images from the FFHQ dataset Karras et al. (2019) such that the age attribute is spuriously correlated with gender. BAR consists of human action images where six human action classes are correlated with six place attributes. We conduct experiments by varying the ratio of bias-conflicting points in the training set to demonstrate the efficacy of our approach across diverse scenarios. Following the experimental settings used by the previous works Liu et al. (2023); Lee et al. (2021); Qi et al. (2022), we vary the ratio of bias-conflicting samples, specifically setting it at $\{0.5\%, 1\%, 2\%, 5\%\}$ for CMNIST and C-CIFAR10, $\{0.5\%\}$ in BFFHQ and $\{1\%, 5\%\}$ in BAR datasets. We employ a subsampled version of CelebA as described in Hong and Yang (2021), maintaining the same data splits for consistency.

Baselines: We compare the performance of our proposed approach to the following bias mitigation techniques; ERM Vapnik (1999), GDRO Sagawa et al. (2020), LfF Nam et al. (2020), JTT Liu et al. (2021), DFA Lee et al. (2021) and LC Liu et al. (2023). Among these, GDRO utilizes supervision on bias whereas LfF and JTT assumes no prior knowledge on the bias labels. DFA and LC utilizes augmentation techniques to increase diversity of minority groups. More details on the baselines are provided in Section 7.8.2 of the Appendix.

Evaluation protocol: We evaluate CMNIST and C-CIFAR10 on unbiased test sets, with target features randomly correlated to spurious features, following the evaluation protocol commonly used in prior debiasing works Nam et al. (2020); Liu et al. (2021); Lee et al. (2021). Nevertheless, for BFFHQ, we do not use the unbiased test set since half of them are bias-aligned points. To ensure fair evaluation on debiasing, we adhere to previous methods Liu et al. (2023); Lee et al. (2021) by exclusively utilizing a test set comprising bias-conflicting points from the unbiased test set. Notably, the BAR test set consists solely of bias-conflicting samples, posing a significant evaluation challenge. Our primary metric is accuracy, with aligned accuracy and conflicting accuracy calculated separately for some ablations on CMNIST and C-CIFAR10 (see Section 4.4). Aligned accuracy is computed solely on bias-aligned data points while conflicting accuracy is determined exclusively based on the bias-conflicting points. For CelebA, we report worst-group accuracy specifically focusing on the bias-conflicting group (Blonde Hair = 0, Male = 0), which contains a substantial number of samples.

Table 1: Testing accuracy on CMNIST and C-CIFAR10, considering diverse percentages of bias-conflicting samples. Baseline results for C-CIFAR10 are taken from Liu et al. (2023), as we employ the same experimental settings. For CMNIST, we utilize the official repositories to obtain the models. Model requirements for spurious attribute annotations (type) are indicated by $\times$ (not required) and $\checkmark$ (required).

Methods	Group Info	CMNIST				C-CIFAR10			
		0.5	1.0	2.0	5.0	0.5	1.0	2.0	5.0
Group DRO	$\checkmark$	59.67	71.33	76.30	84.40	33.44	38.30	45.81	57.32
ERM	$\times$	35.34 (0.13)	50.34 (0.16)	62.29 (1.47)	77.63 (0.13)	23.08 (1.25)	25.82 (0.33)	30.06 (0.71)	39.42 (0.64)
JTT	$\times$	53.03 (3.89)	61.68 (2.02)	74.23 (3.21)	85.03 (1.10)	24.73 (0.60)	26.90 (0.31)	33.40 (1.06)	42.20 (0.31)
LfF	$\times$	63.39 (1.97)	74.01 (2.21)	80.48 (0.45)	85.39 (0.94)	28.57 (1.30)	33.07 (0.77)	39.91 (0.30)	50.27 (1.56)
DFA	$\times$	59.12 (3.15)	71.04 (1.02)	82.86 (2.27)	88.29 (1.50)	29.95 (0.71)	36.49 (1.79)	41.78 (2.29)	51.13 (1.28)
LC	$\times$	63.48 (5.22)	78.41 (1.95)	83.63 (1.43)	88.18 (1.59)	34.56 (0.69)	37.34 (1.26)	47.81 (2.00)	54.55 (1.26)
DeNetDM	$\times$	74.72 (0.99)	85.22 (0.76)	89.29 (0.51)	93.54 (0.22)	38.93 (1.16)	44.20 (0.77)	47.35 (0.70)	56.30 (0.42)

Table 2: Testing accuracy on BAR, BFFHQ, and CelebA. The test set for BAR and BFFHQ contains only bias-conflicting samples. Baseline method results are derived from Lim et al. (2023) for BAR, Liu et al. (2023) for BFFHQ, and Park et al. (2023) for CelebA on the same dataset split since we utilize identical experimental settings.

Methods	Group Info	BAR		BFFHQ	CelebA
		1.0	5.0	1.0	-
ERM	$\times$	57.65 (2.36)	68.60 (2.25)	56.7 (2.7)	47.02
JTT	$\times$	58.17 (3.30)	68.53 (3.29)	65.3 (2.5)	76.80
LfF	$\times$	57.71 (3.12)	67.48 (0.46)	62.2 (1.6)	-
DFA	$\times$	52.31 (1.00)	63.50 (1.47)	63.9 (0.3)	65.26
LC	$\times$	70.94 (1.46)	74.32 (2.42)	70.0 (1.4)	-
DeNetDM (ours)	$\times$	73.84 (2.56)	79.61 (3.18)	75.7 (2.8)	81.04

We conduct five independent trials with different random seeds and report both the mean and standard deviation to ensure statistical robustness.

Implementation details: We perform extensive hyperparameter tuning using a small unbiased validation set with bias annotations to obtain the deep and shallow branches for all the datasets. We consistently utilize the same debiasing model architectures used by the previous methods for our target branch to ensure a fair comparison. Additionally, a linear layer is employed for the classifier for all the datasets. The additional architecture details for different datasets are as follows: **(1) CMNIST:** we use an MLP with three hidden layers for the deep branch and an MLP with a single hidden layer corresponding to the shallow branch. During the second phase of DeNetDM, we use an MLP with three hidden layers for the target branch. **(2) C-CIFAR10, BAR:** we use the ResNet-20 architecture for the deep branch and a 3-layered CNN model for the shallow branch. The target branch used in the second stage of DeNetDM is ResNet-18. **(3) BFFHQ, CelebA:** we use the ResNet-18 architecture as the biased branch and a 4-layered CNN as the shallow branch. We also use the ResNet-18 architecture for the target branch, following the approaches of Liu et al. (2023); Lee et al. (2021). Further details on the datasets and implementation are presented in Section 7.8.

4.2 Evaluation Results

We present a comprehensive comparison of DeNetDM with all the baselines described in Section 4.1 across varying bias conflicting ratios on CMNIST, C-CIFAR10, BFFHQ, BAR and CelebA in Table 1 and Table 2 respectively. As evident from Table 1 and Table 2, DeNetDM consistently outperforms all baselines across different bias ratios for CMNIST, BFFHQ, BAR and CelebA datasets. Notably, on the C-CIFAR10 dataset, DeNetDM exhibits superior performance when bias ratios are at 0.5%, 1%, and 5%, and closely aligns with LC Liu et al. (2023) in the case of 2%. These findings provide evidence for the practical applicability of DeNetDM. It is worth mentioning that the proposed approach demonstrates a significant performance enhancement across all datasets compared to Group DRO, which relies on predefined knowledge of bias. DeNetDM achieves this improvement without any form of supervision on the bias, highlighting the effectiveness of depth modulation in the debiasing.

An intriguing observation from Table 1 is that DeNetDM demonstrates better performance compared to the baselines when the bias-conflicting ratio is lower, particularly evident in the C-CIFAR10 dataset.

