Towards Understanding and Mitigating Dimensional Collapse in Heterogeneous Federated Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Federated learning aims to train models collaboratively across different clients without sharing data. One major challenge for this learning paradigm is the *data heterogeneity* problem, which refers to the potential discrepancy between the distributions of local data among various clients (in particular, we focus on heterogeneity of label distributions). To tackle this problem, we first study how data heterogeneity affects representations of locally trained models. Interestingly, we find that greater data heterogeneity results in local models suffering from more severe *dimensional collapse*, in which representations tend to reside in a lower-dimensional space instead of the ambient space. We theoretically analyze the gradient descent dynamics to shed light on how and why data heterogeneity results in dimensional collapse for local models. In addition, to remedy this problem caused by the data heterogeneity, we propose FEDDECORR, a novel method that can effectively mitigate dimensional collapse in federated learning. Specifically, FEDDECORR applies a regularization term during local training that encourages the different dimensions of representations to be uncorrelated. Our method consistently improves over baseline methods on standard benchmark datasets (e.g., CIFAR10/CIFAR100/TinyImageNet), while being implementation-friendly and computation efficient. Code will be released.

1 Introduction

With the rapid development deep learning, concerns regarding data privacy have been attracting increasingly more attention from industry and academia. To address such concern, [23] propose *Federated Learning*—a decentralized training paradigm enabling collaborative training across different clients without sharing data.

One major challenge in federated learning is the potential discrepancies in the distributions of local training data among clients, which is known as *data heterogeneity* (see Fig. 1 (a) for example). Such discrepancies can result in drastic disagreements between local optimum of each client and the desired global optimum, which leads to severe performance degradation of the global model. Previous works attempting to tackle this challenge mainly focus on the model parameters, either during local training [20, 16, 1] or global aggregation [29, 11]. However, these methods usually incur excessive computation burden or communication costs [17] due to the fact that deep neural networks are heavily over-parameterized.

In contrast, in this work, we focus on the representation[1] space of the locally trained model of each client and study the impact of data heterogeneity. We first train local models under various degrees of data heterogeneity and then visualize how their output representations differ. Specifically, for each of the locally trained models, we estimate the covariance matrices of their output representations

[1] In this work, "representation" is defined as output of the penultimate layer of neural networks.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

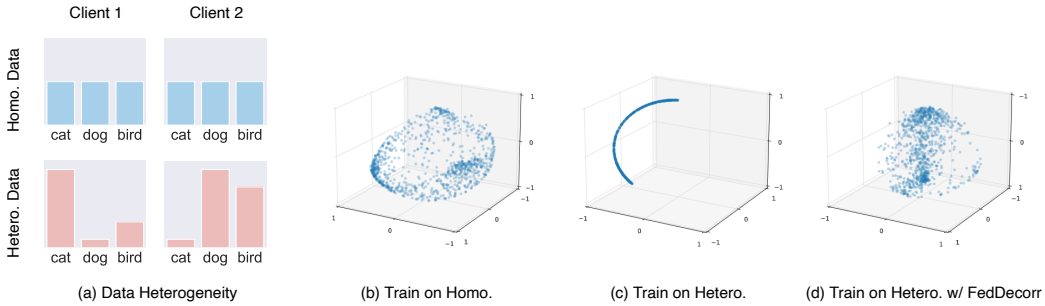

Figure 1: **(a)** illustrates data heterogeneity in terms of number of samples per class. **(b)**, **(c)**, **(d)** show representations (normalized to the unit sphere) of a locally trained model under homogeneous data, heterogeneous data, and heterogeneous data with FEDDECORR, respectively. Only **(c)** suffers dimensional collapse. **(b)**, **(c)**, **(d)** are produced with ResNet20 on CIFAR10. Best viewed in color.

and then analyze their singular values. Interestingly, we find that as the degree of data heterogeneity increases, more singular values tend to evolve towards zero. This observation suggests that stronger data heterogeneity causes representations of the locally trained models to suffer from more severe *dimensional collapse*, whereby representations are biased towards residing in a lower-dimensional space. For graphical illustration on difference between representations of locally trained models under homogeneous data and heterogeneous data, please see Fig. 1 (b-c).

To develop a theoretical understanding of why stronger data heterogeneity exacerbates dimensional collapse, we analyze the gradient descent dynamics for local training. In particular, we show that under stronger data heterogeneity, the weight matrices of local models would evolve to bias to lower rank solution under the gradient descent optimization. Eventually, such impact on the weight matrices would lead to greater dimensional collapse for output representations.