We believe that the effectiveness of inductive bias enforced by DeNetDM in distinguishing between core and bias attributes is superior to that of LC, thereby allowing it to adeptly capture core attributes even when dealing with data points that exhibit fewer bias conflicting points. This emphasizes the applicability of DeNetDM in scenarios where the training data exhibits a significant amount of spurious correlations. Another noteworthy observation in Table 2 is that DeNetDM outperforms LC and DFA by a considerable margin across all datasets, particularly on the complex real-world datasets, BAR and BFFHQ. Both LC and DFA rely on augmentations to enhance the diversity of bias-conflicting points, whereas our approach utilizes depth modulation to efficiently capture the core attribute characteristics in the existing training data. Despite this, DeNetDM still achieves superior performance compared to LC and DFA without relying on augmentations.

4.3 Analysis of Training Dynamics

In Section 3.2, we discussed the variability in linear decodability at various depths and its significance as a motivation for debiasing. To further validate this intuition and identify the elements contributing to its effectiveness, we delve into the training dynamics of DeNetDM during initial stages. We consider the training of Colored MNIST with 1% skewness due to its simplicity. Figure 3 shows how linear decodability of attributes varies across different branches of DeNetDM during training. As depicted in Figure 3, prior to training, the deep branch demonstrates lower linear decodability for both the digit identity (core attribute) and color (bias attribute) compared to the shallow branch. As training progresses, the bias attribute, easier to learn, rapidly increases in linear decodability in both branches, labeled ‘A’ in Figure 3.

Here, the disparity in linear decodability between digit identity and color attributes becomes more pronounced in the deep branch than in the shallow one. This distinction serves as a prior, influencing the deep branch to effectively capture the bias. Since we employ Product of Experts technique, the deep branch becomes proficient in classification using the spurious attribute, thereby compelling the shallow branch to rely on other attributes such as digit for the classification. It is worth noting that the linear decodability of core attributes is more pronounced in the shallow branch, allowing them to capture the core attributes. Thus, the training paradigm of DeNetDM leads to a shallow branch that is robust to spurious correlations, and a deep branch that majorly relies on the biased attribute. This analysis confirms our intuition and provides empirical evidence of effective debiasing.

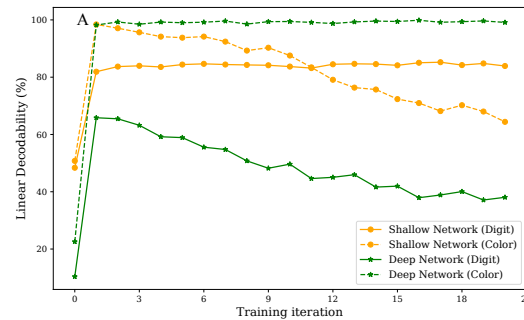


Figure 3: Early training dynamics of DeNetDM. This analysis confirms our intuition and provides empirical evidence of effective debiasing.

4.4 Ablation Studies

We perform several ablation studies to evaluate different facets of DeNetDM. We scrutinize the effect of various loss components on the performance of DeNetDM. Additionally, we explore the influence of network depth, a fundamental element of DeNetDM, and the sensitivity of DeNetDM to number of parameters which are discussed in Section 7.6. All the experiments are conducted on CMNIST and C-CIFAR10 datasets where the ratio of conflicting points is set to 1%. Additional experiments and ablations are also provided in Section 7.6.

Table 3: Ablation study of different losses used in DeNetDM on C-CIFAR10.

$\mathcal{L}_{CE}$ (Stage-1)	$\mathcal{L}_{dist}$ (Stage-2)	$\mathcal{L}_t$ (Stage-2)	Accuracy (%)	Conflicting Accuracy (%)	Aligned Accuracy (%)
✓	-	-	37.47	37.42	72.40
✓	-	✓	42.89	35.74	81.60
✓	✓	-	42.25	38.34	68.52
✓	✓	✓	43.12	39.46	69.53

Effect of loss components: We conduct ablation studies on C-CIFAR10 by selectively removing components to analyze their impact on the testing set accuracy as well as accuracy on bias-aligned and bias-conflicting points. The results are summarized in Table 3. When considering $\mathcal{L}_{CE}$ alone,

corresponding to the first stage of DeNetDM involving depth modulation, the model achieves 37.42% accuracy, showing a strong ability to learn target attributes. However, introducing the second stage of DeNetDM training with $\mathcal{L}_t$ alone leads to capturing significant bias information alongside core attributes, evidenced by high accuracy on aligned points (81.60%). When introducing $\mathcal{L}_{\text{dist}}$ alone, the model distills knowledge from the shallow branch obtained in the first stage, resulting in performance similar to stage 1 training. However, performing the second stage of DeNetDM training using both $\mathcal{L}_t$ and $\mathcal{L}_{\text{dist}}$ prevents capturing bias, focusing more on learning core features and resulting in improved conflicting and overall accuracy. A similar trend can be observed for CMNIST dataset and the results are summarized in Section 7.7.2.

5 Conclusion

We introduce DeNetDM, a novel debiasing method leveraging variations in linear decodability across network depths. Through extensive theoretical and experimental analysis, we uncover insights into the interplay between network architecture, attribute decodability, and training methodologies. DeNetDM employs paired deep and shallow branches inspired by the Product of Experts methodology, transferring debiasing capabilities to the desired architecture. By modulating network depths, it captures core attributes without explicit reweighting or data augmentation. Extensive experiments across various datasets, including synthetic ones like Colored MNIST and Corrupted CIFAR-10, as well as real-world datasets like Biased FFHQ and BAR, validate its robustness and superiority. Importantly, DeNetDM achieves performance comparable to supervised approaches, even without bias annotations.

6 Acknowledgments

Silpa Vadakkeveetil Sreelatha is partly supported by the Pioneer Centre for AI, DNRG grant number P1.

References

- Kartik Ahuja, Karthikeyan Shanmugam, Kush Varshney, and Amit Dhurandhar. Invariant risk minimization games. In *ICML*, 2020.
- Sungha Choi, Joanne T. Kim, and Jaegul Choo. Cars can’t fly up in the sky: Improving urban-scene segmentation via height-driven attention networks. In *CVPR*, 2020.
- Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. Don’t take the easy way out: Ensemble based methods for avoiding known dataset biases. In *EMNLP-IJCNLP*, 2019.
- James Diffenderfer, Brian R Bartoldson, Shreya Chaganti, Jize Zhang, and Bhavya Kaillkhura. A winning hand: Compressing deep networks can improve out-of-distribution robustness. In *NeurIPS*, 2021.
- Songwei Ge, Shlok Mishra, Chun-Liang Li, Haohan Wang, and David Jacobs. Robust contrastive learning using negative samples with diminished semantics. In *NeurIPS*, 2021.
- Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019.
- Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.
- Katherine Hermann and Andrew Lampinen. What shapes feature representations? exploring datasets, architectures, and training. In *NeurIPS*, 2020.
- Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 2002.
- Youngkyu Hong and Eunho Yang. Unbiased classification through bias-contrastive and bias-balanced learning. In *NeurIPS*, 2021.
- Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *TMLR*, 2023.

- Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. End-to-end bias mitigation by modelling biases in corpora. In *ACL*, 2020.
- Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.
- Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. Learning not to learn: Training deep neural networks with biased data. In *CVPR*, 2019.
- Eungyeup Kim, Jiyeon Lee, and Jaegul Choo. Biaswap: Removing dataset bias with bias-tailored swapping augmentation. In *ICCV*, 2021.
- Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanus Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. Wilds: A benchmark of in-the-wild distribution shifts. In *ICML*, 2021.
- Jungsoo Lee, Eungyeup Kim, Juyoung Lee, Jiyeon Lee, and Jaegul Choo. Learning debiased representation via disentangled feature augmentation. In *NeurIPS*, 2021.
- Jungsoo Lee, Jeonghoon Park, Daeyoung Kim, Juyoung Lee, Edward Choi, and Jaegul Choo. Revisiting the importance of amplifying bias for debiasing. In *AAAI*, 2023.
- Jongin Lim, Youngdong Kim, Byungjai Kim, Chanho Ahn, Jinwoo Shin, Eunho Yang, and Seungju Han. Biasadv: Bias-adversarial augmentation for model debiasing. In *CVPR*, 2023.
- 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 *ICML*, 2021.
- Sheng Liu, Xu Zhang, Nitesh Sekhar, Yue Wu, Prateek Singhal, and Carlos Fernandez-Granda. Avoiding spurious correlations via logit correction. In *ICLR*, 2023.
- Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Attributes in the Wild. In *ICCV*, 2015.
- Puspita Majumdar, Richa Singh, and Mayank Vatsa. Attention aware debiasing for unbiased model prediction. In *ICCVW*, 2021.
- Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 2021.
- Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: De-biasing classifier from biased classifier. In *NeurIPS*, 2020.
- Geon Yeong Park, Sangmin Lee, Sang Wan Lee, and Jong Chul Ye. Training debiased subnetworks with contrastive weight pruning. In *CVPR*, 2023.
- Jiaxin Qi, Kaihua Tang, Qianru Sun, Xian-Sheng Hua, and Hanwang Zhang. Class is invariant to context and vice versa: On learning invariance for out-of-distribution generalization. In *ECCV*, 2022.
- Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *ESPC*, 2007.
- Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *ICLR*, 2020.
- Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *ICLR*, 2017.
- Seonguk Seo, Joon-Young Lee, and Bohyung Han. Unsupervised learning of debiased representations with pseudo-attributes. In *CVPR*, 2022.
- Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In *NeurIPS*, 2020.
- Robik Shrestha, Kushal Kafle, and Christopher Kanan. Occamnets: Mitigating dataset bias by favoring simpler hypotheses. In *ECCV*, 2022.

- Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. In *NeurIPS*, 2020.
- Rishabh Tiwari and Pradeep Shenoy. Overcoming simplicity bias in deep networks using a feature sieve. In *ICML*, 2023.
- Vladimir Vapnik. An overview of statistical learning theory. *IEEE TNN*, 1999.
- Haohan Wang, Zexue He, Zachary L. Lipton, and Eric P. Xing. Learning robust representations by projecting superficial statistics out. In *ICLR*, 2019.
- Zihan Wang and Arthur Jacot. Implicit bias of SGD in ℓ_2 -regularized linear DNNs: One-way jumps from high to low rank. In *ICLR*, 2024.
- Zeyu Wang, Klint Qinami, Ioannis Karakozis, Kyle Genova, Prem Nair, Kenji Hata, and Olga Russakovsky. Towards fairness in visual recognition: Effective strategies for bias mitigation. In *CVPR*, 2020.
- Wikipedia. Cosine similarity. https://en.wikipedia.org/wiki/Cosine_similarity#L2-normalized_Euclidean_distance, 2024.
- D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE TEC*, 1997.
- Dinghui Zhang, Kartik Ahuja, Yilun Xu, Yisen Wang, and Aaron Courville. Can subnetwork structure be the key to out-of-distribution generalization? In *ICML*, 2021.
- Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

7 Appendix

In the primary text of our submission, we introduce DeNetDM, a novel debiasing framework that leverages the variation of linear decodability across network depths to effectively disentangle bias from core attributes. This technique sets a new benchmark for bias mitigation, achieving unparalleled performance without reliance on data augmentations. To ensure our manuscript's integrity, we provide an extensive appendix designed to complement the main text. This includes a series of additional experiments, expanded ablation studies, comprehensive implementation protocols, and deeper analyses of our findings. The Appendix is presented to bridge the content gap necessitated by the page constraints of the main manuscript, providing a detailed exposition of our methodology and its broader impact on the domain.

7.1 Notations

- B : Bias attributes
- C : Core attributes
- X, Y : Sample set (X : Inputs, Y : Labels)
- X_a : Bias-aligned points
- X_b : Bias-conflicting points
- ϕ : Encoder
- $\phi[d]$: Encoder at depth d
- $\rho(\phi[d])$: Effective rank of an encoder at depth d
- $\pi_i(d)$: Propagation probability of an attribute (indexed i) at depth d
- Ω^d : Propagation probability distribution of an attribute set at layer d of a neural network
- $\text{rank}(\cdot)$: Rank of matrix
- $\text{dim}(\cdot)$: Dimensionality of a tensor / space
- ε : Knock-off probability when transitioning from depth d to $d + 1$

7.2 Proofs

Theorem 1 (Partition Rank): When the partitioning $X = X_a \cup X_c$ is stable wrt. C , the rank of the bias-aligned partition is upper-bounded by the rank of the bias-conflicting partition, *i.e.*,

$$\text{rank}(X_a) \leq \text{rank}(X_c)$$

Proof. The theorem assumes a stable partitioning of the sample set X , *i.e.*, in both the bias-aligned and conflicting subsets, the distribution of the core attributes are equal to that of the original sample set, *i.e.*,

$$P(X_a^C) = P(X_c^C) = P(X^C)$$

Under this condition, the only component in either of the subsets that determines the subset's rank should be the bias attributes, under the simplifying assumption (without loss of generality) that the attribute space is made up of only the core and the bias attributes.

For the bias aligned partition X_a , all the data points within a class have very low variance within the set of values for the bias attribute B , since it is spuriously correlated with the class label. So, B within a class collapses to a much lower dimensional manifold $b \subseteq B$, such that $\text{rank}(b) \leq \text{dim}(B)$. Extending this across classes, without loss of generality, assuming that the number of classes is higher than the variance in B among the bias aligned samples, *i.e.*, $\text{rank}(b)$, B over the set of all classes in X_a would map to a manifold of dimensionality $\text{rank}(b)$. Therefore, since the whole of B in X_a can be represented by a manifold of dimensionality $\text{rank}(b)$ orthogonal to the basis of C , the rank of X_a is given by:

$$\text{rank}(X_a) = \text{rank}(C) + \text{rank}(b)$$

For the bias conflicting partition, since there is no correlation between the class labels and B , within each class, the bias attributes would require a $\dim(B)$ dimensional subspace independent of C , to be represented, since $B \perp C$. This implies that the rank of the bias conflicting points would be:

$$\text{rank}(X_c) = \text{rank}(C) + \dim(B),$$

Since we know that $b \subseteq B$, which leads to $\text{rank}(b) \leq \dim(B)$, it is ultimately implied that $\text{rank}(X_a) \leq \text{rank}(X_c)$.

This completes the proof of the theorem. $\square$

Lemma 1. Let $\mathcal{A} = [A_0, A_1, \dots, A_n]$ be the attribute subspace of X with increasing ranks, i.e., $\text{rank}(A_0) < \text{rank}(A_1) < \dots < \text{rank}(A_n)$, such that every $A \in \mathcal{A}$ is maximally and equally informative of the label Y , i.e., $I(A_0, Y) = I(A_1, Y) = \dots = I(A_n, Y)$. Then, at any given depth d of a neural network, the probability of propagation $\pi_i(d)$ of an attribute A_i is directly proportional to the effective rank $\rho(\phi[d])$ of the network at that depth, i.e.,

$$\pi_i(d) = \mathcal{O}(\rho(\phi[d]))$$

Proof. Let the total rank of $\mathcal{A}$ be R . Consider some reference attribute $A \in \mathcal{A}$ with rank r . According to the results on the low rank simplicity bias (Huh et al., 2023; Wang and Jacot, 2024) and deep information propagation (Schoenholz et al., 2017), after propagation through each layer, εR of the bases would be knocked off, resulting in a pruned version of $\mathcal{A}$. The total number of ways in which $\mathcal{A}$ can be pruned is given by $\binom{R}{\varepsilon R}$. Also, the number of ways that A features in that pruning is given by $\binom{\varepsilon R}{r}$. Thus, the probability of A being knocked-off in layer-1 of ϕ is given by:

$$\binom{\varepsilon R}{r} / \binom{R}{\varepsilon R} = \frac{r!}{(\varepsilon R - r)! R! (1 - \varepsilon) R!}$$

Therefore, probability of survival at layer d :

$$\pi_i(d) = \left(1 - \frac{r!}{\underbrace{(\varepsilon R - r)!}_{a} \underbrace{R! (1 - \varepsilon) R!}_{b}} \right)^d \quad (9)$$

Therefore, the probability of survival $\pi_i(d)$ of any attribute A_i at depth d increases exponentially with increasing rank r of A_i , and decreases exponentially with the knock-off rate ε . a is the part of the knocked-off basis not in A_i . b is the part of the complete basis of $\mathcal{A}$ not affected by the first knock-off at layer 1. Thus, at depth d , b^d indicates the size of the subspace of $\mathcal{A}$ that survives at depth d , therefore being proportional to the effective rank of $\phi[d]$. Based on this, the effective rank at depth d can be written as:

$$\pi_i(d) \propto (1 - \varepsilon)^d R^d = \mathcal{O}((1 - \varepsilon)^d R^d) = \mathcal{O}(\rho(\phi[d])),$$

This completes the proof of the lemma. $\square$

Lemma 2. Let $\mathcal{A} = [A_0, A_1, \dots, A_n]$ be the attribute subspace of X with increasing ranks, i.e., $\text{rank}(A_0) < \text{rank}(A_1) < \dots < \text{rank}(A_n)$, such that every $A \in \mathcal{A}$ is maximally and equally informative of the label Y , i.e., $I(A_0, Y) = I(A_1, Y) = \dots = I(A_n, Y)$. Then, at any given layer d of a neural network, the propagation probability of an attribute decreases with rank, i.e.,

$$\pi_1(d) \geq \pi_2(d) \geq \dots \geq \pi_n(d),$$

at a rate that is polynomial in the attribute rank, with degree equal to the depth, i.e.,

$$\frac{\pi_{i+1}(d)}{\pi_i(d)} = \mathcal{O}(r^{-d})$$

Proof. Continuing from Equation (9), we have the propagation probability of A_i at depth d as:

$$\pi_i(d) = \left(1 - \frac{r!}{\underbrace{(\varepsilon R - r)!}_{a} \underbrace{R!}_{b} (1 - \varepsilon) R!} \right)^d$$

Note that when r increases, *i.e.*, for a higher rank attribute, it leads to a drop in a , and in a subsequent exponential decrease in $\pi_i(d)$ as follows:

$$\begin{aligned} \pi_{i+k}(d) &= \left(1 - \frac{(r+k)!}{(\varepsilon R - (r+k))! R! (1 - \varepsilon) R!} \right)^d \\ &\implies \frac{\pi_{i+k}(d)}{\pi_i(d)} \leq 1 \\ &\implies \pi_{i+k}(d) \leq \pi_i(d) \\ &\implies \pi_1(d) \geq \pi_2(d) \geq \dots \geq \pi_n(d), \end{aligned}$$

which proves the first part of the lemma.

Now, taking the ratio of the propagation probabilities of attributes with rank $(i+k)$ and i at depth d , we get:

$$\frac{\pi_{i+k}(d)}{\pi_i(d)} = \mathcal{O} \left(\frac{(r+k)^d}{r^d} \right) = \mathcal{O} \left(\left(\frac{r+k}{r} \right)^d \right) = \mathcal{O} \left(\left(1 + \frac{k}{r} \right)^d \right) = \mathcal{O}(k^d r^{-d})$$

For propagation on to the next layer, $k = 1$. We thus have:

$$\frac{\pi_{i+1}(d)}{\pi_i(d)} = \mathcal{O}(r^{-d})$$

This completes the proof of the lemma. $\square$

Theorem 2 (Depth-Rank Duality): Let $\mathcal{A} = [A_0, A_1, \dots, A_n]$ be the attribute subspace of X with increasing ranks, *i.e.*, $\text{rank}(A_0) < \text{rank}(A_1) < \dots < \text{rank}(A_n)$, such that every $A \in \mathcal{A}$ is maximally and equally informative of the label Y , *i.e.*, $I(A_0, Y) = I(A_1, Y) = \dots = I(A_n, Y)$. Then, across the depth of the encoder ϕ , SGD yields a parameterization that optimizes the following objective:

$$\underbrace{\min_{\phi, f} \mathcal{L}(f(\phi(X)), Y)}_{\text{ERM}} + \min_{\phi} \sum_d \left\| \phi[d](\tilde{X}) - \Omega^d \odot \mathcal{A} \right\|_2,$$

where $\mathcal{L}(\cdot, \cdot)$ is the empirical risk, $f(\cdot)$ is a classifier head, $\phi[i](\cdot)$ is the output of the encoder ϕ (optimized end-to-end) at depth d , $\|\cdot\|_2$ is the l^2 -norm, $\odot$ is the element-wise product, $\tilde{X}$ is the l_2 -normalized version of X , $\Omega^d = [\mathbb{1}_{\pi_1(d)}; \mathbb{1}_{\pi_2(d)}; \dots; \mathbb{1}_{\pi_n(d)}]$, $\mathbb{1}_{\pi}$ is a random binary function that outputs 1 with a probability π , and $\pi_i(d)$ is the propagation probability of A_i at depth d bounded as:

$$\pi_i(d) = \mathcal{O}(\rho(\phi[d]) r_i^{-d}),$$

where $\rho(\phi[d])$ is the effective rank of the $\phi[d]$ representation space, and $r_i = \text{rank}(A_i)$.

Proof. Since all $A \in \mathcal{A}$ are equally informative about the label Y , they all equally minimize $\mathcal{L}(\cdot, \cdot)$. Thus, the representations learned by ϕ are solely determined by the second term in the summation of Equation (1). This means that the SGD must employ a selection mechanism to choose from the $\mathcal{A}$ that optimally utilizes the available parameters in $\mathcal{A}$.

If $\phi[d]$ has sufficiently many parameters to accommodate all of $\mathcal{A}$, SGD should have no reason to discard any of them. However, a number of works that analyze the representational properties of DNNs have found that as we go deeper into a network, the effective number of dimensions available for encoding information, formally known as the effective rank and denoted as $\rho(\phi[d])$ (effective rank of ϕ at depth d), decreases (Huh et al., 2023; Wang and Jacot, 2024). This characteristic is also known

as the simplicity bias of DNNs. Given the simplicity bias, SGD must learn a parameterization for ϕ that optimally selects from $\mathcal{A}$ when the effective rank at a particular layer is lower than $\text{rank}(\mathcal{A})$. In order to stay at the minimum of $\mathcal{L}(\cdot, \cdot)$, ϕ must rely on the complete basis of at least one attribute, as only partially learning an attribute would cause deviation from the minimum. So every attribute that is retained for prediction, has to be retained fully. Given this condition, the optimum choice for SGD under constrained effective ranks is thus, to choose $A \in \mathcal{A}$ in increasing order of effective ranks. In other words, the A_0 has the highest likelihood of getting chosen, followed by A_1 , then A_2 , and so on.