Our observations and analyses suggest that dimensional collapse of local models might be one of the reasons why federated learning methods struggle under data heterogeneity. Dimensional collapse is essentially a form of oversimplification in terms of the model, as the representation space is not fully utilized to discriminate data of different classes. Furthermore, the fact that stronger data heterogeneity results in a higher degree of dimensional collapse indicates that such oversimplification might be the chief culprit that results in the poor performance of federated learning. Therefore, we conjecture that by mitigating dimensional collapse under data heterogeneity, we can prevent local models from being optimized towards over-simplified solutions, thus improving federated learning.

To this end, we propose a novel federated learning method termed FEDDECORR. Specifically, FEDDECORR adds a regularization term during local training to encourage the Frobenius norm of the correlation matrix of representations to be small. We show theoretically and empirically that this proposed regularization term can effectively mitigate dimensional collapse (see Fig. 1 (d) for example). Next, through extensive experiments on standard benchmark datasets including CIFAR10, CIFAR100, and TinyImageNet, we show that FEDDECORR can consistently improve over baseline methods. In addition, we find that FEDDECORR yields more improvements as the federated learning setup becomes more challenging. For example, FEDDECORR's improvements over the baseline methods are more pronounced when there is stronger heterogeneity or more clients. Furthermore, FEDDECORR is implementation-friendly and yields extremely low computation overhead.

Our contributions are summarized as follows: First, we discover through empirical experiments that stronger data heterogeneity in federated learning leads to greater dimensional collapse for local models. Second, we develop a theoretical understanding of the dynamics behind our empirical discovery about data heterogeneity and dimensional collapse. Third, based on the motivation of mitigating dimensional collapse, we propose a novel method called FEDDECORR, which yields consistent improvements while being implementation-friendly and computation efficient.

2 Related Works

Federated Learning. McMahan *et al.* [23] proposed FedAvg, which adopts a simple averaging scheme to aggregate local models into the global model. However, under data heterogeneity, FedAvg suffers from unstable and slow convergence, resulting in performance degradation. To tackle this

challenge, previous works either improve local training [18, 20, 16, 1, 2, 30] or global aggregation [29, 11, 22, 28, 21]. Most of these methods focus on the model parameter space, which may result in high computation cost or communication cost due to deep neural networks being over-parameterized. Li *et al.* [18] focuses on model representations and uses a contrastive loss to maximize agreements between representations of local models and the global model. However, one drawback of [18] is that it requires additional forward passes during training, which almost doubles the training cost. In this work, based on our study on how data heterogeneity affects local model representations, we propose an effective yet highly efficient method to handle heterogeneous data. Another research trend is in personalized federated learning [3, 19, 9, 27, 10, 14, 34], which aims to train personalized local models for each client. In this work, however, we focus on the typical setting that aims to train one global model for all clients.

Dimensional Collapse. Dimensional collapse of representations has been studied in metric learning [24], self-supervised learning (SSL) [15], and class incremental learning (CIL) [26]. In this work, we focus on federated learning and discover that stronger data heterogeneity causes a higher degree of dimensional collapse for locally trained models. To the best of our knowledge, this work is the first to discover and analyze dimensional collapse of representations in federated learning.

Gradient Descent Dynamics. Arora *et al.* [4, 5] introduce the gradient descent dynamics framework to analyze the dynamics of multi-layer linear neural networks under the ℓ_2-loss and find deeper neural networks biasing towards low rank solution during optimization. Following their works, Jing *et al.* [15] finds two factors that cause dimensional collapse in SSL, namely strong data augmentation and implicit regularization from depth. In contrast to their works, we focus on federated learning with the cross-entropy loss. More importantly, our analysis focuses on dimensional collapse caused by data heterogeneity in federated learning instead of the depth of neural networks as in previous works.

Feature Decorrelation. Previously, feature decorrelation is used for different purposes, such as alleviating mode collapse in SSL [6, 33, 12], boosting generalization [7, 13, 31] and improving CIL at initial phase [26]. Differently, we apply feature decorrelation to counter the undesired dimensional collapse incurred by data heterogeneity in federated learning.

3 Dimensional Collapse Caused by Heterogeneous Data

In this section, we first empirically compare the output representations of local models trained under different degrees of heterogeneity. Next, to provide a theoretical understanding for our observations, we analyze the gradient descent dynamics and demonstrate how data heterogeneity causes dimensional collapse.