Lemma 1 and Lemma 2 provide bounds for the quantification of the associated probabilities at a given depth, for an attribute of a given rank. Combining them, we get the propagation probability of A_i at depth d as:

$$\pi_i(d) = \mathcal{O}(\rho(\phi[d])r_i^{-d}),$$

where r_i is the rank of A_i . We denote the distribution of π for an attribute across a network as Ω_i . Without loss of generality, assuming the retention of all attributes at depth $d - 1$, we get the forward pass output at depth d as:

$$\phi[d](X) = \gamma(W_d \cdot \phi[d-1](X)),$$

where W_d is the weight matrix at layer d and gamma is a non-linearity. Under the most general setting where the elimination of attributes comes only with a decrease in the effective rank and not in the reduction in the dimensionality of the weight matrix, applying Lemmas 1 and 2 we obtain the survival probability of the basis corresponding to all $A \in \mathcal{A}$ in W as:

$$\begin{aligned} W^d &= [\mathbb{1}_{\pi_0(d)} W_0^d; \mathbb{1}_{\pi_1(d)} W_1^d; \dots; \mathbb{1}_{\pi_n(d)} W_n^d] \\ \implies W^d \cdot X' &= [\mathbb{1}_{\pi_0(d)} W_0^d A_0; \mathbb{1}_{\pi_1(d)} W_1^d A_1; \dots; \mathbb{1}_{\pi_n(d)} W_n^d A_n] \\ &= \Omega^d \odot \mathcal{A} \cdot W \end{aligned}$$

where $\mathbb{1}_\pi$ is a random binary function that outputs 1 with a probability π , $\Omega^d = [\mathbb{1}_{\pi_1(d)}; \mathbb{1}_{\pi_2(d)}; \dots; \mathbb{1}_{\pi_n(d)}]$, and $X' = \phi[d-1](X)$. To keep $\mathcal{L}$ at a minimum, W^d must correctly activate for the informative features in x' , for which it must maximize $\Omega^d \odot \mathcal{A} \cdot W^d$. Now, $\Omega^d \odot \mathcal{A} \cdot W^d$ is maximized when $W^d = \Omega^d \odot \mathcal{A}$. Thus, the optimal strategy for SGD is to parameterize W^d such that it captures the attributes in $\mathcal{A}$ according to the distribution Ω^d . Over the full depth, the optimization objective would then be:

$$\max_{\phi} \sum_d \Omega^d \odot \mathcal{A} \cdot \phi[d](X) \equiv \min_{\phi} \sum_d \left\| \phi[d](\tilde{X}) - \Omega^d \odot \mathcal{A} \right\|_2$$

where $\tilde{X}$ is the l_2 -normalized version of X , and the equivalence comes from the equivalence of maximizing the dot product and minimizing the l_2 -distance of the normalized samples (Wikipedia, 2024).

This completes the proof of the theorem. $\square$

Corollary 2.1. *Let $\mathcal{A} = [A_0, A_1, \dots, A_n]$ be the attribute subspace of X with increasing ranks, i.e., $\text{rank}(A_0) < \text{rank}(A_1) < \dots < \text{rank}(A_n)$, such that every $A \in \mathcal{A}$ is maximally and equally informative of the label Y , i.e., $I(A_0, Y) = I(A_1, Y) = \dots = I(A_n, Y)$. Then, across the depth of a randomly initialized encoder ϕ , the output of ϕ at depth d follows the propagation distribution Ω^d of the attribute space $\mathcal{A}$ as:*

$$\phi[d](\tilde{X}) \propto \Omega^d \odot \mathcal{A}, \quad (10)$$

where $\phi[i](\cdot)$ is the output of the encoder ϕ at depth d , $\odot$ is the element-wise product, $\tilde{X}$ is the l_2 -normalized version of X , $\Omega^d = [\mathbb{1}_{\pi_1(d)}; \mathbb{1}_{\pi_2(d)}; \dots; \mathbb{1}_{\pi_n(d)}]$, $\mathbb{1}_\pi$ is a random binary function that outputs 1 with a probability π , and $\pi_i(d)$ is the probability of propagation of A_i of rank r_i at depth d bounded as:

$$\pi_i(d) = \mathcal{O}(\rho(\phi[d])r_i^{-d}), \quad (11)$$

Discussion: Let $\mathbb{L}$ be the space of all empirical risks $\{\mathcal{L}_1, \mathcal{L}_2, \dots\}$ over X . According to the No Free Lunch theorem (Wolpert and Macready, 1997), if an attribute minimizes some $\mathcal{L}_i \in \mathbb{L}$, there exists another $\mathcal{L}_j \in \mathbb{L}$ which it maximizes. So, if we consider the probability of survival of attributes in a randomly initialized network, we need to marginalize the ERM part of Equation (1) across the entirety of $\mathbb{L}$. Assuming an unbiased random initialization scheme, the distribution associated with

$\mathbb{L}$ would be uniform (because no concrete form of empirical risk is defined, we can consider all functions $\mathcal{L} \in \mathbb{L}$ to be equally likely, under the unbiased initialization assumption) as follows:

$$\int_{\mathcal{L} \in \mathbb{L}} \mathcal{L}(f(\phi(X))) \Pr(\mathcal{L}) d\mathcal{L},$$

where $\Pr(\mathcal{L})$ is the probability associated with the function $\mathcal{L} \in \mathbb{L}$, which can be assumed to be uniform, as argued before. Then, due to the No Free Lunch Theorem (Wolpert and Macready, 1997), the expected informativeness of all attributes in X is the same, satisfying the $I(A_0, Y) = I(A_1, Y) = \dots = I(A_n, Y)$ criterion in the theorem, where the nature of Y is determined by the specific choice of $\mathcal{L}$. The remainder of the reasoning for $\phi[d](\tilde{X}) \propto \Omega^d \odot \mathcal{A}$ is the same as the proof for $\min_{\phi} \sum_d \left\| \phi[d](\tilde{X}) - \Omega^d \odot \mathcal{A} \right\|_2$ in Theorem 2.

7.3 Equivalence with Product of Experts Framework

In Section 3.2 of the main text, we asserted that our training methodology is derived from the Product of Experts. In this section, we elucidate this mathematically:

$$f : \mathbb{R}^F \xrightarrow{\text{linear}} \mathbb{R}^c, \quad \tilde{f}(x) = \text{softmax}(f(x))$$

$$\phi_b : \mathbb{R}^{C \times H \times W} \longrightarrow \mathbb{R}^F, \quad \text{where } F \text{ is the feature dimension}$$

$$\phi_d : \mathbb{R}^{C \times H \times W} \longrightarrow \mathbb{R}^F, \quad \text{such that } \text{depth}(\phi_b) > \text{depth}(\phi_d)$$

$$L(x, y; \phi_b, \phi_d) = - \sum_{c=1}^C y_c \log(\hat{p}_{\phi_b, \phi_d}^c) \quad (\text{Loss function definition})$$

$$\begin{aligned} \hat{p}_{\phi_b, \phi_d} &= \frac{\tilde{f}_c(\phi_b(x)) \cdot \tilde{f}_c(\phi_d(x))}{\sum_{c=1}^C \tilde{f}_c(\phi_b(x)) \cdot \tilde{f}_c(\phi_d(x))} \quad (\text{Product of Experts}) \\ &= \text{softmax}_c(\log(\tilde{f}(\phi_b(x))) + \log(\tilde{f}(\phi_d(x)))) \quad (\text{Softmax log-sum-exp trick}) \\ &= \text{softmax}_c(f(\phi_b(x)) + f(\phi_d(x))) \quad (\text{Translation invariance of softmax}) \\ &= \text{softmax}_c(f(\phi_b(x) + \phi_d(x))) \quad (\text{Linearity of classifier } f) \end{aligned}$$

We utilize $\hat{p}_{\phi_b, \phi_d}$ to compute the probabilities in DeNetDM which is the same as Equation 2 presented in the main paper.