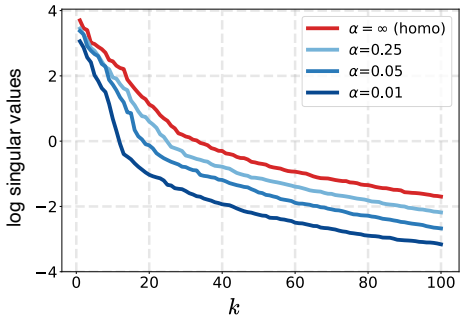

Figure 2: **Heterogeneous local training data cause dimensional collapse.** Given the four models trained under different degree of heterogeneity, we plot singular values of covariance matrix of representations in descending order. Representations are computed over CIFAR100 test set. The x-axis (k) is the index of singular values and the y-axis is the logarithm of singular values.

3.1 Empirical Observations

In this section, we empirically demonstrate that stronger data heterogeneity causes more severe dimensional collapse for local models. Specifically, we first separate the training examples of CIFAR100 into 10 splits, each corresponding to the local training data of a client. To simulate data heterogeneity among clients as in previous works [32, 28, 18], we sample a probability vector $\mathbf{p}_c \sim \mathrm{Dir}_K(\alpha)$ and allocate a $\mathbf{p}_{c,k}$ proportion of instances of class $c \in [C]$ to client $k \in [K]$, where $\mathrm{Dir}_K(\alpha)$ is the Dirichlet distribution with K categories and α is the (uniform) concentration parameter. In this partition scheme, a smaller α implies stronger data heterogeneity ($\alpha = \infty$ corresponds to homogeneous setting). Here, we set α to be 0.01, 0.05, 0.25 and ∞.

Then, for each of the settings generated by different α's, we train MobileNetV2 [25] on local data of one client (observations on other clients are similar and are provided in Appendix D). Next, for each of the four trained models, we compute the covariance matrix of the representations over the

test data of CIFAR100. Namely, we compute $\mathrm{Cov} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{z}_i - \bar{\mathbf{z}})(\mathbf{z}_i - \bar{\mathbf{z}})^\top$, where $\mathbf{z}_i$ is the representation given the i-th test data point, N is number of test samples, and $\bar{\mathbf{z}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{z}_i$.

Finally, we apply a singular value decomposition (SVD) on each of the covariance matrix and visualize the top 100 singular values. Results are shown in Fig. 2. As can be observed, for the homogeneous setting, all singular values can evolve to surpass the threshold of e^{-2}. However, under smaller α, not all singular values exceed the threshold. Moreover, as α decreases, the number of singular values exceeding the threshold decreases. This implies that with stronger heterogeneity among local training data, the representation vectors produced by the trained local models will tend to reside in a lower-dimensional space, corresponding to more severe dimensional collapse.

3.2 A Theoretical Explanation for Dimensional Collapse

Given the empirical observations made in Sec. 3.1, in this section, we seek to develop a theoretical understanding on why heterogeneous training data causes dimensional collapse for the learned representations. Our analysis focuses on local training of one arbitrary client without loss of generality. Specifically, we first analyze the dynamics of the model weights during the training via gradient descent. This analysis shows how heterogeneous training data drives the model weights to be biased towards being low rank, which eventually leads to dimensional collapse for the representations.

3.2.1 Setups and Notations

We denote number of training samples as N, the dimension of input data as d_{in}, total number of classes as C. The i-th example is denoted as $X_i \in \mathbb{R}^{d_{\text{in}} \times 1}$, and its corresponding one-hot encoded label is $\mathbf{y}_i \in \mathbb{R}^{C \times 1}$. The collection of all N training examples is denoted as $X = [X_1, X_2 \ldots, X_N] \in \mathbb{R}^{d_{\text{in}} \times N}$, and all the N training labels is denoted as $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N] \in \mathbb{R}^{C \times N}$.

For simplicity in exposition, we follow [4, 5, 15] and analyze linear neural networks (without nonlinear activation layers). We consider $(L+1)$-layer ($L \geq 1$) linear neural networks trained using the cross entropy loss. The weight matrices of the $L+1$ layers after the t-th optimization step are denoted as $W_i(t)$ for $i = 1, 2, \ldots, L+1$, and we have:

$$W_i(t+1) - W_i(t) = -\eta \frac{\partial}{\partial W_i} \ell(W_1(t), \ldots, W_{L+1}(t)), \tag{1}$$

where ℓ denotes the cross-entropy loss and $\eta > 0$ is the learning rate.

In addition, we denote $\mathbf{z}_i(t) \in \mathbb{R}^{d \times 1}$ (d being number of dimension of representations) and $\gamma_i(t) \in \mathbb{R}^{C \times 1}$ as the output representation vector and output softmax probability vector given X_i at the t-th gradient descent step, respectively. We have

$$\gamma_i(t) = \mathrm{softmax}(W_{L+1}(t)\mathbf{z}_i(t)) = \mathrm{softmax}(W_{L+1}(t)W_L(t)\ldots W_1(t)X_i). \tag{2}$$

We define $\mu_c = N_c/N$, where N_c is number of data samples belonging to class c. We denote $\mathbf{e}_c$ as the C-dimensional one-hot vector where only the c-th entry is 1. In addition, let $\bar{\gamma}_c(t) = \frac{1}{N_c}\sum_{i=1}^{N}\gamma_i(t)\mathbb{1}\{\mathbf{y}_i = \mathbf{e}_c\}$ and $\bar{X}_c = \frac{1}{N_c}\sum_{i=1}^{N}X_i\mathbb{1}\{\mathbf{y}_i = \mathbf{e}_c\}$, where $\mathbb{1}\{\cdot\}$ is the indicator function.

3.2.2 Analysis on Gradient Descent Dynamics

Since our goal is to analyze model representations $\mathbf{z}_i(t)$, we focus on weight matrices that directly produce representations (i.e., the first L layers). Specifically, we denote the product of weight matrices of the first L layers as $V(t) = W_L(t)\ldots W_1(t)$ and analyze the behavior of $V(t)$ under the gradient descent dynamics. In particular, we derive the following result for the singular values of $V(t)$.

Theorem 1. *Assume that the initial values of the weight matrices satisfy $W_{i+1}^\top(0)W_{i+1}(0) = W_i(0)W_i^\top(0)$ for any $i = 1, \ldots, L-1$. Let $\sigma_k(t)$ be the k-th largest singular value of $V(t)$ ($k = 1, \ldots, d$). Then we have*

$$\sigma_k(t+1) - \sigma_k(t) = \eta NL\,(\sigma_k(t))^{2-\frac{2}{L}}\sqrt{(\sigma_k(t))^{\frac{2}{L}} + M}\;(\mathbf{u}_{L+1,k}(t))^\top G(t)\mathbf{v}_k(t), \tag{3}$$

where $\mathbf{u}_{L+1,k}(t)$ is the k-th left singular vector of $W_{L+1}(t)$, $\mathbf{v}_k(t)$ is the k-th right singular vector of $V(t)$, M is a constant, and $G(t)$ is defined as

$$G(t) = \sum_{c=1}^{C} \mu_c(\mathbf{e}_c - \bar{\gamma}_c(t))\bar{X}_c^\top, \tag{4}$$

where $\bar{\gamma}_c(t)$ and $\bar{X}_c^\top$ are defined after Eqn. (2).

The proof of Theorem 1 is provided in Appendix A. Based on Theorem 1, we are able to explain why greater data heterogeneity causes $V(t)$ to be biased to become lower rank. Note that strong data heterogeneity causes local training data of one client being highly imbalanced in terms of the number of data samples per class (recall Fig. 1 (a)), which means μ_c will be close to 0 for some classes. Next, based on the definition of $G(t)$ in Eqn. (4), more μ_c's being close to 0 would result in $G(t)$ being biased towards a lower rank matrix (an extreme example is when $\mu_c > 0$ only for one class, then $G(t)$ will be a matrix with rank 1). If $G(t)$ is biased to lower rank, then the term $(\mathbf{u}_{L+1,k}(t))^\top G(t)\mathbf{v}_k(t)$ in Eqn. (3) will only be significant (large in magnitude) for fewer values of k. This is because $\mathbf{u}_{L+1,k}(t)$ and $\mathbf{v}_k(t)$ are both singular vectors, which are orthogonal among different k's. This further leads to the left hand side of Eqn. (3) ($\sigma_k(t+1) - \sigma_k(t)$), which is the difference of the same singular value in two consecutive optimization step, being small for most of the k throughout the training. This means that only a few singular values of $V(t)$ will increase significantly after training, while other singular values remain close to their small initial values.

Furthermore, $V(t)$ being biased towards being lower rank will directly lead to dimensional collapse for the representations. To see this, we simply write the covariance matrix of the representations in terms of $V(t)$ as

$$\text{Cov} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{z}_i(t) - \bar{\mathbf{z}}(t))(\mathbf{z}_i(t) - \bar{\mathbf{z}}(t))^\top = \frac{1}{N}V(t)\sum_{i=1}^{N}(X_i - \bar{X})(X_i - \bar{X})^\top V(t)^\top. \tag{5}$$

From Eqn. (5), we observe that if $V(t)$ evolves to being a lower rank matrix, Cov will also tend to be lower rank, which corresponds to the stronger dimensional collapse observed in Fig. 2.