7.4 Pseudocode

The pseudocode for the entire training process of DeNetDM is provided in Algorithm 1.

7.5 Feature Decodability

We utilize feature decodability to gauge the extent to which specific dataset features can be reliably decoded across models of varying depths. Hermann and Lampinen (2020) demonstrated that the visual features can be decoded from the higher layers of untrained models. Additionally, they observed that the feature decodability from an untrained model has a significant impact in determining which features are emphasized and suppressed during the model training. Following their approach, we specifically focus on assessing the decodability of bias and core attributes from the penultimate layer of untrained models. In order to evaluate the decodability of an attribute in a dataset, we train a decoder to map the activations from the penultimate layer of a frozen, untrained model to attribute labels. The decoder comprises a single linear layer followed by a softmax activation function. The

Algorithm 1 DeNetDM: Training

Input: Data: $\{(x, y)_i\}_{i=1}^N$
Output: ϕ_t, f_t
Initialize: $\phi_t, f_t, \phi_b, \phi_d$ such that $\text{depth}(\phi_b) > \text{depth}(\phi_d)$

- 1: **repeat**
- 2: Fetch minibatch data $\{(x, y)_i\}_{i=1}^K$
- 3: **for** $i = 1$ to K (in parallel) **do**
- 4: Compute $\hat{p}$ using (4) to obtain $(\hat{p}, y)_i$
- 5: **end for**
- 6: Update ϕ_b, ϕ_d, f by minimizing $\mathcal{L}_{CE}$ in (3) via SGD
- 7: **until** Convergence ▷ stage1
- 8: **repeat**
- 9: Fetch minibatch data $\{(x, y)_i\}_{i=1}^K$
- 10: **for** $i = 1$ to K (in parallel) **do**
- 11: Compute $\hat{p}, \hat{p}_s, \hat{p}_t$ via (5), (7), (8) respectively
- 12: **end for**
- 13: Update ϕ_t, f_t by minimizing $\mathcal{L}$ in (6) via SGD
- 14: **until** Convergence ▷ stage2

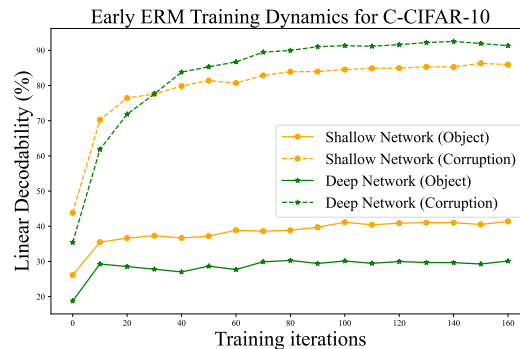


Figure 4: Early training dynamics of DeNetDM on C-CIFAR10 dataset.

decoder is trained using an unbiased validation set associated with the dataset, where each instance is labeled according to the attribute under consideration. Subsequently, the linear decodability of the attribute, measured in accuracy, is reported on the unbiased test set. We investigate the decodability of digit and color attributes in the CMNIST dataset from MLP models with varying depths, including 3, 4, and 5 layers, and the results are depicted in Figure 2a. To investigate how feature decodability evolves during the early stages of Empirical Risk Minimization (ERM) training across networks with varying depths, we train 3-layer and 5-layer MLPs on the CMNIST dataset. Following the training, we evaluate the model's linear decodability for digit and color attributes.

7.6 Additional Experiments

7.6.1 Feature Decodability on C-CIFAR10

Analogous to Figure 2 in the main paper, Figure 4 illustrates the variation in feature decodability for corruption (bias) and object (core) in the C-CIFAR10 dataset as ERM training advances. We chose ResNet 20 as the deep network and a 3-layer CNN as the shallow network since these are the architectures used for DeNetDM. The training dynamics show a similar trend to those observed in ColoredMNIST concerning bias and core attributes. As training progresses, corruption (bias) becomes highly decodable by both deep and shallow networks, with the deep branch slightly outperforming the shallow branch. However, the object attribute (core) is more decodable by the shallow network as training progresses, during the initial training dynamics. These observations align with the early training dynamics observed in CMNIST.

7.7 Generalization to other tasks

We evaluate the performance of DeNetDM on the CivilComments dataset Koh et al. (2021), which involves natural language debiasing. The task requires classifying online comments as toxic or non-toxic, with labels spuriously correlated with mentions of certain demographic identities. As shown in Table 4, our approach performs comparably to state-of-the-art methods. Due to the constrained rebuttal timeline, we just applied our model out of the box, without any reasonable hyperparameter tuning. The observations illustrate the applicability of DeNetDM to domains beyond vision.

Table 4: Worst group accuracy (%) comparison between different methods on CivilComments dataset.

Method	Worst Group Acc (%)
ERM	58.6 (1.7)
JTT	69.3 (-)
LfF	58.3 (0.5)
LC	70.3 (1.2)
DeNetDM (ours)	68.33 (-)

7.7.1 Effect of depth modulation

To validate our hypothesis on the significance of network depth in DeNetDM, we conduct an ablation by setting the same depth for both branches and compare it with the default DeNetDM, where one branch is deeper than the other. We focus on the first stage of DeNetDM training for 5 different random seeds, reporting the averaged test accuracy on bias-aligned and bias-conflicting points for individual branches in Table 5. Branch 1 and Branch 2 in Table 5 correspond to the deep and shallow branches in DeNetDM, respectively. We ignore the second stage of training since our focus was primarily on the segregation of bias and core attributes. An interesting observation is the significant standard deviation in accuracies when the branches have the same depth, observed in both datasets. This phenomenon occurs because in such a configuration, DeNetDM loses its ability to clearly distinguish between branches. This is due to the similarity in feature decodability of bias and core attributes across both branches, as discussed in Section 3.2. As a result, DeNetDM may distribute information across multiple branches or still separate core and bias attributes, but the specific branch capturing core attributes varies with different initialization. In contrast, when depths are unequal, the deeper branch tends to focus on aligned points, disregarding conflicting ones, as seen in the test accuracies provided in Table 5. Additionally, the shallow branch emphasizes capturing core attributes, consistently enhancing conflicting accuracy. This shows the pivotal role of depth modulation in the DeNetDM framework for effectively segregating bias and core attributes.

Table 5: Performance of DeNetDM using different network depths for the two branches of DeNetDM.

Dataset	Depth (Branch 1, Branch 2)	Branch	Conflicting Accuracy (%)	Aligned Accuracy (%)
CMNIST	(5, 5)	Branch 1	44.94 (22.25)	74.85 (12.71)
		Branch 2	17.25 (7.89)	88.57 (9.50)
	(5, 3)	Branch 1	1.921 (0.29)	99.92 (0.25)
		Branch 2	83.17 (0.96)	88.25 (2.254)
C-CIFAR10	(ResNet-20, ResNet-20)	Branch 1	19.54 (11.16)	85.83 (8.19)
		Branch 2	24.42 (16.93)	86.95 (11.04)
	(ResNet-20, 3-layer CNN)	Branch 1	3.0 (1.29)	99.34 (0.47)
		Branch 2	38.52 (0.99)	76.72 (2.19)

7.7.2 Effect of loss components on CMNIST

The primary text, constrained by spatial limitations, only includes an ablation study detailing the effect of individual loss components of DeNetDM on the C-CIFAR10 dataset. However, this section extends the scope of our analysis to encompass the CMNIST dataset and the results are summarized

in Table 6. The proposed approach exhibits a similar trend as observed in the case of C-CIFAR10 (presented in Section 4.4).

Table 6: Ablation study of different losses used in DeNetDM on CMNIST dataset.