To further validate our theoretical analyses, we experimentally show how singular values of both $V(t)$ and representation covariance matrix evolve for linear neural networks when trained under different heterogeneity. We partition the MNIST dataset into 10 clients by with $\alpha \in \{0.1, 0.5, \infty\}$ and train a three-layer linear neural network in each setting. All three trained models share the same parameter initialization for the sake of consistency. For the three trained models and the randomly initialized model, we plot singular values of V (in this case, $V = W_2W_1$) and singular values of representation covariance matrix in Fig. 3(a) and Fig. 3(b), respectively.

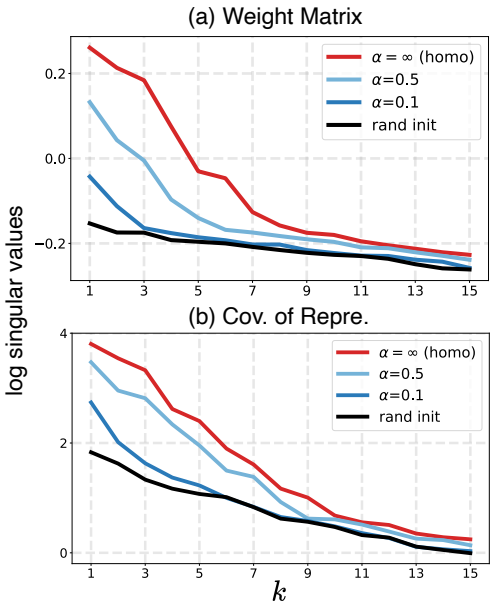

Figure 3: Given the three linear neural networks trained with $\alpha \in \{0.1, 0.5, \infty\}$ and the randomly initialized one, we plot: **(a)** singular values of V of the four models; **(b)** singular values of the representation covariance matrix of the four models. Representations are computed over MNIST test set. The x-axis (k) is the index of the singular values and the y-axis is the logarithm of singular values.

As shown in the Fig. 3(a), in the most heterogeneous case with $\alpha = 0.1$, only 2 singular values of V increase significantly after training while other singular values remain relatively unchanged (corresponding to Theorem 1). This further results in the similar observation for representation covariance matrix under $\alpha = 0.1$ in Fig. 3(b) (corresponding to Eqn. (5)). In contrast, as α increases, more singular values of V increase significantly, resulting in more singular values of the representation covariance matrix being larger, signifying a smaller degree of dimensional collapse.

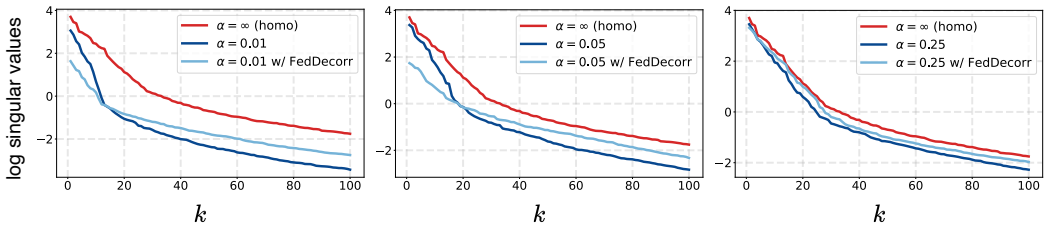

Figure 4: FEDDECORR **mitigates dimensional collapse.** Under the settings of $\alpha = 0.01/0.05/0.25$, we apply FEDDECORR and plot singular values of representation covariance matrix. Results of homogeneous setting is also plotted for comparison. Representations are computed on CIFAR100 test data. The x-axis (k) is singular value index, and the y-axis is singular values in logarithmic-scale.

4 Mitigating Dimensional Collapse with FEDDECORR

Motivated by the above observations and analyses on dimensional collapse caused by data heterogeneity in federated learning, we now explore how to prevent excessive dimensional collapse and thus improve the performance of federated learning under data heterogeneity.

One natural way to prevent dimensional collapse is to add the following regularization term on the representations during training

$$L_{\text{singular}} = \frac{1}{d}\sum_{i=1}^{d}\left(\lambda_i - \frac{1}{d}\sum_{j=1}^{d}\lambda_j\right)^2, \tag{6}$$

where λ_i is the i-th singular value of the covariance matrix of the representations. Essentially, L_{singular} penalizes the variance among the set of singular values thus discouraging tail singular values from collapsing to 0, mitigating dimensional collapse. However, this regularization term is not practical as it requires calculating all the singular values, which is computationally expensive.

Therefore, to derive a training objective that is easy-to-compute, we first apply the z-score normalization on all the representation vectors $\mathbf{z}_i$ as follows: $\hat{\mathbf{z}}_i = (\mathbf{z}_i - \bar{\mathbf{z}})/\sqrt{\text{Var}(\mathbf{z})}$. This results in the covariance matrix of $\hat{\mathbf{z}}_i$ being equivalent to its correlation matrix (i.e., the matrix of correlation coefficients). The following proposition suggests a more natural cost function to regularize.

Proposition 1. *For a d-by-d correlation matrix K and its singular values $(\lambda_1, \ldots, \lambda_d)$, we have:*

$$\sum_{i=1}^{d}\left(\lambda_i - \frac{1}{d}\sum_{j=1}^{d}\lambda_j\right)^2 = \|K\|_{\text{F}}^2 - d. \tag{7}$$

The proof of Proposition 1 can be found in Appendix B. This proposition suggests that regularizing the Frobenius norm of the correlation matrix $\|K\|_{\text{F}}$ achieves the same effect as minimizing L_{singular}. In contrast to the singular values, $\|K\|_{\text{F}}$ can be computed efficiently.

To this end, we propose a novel method, FEDDECORR, which regularizes the Frobenius norm of the correlation matrix of the representation vectors during local training on each client. Formally, the proposed regularization term is defined as:

$$L_{\text{FedDecorr}}(w, X) = \frac{1}{d^2}\|K\|_{\text{F}}^2, \tag{8}$$

where w is the vector of model parameters, K is the correlation matrix of the representations. The overall local objective of each client is

$$\min_{w} \ell(w, X, \mathbf{y}) + \beta L_{\text{FedDecorr}}(w, X), \tag{9}$$

where ℓ is the cross entropy loss, and β is the regularization coefficient of $L_{\text{FedDecorr}}$.

To visualize the effectiveness of $L_{\text{FedDecorr}}$ in mitigating dimensional collapse, we revisit the experiments of Fig. 2 and apply $L_{\text{FedDecorr}}$ under the heterogeneous setting where $\alpha \in \{0.01, 0.05, 0.25\}$. We plot our results in Fig. 4. Clearly, adding $L_{\text{FedDecorr}}$ encourages the variance among the singular values to be smaller, which corresponds to what we have derived in Proposition 1. More importantly, FEDDECORR encourages the tail singular values to not collapse to 0 and thus effectively mitigates dimensional collapse. Pseudocode of our method is provided in Appendix C.

Method	CIFAR10				CIFAR100			
	$\alpha = 0.05$	0.1	0.5	∞	0.05	0.1	0.5	∞
FedAvg [23]	$64.85_{\pm 2.01}$	$76.28_{\pm 1.22}$	$89.84_{\pm 0.13}$	$92.39_{\pm 0.26}$	$59.87_{\pm 0.25}$	$66.46_{\pm 0.16}$	$71.69_{\pm 0.15}$	$\mathbf{74.54}_{\pm 0.15}$
+ FEDDECORR	$73.06_{\pm 0.81}$	$80.60_{\pm 0.91}$	$89.84_{\pm 0.05}$	$92.19_{\pm 0.10}$	$\mathbf{61.53}_{\pm 0.11}$	$\mathbf{67.12}_{\pm 0.09}$	$\mathbf{71.91}_{\pm 0.04}$	$73.87_{\pm 0.18}$
FedProx [20]	$64.11_{\pm 0.84}$	$76.10_{\pm 0.40}$	$89.57_{\pm 0.04}$	$92.38_{\pm 0.09}$	$60.02_{\pm 0.46}$	$66.41_{\pm 0.27}$	$71.78_{\pm 0.19}$	$74.34_{\pm 0.03}$
+ FEDDECORR	$71.38_{\pm 0.81}$	$\mathbf{81.74}_{\pm 0.34}$	$89.96_{\pm 0.26}$	$92.14_{\pm 0.20}$	$61.33_{\pm 0.19}$	$67.00_{\pm 0.46}$	$71.64_{\pm 0.10}$	$74.15_{\pm 0.06}$
FedAvgM [11]	$71.34_{\pm 0.71}$	$77.51_{\pm 0.58}$	$88.39_{\pm 0.17}$	$91.35_{\pm 0.15}$	$59.64_{\pm 0.20}$	$66.36_{\pm 0.14}$	$71.17_{\pm 0.22}$	$74.20_{\pm 0.16}$
+ FEDDECORR	$\mathbf{73.60}_{\pm 0.82}$	$79.21_{\pm 0.15}$	$88.70_{\pm 0.26}$	$91.33_{\pm 0.13}$	$61.48_{\pm 0.27}$	$66.60_{\pm 0.11}$	$71.26_{\pm 0.21}$	$73.86_{\pm 0.25}$
MOON [18]	$68.79_{\pm 0.69}$	$78.70_{\pm 0.66}$	$90.08_{\pm 0.10}$	$92.62_{\pm 0.17}$	$56.79_{\pm 0.17}$	$65.48_{\pm 0.29}$	$71.81_{\pm 0.14}$	$74.30_{\pm 0.12}$
+ FEDDECORR	$73.46_{\pm 0.84}$	$81.63_{\pm 0.55}$	$\mathbf{90.61}_{\pm 0.05}$	$\mathbf{92.63}_{\pm 0.19}$	$59.43_{\pm 0.34}$	$66.12_{\pm 0.20}$	$71.68_{\pm 0.05}$	$73.70_{\pm 0.25}$