$\mathcal{L}_{\text{CE}}$ (Stage-1)	$\mathcal{L}_{\text{dist}}$ (Stage-2)	$\mathcal{L}_t$ (Stage-2)	Accuracy (%)	Conflicting Accuracy (%)	Aligned Accuracy (%)
✓	-	-	81.61	83.28	89.66
✓	-	✓	82.96	81.53	95.85
✓	✓	-	84.05	83.41	89.86
✓	✓	✓	84.97	84.44	89.17

7.7.3 Depth vs. Number of parameters

DeNetDM employs depth modulation as its principal strategy for mitigating bias. We investigate the influence of the number of parameters of both branches on DeNetDM performance. We opt for the optimal configuration of the proposed approach on C-CIFAR10 and conducted an ablation study, employing ResNet-20 ($\text{depth}(\phi_b) = 20$) as the deep network and a 3-layer CNN ($\text{depth}(\phi_d) = 3$) as the shallow network. We explore three scenarios where $|\phi_b| < |\phi_d|$, $|\phi_b| \approx |\phi_d|$, and $|\phi_b| > |\phi_d|$. The first stage of DeNetDM training is then performed to analyze learning in the deep and shallow models in each of the cases, and the results are presented in Table 7. As indicated in Table 7, the shallow model exhibits increased resilience to spurious correlations, while the deep model captures bias in all three cases. This suggests that DeNetDM effectively segregates bias and core attributes regardless of the number of parameters in both branches. Interestingly, a notable finding is that the shallow model exhibits better robustness against correlations when the shallow branch possesses a greater number of parameters compared to the deep model, as evident from Table 7.

The findings for CMNIST mirror those observed for C-CIFAR10 as presented in Table 8: the shallow branch demonstrates robustness to spurious correlations, whereas the deep branch consistently assimilates bias irrespective of the number of parameters in both branches. These consistent patterns across datasets reinforce the efficacy of DeNetDM in distinguishing between bias and core attributes.

Table 7: Ablation study on the number of parameters of deep and shallow branches in DeNetDM using C-CIFAR10 dataset.

Case	Branch	Conflict (%)	Align (%)
$\phi_b > \phi_d$	ϕ_b	3.08	96.8
	ϕ_d	29.78	62.61
$\phi_b \approx \phi_d$	ϕ_b	3.48	95.91
	ϕ_d	28.64	64.32
$\phi_b < \phi_d$	ϕ_b	2.04	99.01
	ϕ_d	39.05	67.68

Table 8: Ablation study on the number of parameters of deep and shallow branches in DeNetDM using CMNIST dataset.

Case	Branch	Conflicting Accuracy (%)	Aligned Accuracy (%)
$\phi_b < \phi_d$	ϕ_b	11.90	99.93
	ϕ_d	83.89	88.78
$\phi_b \approx \phi_d$	ϕ_b	11.87	99.90
	ϕ_d	83.07	89.09
$\phi_b > \phi_d$	ϕ_b	10.79	98.26
	ϕ_d	83.32	88.61

7.7.4 Effect of Network Depth on DeNetDM

In the main text, we have illustrated how the variation in network depth affects the performance of DeNetDM. We provide an in-depth analysis in this section. As observed in the first three rows of Table 9, as the difference in network depth of deep and shallow progressively increases, the performance of the debiased model increases monotonically. Further, when we decrease the difference in depth of shallow and deep branches (rows 3 and 4) the performance decreases to 80.42% compared to 87.37%. Similar performance degradation can be seen when we increase the depth of the shallow network from 4 to 6 (rows 5 and 6). Hence, DeNetDM is able to distinguish bias and core attributes better when there is a significant difference between the depths of shallow and deep branches. This aligns with the observations presented in Section 3.2 of the main text (Effect of depth modulation).

Table 9: Performance comparison of DeNetDM for various depths of shallow and deep branches.

Depth (Shallow, Deep)	Conflicting Accuracy (%)	Aligned Accuracy(%)
(3, 4)	72.2	98.33
(3, 5)	80.46	92.87
(3, 7)	87.37	93.62
(6, 7)	80.42	96.45
(4, 8)	91.19	94.62
(6, 8)	69.55	93.83

7.7.5 Performance on varying bias-conflicting ratios

We perform experiments on the CMNIST dataset with bias-conflicting ratios of 10% and 20% to evaluate our method’s efficacy across a broader range of ratios. The findings, presented in Table 10, show that DeNetDM performs as expected, effectively capturing core attributes in the shallow branch for varied bias ratios.

7.7.6 Early training dynamics in ResNet architectures

We also examine the early training dynamics of ResNet-8, ResNet-32, and ResNet-50, akin to Figure 2b in C-CIFAR10 dataset to assess the scalability of DeNetDM to larger ResNet models. After 200 iterations, texture (bias) decodability in all architectures neared 99%, while core attribute decodability for ResNet-8, ResNet-32, and ResNet-50 was 18.74%, 24.32%, and 12.91%, respectively. This aligns with our hypothesis that ResNet-50 would prefer texture attribute over core when paired with ResNet-8 or ResNet-32. To confirm, we tested two setups: (1) ResNet-8 and ResNet-50, and (2) ResNet-32 and ResNet-50. The results, shown in Table 11, indicate high bias-aligned accuracy for ResNet-50 and high bias-conflicting accuracy for ResNet-8 and ResNet-32 respectively. Since ResNet-50 has lower core attribute decodability than ResNet-8 and ResNet-32, it favors bias attributes, while the shallow branches capture core attributes. This experimental results suggest DeNetDM’s applicability to diverse, complex and larger models / architectures.

7.8 Additional details

In this section, we provide an in-depth discussion of various datasets used along with finer implementation details that enhance the reproducibility of our method.

Table 10: Results on CMNIST with wider bias conflicting ratios.

Bias ratio	Branch	Conflicting Accuracy (%)	Aligned Accuracy (%)
10%	Deep	1.84(0.5)	99.14(0.2)
	Shallow	93.12(0.8)	96.47(1.3)
20%	Deep	3.23(2.8)	97.93(2.1)
	Shallow	94.49(2.4)	97.51(3.4)

Table 11: Comparison of the performance of DeNetDM using different network depths for the two branches of DeNetDM.

Depth (Branch 1, Branch 2)	Branch	Conflicting Accuracy (%)	Aligned Accuracy (%)
(ResNet-50, ResNet-32)	Branch 1	3.48 (0.98)	97.15 (2.10)
	Branch 2	30.88 (1.22)	81.72 (0.73)
(ResNet-50, ResNet-8)	Branch 1	9.38 (1.52)	98.60 (0.86)
	Branch 2	20.32 (1.90)	59.94 (2.61)

7.8.1 Datasets

We provide a detailed description of various datasets used along with a representative sample of all of them.