Table 1: **CIFAR10/100 Experiments. Comparison of test accuracy (%) with and without** FEDDECORR**.** We ran the experiments under various degree of heterogeneity ($\alpha \in \{0.05, 0.1, 0.5, \infty\}$) and report our results. All results are (re)produced by us and are averaged over 3 runs (mean $\pm$ std). Bold font highlights the highest accuracy in each column.

5 Experiments

In this section, we first elaborate on experimental setups in Sec. 5.1. Next, we show that FEDDECORR consistently improves over previous methods in Sec. 5.2. Then, we perform ablation study on the number of clients in Sec. 5.3 to show this affects the improvements yielded by FEDDECORR. We compare FEDDECORR with some previously proposed decorrelation methods in Sec 5.4 and show FEDDECORR yields greater improvements. Lastly, we compare the computation cost of various methods and demonstrate FEDDECORR's advantage in terms of its computation efficiency.

Method	TinyImageNet			
	$\alpha = 0.05$	0.1	0.5	∞
FedAvg [23]	$35.02_{\pm 0.46}$	$39.30_{\pm 0.23}$	$46.92_{\pm 0.25}$	$49.33_{\pm 0.19}$
+ FEDDECORR	$40.29_{\pm 0.18}$	$43.86_{\pm 0.50}$	$50.01_{\pm 0.27}$	$52.63_{\pm 0.26}$
FedProx [20]	$35.20_{\pm 0.30}$	$39.66_{\pm 0.43}$	$47.16_{\pm 0.07}$	$49.76_{\pm 0.36}$
+ FEDDECORR	$\mathbf{40.63}_{\pm 0.05}$	$44.19_{\pm 0.14}$	$50.26_{\pm 0.27}$	$52.37_{\pm 0.36}$
FedAvgM [11]	$34.81_{\pm 0.09}$	$39.72_{\pm 0.11}$	$47.11_{\pm 0.04}$	$49.67_{\pm 0.25}$
+ FEDDECORR	$39.97_{\pm 0.23}$	$43.95_{\pm 0.26}$	$50.14_{\pm 0.11}$	$52.05_{\pm 0.37}$
MOON [18]	$35.23_{\pm 0.26}$	$40.53_{\pm 0.28}$	$47.25_{\pm 0.66}$	$50.48_{\pm 0.57}$
+ FEDDECORR	$40.40_{\pm 0.24}$	$\mathbf{44.20}_{\pm 0.22}$	$\mathbf{50.81}_{\pm 0.51}$	$\mathbf{53.01}_{\pm 0.45}$

Table 2: **TinyImageNet Experiments. Comparison of test accuracy (%) with or without** FEDDECORR**.** We run under various degrees of heterogeneity, specifically with $\alpha \in \{0.05, 0.1, 0.5, \infty\}$) and report our results. All results are (re)produced by us and are averaged over 3 runs (mean $\pm$ std is reported). Bold font highlights the highest accuracy in each column.

5.1 Experimental Setups

Datasets: We adopt three popular benchmarks, namely CIFAR10, CIFAR100, and TinyImageNet. CIFAR10 and CIFAR100 both have 50,000 training samples and 10,000 testing samples, and the size of each image is 32×32. TinyImageNet contains 200 classes, with 100,000 training samples and 10,000 testing samples, and each image is 64×64. The method that generates local data for each client was introduced in Sec. 3.1.