- **Colored MNIST(CMNIST)**: CMNIST is an adaptation of the MNIST, that introduces color variation to the images. For each digit class, the majority $(1 - \alpha)$ of the images are correlated with the corresponding color c_i , with i matching the digit label y . The remaining images are randomly assigned one of the other colors c_j , where $j \neq y$. The challenge of this dataset lies in identifying the digits despite the strong color bias. To incorporate color variability, a noise vector v drawn from a normal distribution is added to c_i . The dataset and its characteristics are illustrated in Figure 5. Among multiple choices of severity, we choose the most severe corruption to simulate the worst-case scenario as done in other works.
- **Corrupted CIFAR10 (C-CIFAR10)**: The Corrupted CIFAR dataset represents an evolved form of the classic CIFAR set, with an emphasis on two particular features: the object depicted and the type of corruption applied. In an approach akin to that used for CMNIST, this dataset adopts an array of corruption styles, labeled from c_0 , symbolizing blurring, to c_9 , indicative of snow. Within each object category, a proportion $1 - \alpha$ of the images is intentionally altered with the corruption type c_i , corresponding to the object's label y . The remainder of the images is processed with a randomly selected corruption type c_j , chosen to ensure $j \neq y$. In our dataset, we employ the highest degree of corruption out of the five levels outlined in the original CMNIST dataset. Illustrative samples from this dataset are demonstrated in Figure 5.
- **Biased FFHQ (BFFHQ)**² The BFFHQ dataset is a selectively reduced subset derived from the larger FFHQ database of facial images, with a focus on the attributes of gender and age. Gender is designated as the primary attribute of analysis, with age being the secondary attribute that could introduce bias. The gender classification is binary, encompassing male and female categories. The dataset predominantly features male images of subjects aged between 40 and 59, whereas female images are generally of subjects aged between 10 and 29. Samples that defy these age associations—such as younger male or older female subjects—are also present, countering the main age distribution.
- **Biased Action Recognition (BAR)** : The Biased Action Recognition (BAR) dataset comprises real-world images classified into six action categories, each biased towards specific locations. The chosen pairs encompass six common action-location combinations: Climbing on a Rock Wall, Diving underwater, Fishing on a Water Surface, Racing on a Paved Track, Throwing on a Playing Field, and Vaulting into the Sky. The testing set consists solely of samples that present conflicts in bias. Consequently, achieving higher accuracy results on this set indicates superior debiasing performance.

7.8.2 Baselines

In this section, we provide a detailed overview of the baselines:

- **Empirical Risk Minimization (ERM)** Vapnik (1999): Standard ERM using cross-entropy loss.
- **Group DRO (GDRO)** Sagawa et al. (2020): A supervised approach that utilizes group labels to identify the worst group and learn an unbiased classifier.

²<https://github.com/kakaoenterprise/Learning-Debiased-Disentangled>

- **Learning from Failure (Lff)** Nam et al. (2020): Identifies bias-conflicting points through the Generalized Cross Entropy (GCE) loss and upweighting for debiasing.
- **Just Train Twice (JTT)** Liu et al. (2021): Treats misclassified points by ERM-based classifiers as bias-conflicting and upweights them for debiasing.
- **Disentangled Feature Augmentation (DFA)** Lee et al. (2021): Introducing feature augmentation to improve the diversity of bias-conflicting points and enhance unbiased accuracy.
- **Logit Correction (LC)** Liu et al. (2023): Proposes logit correction for bias mitigation along with MixUp Zhang et al. (2018) inspired data augmentation for increasing diversity.



Figure 5: Samples from training data of CMNIST, Corrupted-CIFAR10 and Biased FFHQ.

Table 12: Optimal hyperparameters for the CMNIST, C-CIFAR10, BAR and BFFHQ datasets determined through extensive experimentation. The tuples represent optimal hyperparameters for Stage 1 and Stage 2, respectively.

Parameter	CMNIST	C-CIFAR10, BAR	BFFHQ
Learning Rate (LR)	$(1.0 \times 10^{-3}, 1.0 \times 10^{-3})$	$(1.0 \times 10^{-3}, 1.0 \times 10^{-4})$	$(1.0 \times 10^{-3}, 1.0 \times 10^{-4})$
Batch Size	(64, 64)	(256, 256)	(64, 64)
Momentum	0.9	0.9	0.9
Weight Decay	$(1.0 \times 10^{-3}, 0)$	$(1.0 \times 10^{-3}, 0)$	(0, 0)
Epochs	(100, 100)	(100, 200)	(10, 100)

7.8.3 Implementation details

In this section, we detail the optimal hyperparameters identified for various datasets, which were instrumental in achieving the results reported in the main manuscript. The optimal hyperparameters obtained for various datasets are listed in Table 12. Additional parameters not mentioned in Table 12 follow the default values of PyTorch.

Data Augmentations: The training phase of DeNetDM incorporated specific data augmentation techniques tailored to each dataset. For instance, the CMNIST dataset did not utilize any form of augmentation. In contrast, the C-CIFAR10 and BFFHQ datasets applied Random Horizontal Flip and random cropping, with the latter involving crops from images padded by 4 pixels. These augmentations are critical as they introduce variability into the dataset, aiding the generalization ability of the neural network.

Experimental compute: We utilize RTX 3090 GPUs for all our experiments.

Architectural Details: Depth modulation is a critical component of our debiasing strategy. We enumerate the architecture specifics of the shallow branches tailored for each dataset below.

CMNIST:

```
(shallow branch): Sequential(
  (c1): Linear(in_features=2352, out_features=100, bias=True)
```

```

(r1): ReLU()
(s1): MLPHiddenlayers(
  (hidden_layers): ModuleList(
    (0): Linear(in_features=100, out_features=100, bias=True)
  )
  (act): ReLU()
)
)

```

C-CIFAR10 and BAR:

```

(shallow branch): Sequential(
  (c1): Conv2d(3, 32, kernel_size=(5, 5), stride=(1, 1))
  (b1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
  (r1): ReLU()
  (s1): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1,
    ceil_mode=False)
  (c2): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
  (b2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
  (r2): ReLU()
  (s2): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1,
    ceil_mode=False)
  (c3): Conv2d(64, 64, kernel_size=(5, 5), stride=(1, 1))
  (b3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
  (r3): ReLU()
  (f1): Flatten(start_dim=1, end_dim=-1)
)
(classifier): Linear(in_features=64, out_features=10, bias=True)
(act): ReLU()

```

BFFHQ:

```

(shallow branch): Sequential(
  (c1): Conv2d(3, 64, kernel_size=(7, 7), stride=(1, 1))
  (b1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
  (r1): ReLU(inplace=True)
  (s1): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1,
    ceil_mode=False)
  (c2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
  (b2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
  (r2): ReLU(inplace=True)
  (s2): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1,
    ceil_mode=False)
  (c3): Conv2d(128, 512, kernel_size=(3, 3), stride=(1, 1))
  (s3): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1,
    ceil_mode=False)
  (b3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
  (r3): ReLU(inplace=True)
  (c4): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1))
  (b4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
  (r4): ReLU(inplace=True)
  (a1): AdaptiveAvgPool2d(output_size=(1, 1))
  (f1): Flatten(start_dim=1, end_dim=-1)
)

```

7.9 Limitations & Broader Impact

The primary challenge with this approach is the scalability issue when applied to a multi-bias setting. As the number of bias attributes increases, the subtle variations in linear decodability across the various branches could become so refined that accurately identifying biases may fail to achieve high fidelity. Moreover, depending on the network architecture might compel the model to depend excessively on intricate hyperparameter adjustments.

The societal impacts of identifying and mitigating biases in neural networks are extensive, resulting in fairer, more equitable, and trustworthy AI systems. Some of them are as follows :

1. Bias Mitigation in AI : contributes to more equitable AI systems by reducing the influence of spurious correlations.
2. Societal Benefits: contributes to societal fairness by reducing biased decision-making in AI systems and potentially decreases the risk of discrimination in AI applications.
3. Ethical AI Development: encourages transparency and accountability in AI research and deployment.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Refer to Section 1.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Refer to Section 7.9.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: All proofs are provided in Section 7.2.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All the experimental details required for reproducibility are covered in the Sections 4.1 and 7.8.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: Source code is provided in <https://github.com/kadarsh22/DeNetDM>.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Refer to Sections 4.1 and 7.8 .

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: All the experiments are conducted for 5 different random seeds and we report the mean and standard deviation.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: Please refer to Section 7.8.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [\[Yes\]](#)

Justification: The proposed approach conforms with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: Please refer to Section 7.9

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper uses existing public datasets and models and are properly credited wher ever required throughout the paper.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.