Implementation Details: Our code is based on the code of [18] (under MIT license). For all experiments, we use the model MobileNetV2 [25]. We run 100 communication rounds for all experiments on CIFAR10/100 and 50 communication rounds for all experiments on TinyImageNet. We conduct local training for 10 epochs in each communication round using SGD with learning rate of 0.01, SGD momentum of 0.9, and batch size of 64. Weight decay is set to 10^{-5} for CIFAR10 and 10^{-4} for CIFAR100/TinyImageNet. We apply the data augmentation of [8] in all CIFAR100/TinyImageNet experiments. The regularization coefficient of FedProx [20] and MOON [18] is tuned to be 0.001 and 1.0, respectively. The server momentum of FedAvgM [11] is tuned to be 0.5. The regularization coefficient of FEDDECORR (i.e., β in Eqn. (9)) is set to 0.1.

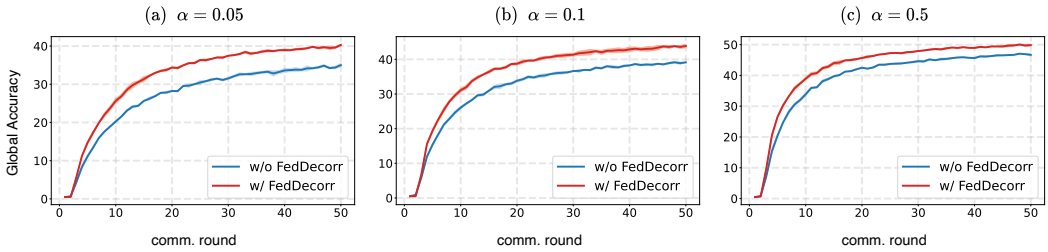

Figure 5: Global model testing accuracy (%) throughout federated learning process with and without FEDDECORR. The y-axis is testing accuracy and the x-axis is number of communication rounds. Results are reported on TinyImageNet, and the baseline is FedAvg. Results are averaged over 3 runs. Shading areas denote standard deviation.

5.2 FEDDECORR Improves Baseline Methods

We apply FEDDECORR on four baseline methods, namely FedAvg [23], FedAvgM [11], FedProx [20], and MOON [18] to validate the effectiveness of our method. We partition the three benchmark datasets (CIFAR10/CIFAR100/TinyImageNet) into 10 clients with $\alpha \in \{0.05, 0.1, 0.5, \infty\}$. Since $\alpha = \infty$ is the homogeneous setting where local models should be free from the hazard of dimensional collapse, we only expect FEDDECORR to perform on par with the baselines in this setting.

We show CIFAR10/100 results in Tab. 1 and TinyImageNet results in Tab. 2. From the results, we can see that for all of the heterogeneous settings on all datasets, the highest accuracies are all achieved by adding FEDDECORR on top of one certain baseline method. In particular, in the strong heterogeneous settings where $\alpha = 0.05/0.1$, adding FEDDECORR yields significant improvements of around $\mathbf{2}\% \sim \mathbf{9}\%$ over baseline methods on all datasets. On the other hand, for the less heterogeneous setting of $\alpha = 0.5$, the problem of dimensional collapse become less prominent as discussed in Sec 3, leading to less improvements from FEDDECORR. Such decreases in improvements are also observed in FedProx, FedAvgM, and MOON when comparing to the naïve FedAvg. In addition, surprisingly, in the homogeneous setting of $\alpha = \infty$, FEDDECORR still produces around 2% of improvements on TinyImageNet. We conjecture that this is because TinyImageNet is much more complicated than CIFAR10/100, and some other factors besides heterogeneity of label can cause undesired dimensional collapse in federated learning setup. Therefore, federated learning on TinyImageNet can benefit from FEDDECORR even in homogeneous setting.

To further demonstrate the advantages of FEDDECORR, we plot how testing accuracy of the global model evolves throughout federated learning process with and without FEDDECORR in Fig. 5. In this figure, if we set a certain value of the testing accuracy as a threshold, we see that adding FEDDECORR can significantly reduce number of communication rounds needed to achieve the given threshold. This further shows that FEDDECORR not only improves the final performance, but also greatly boosts communication efficiency in federated learning.

# clients	Method	$\alpha = 0.05$	0.1	0.5
10	FedAvg	35.02	39.30	46.92
	+ FEDDECORR	40.29	43.86	50.01
20	FedAvg	31.21	35.30	43.64
	+ FEDDECORR	39.41	41.27	46.17
30	FedAvg	26.20	30.88	37.22
	+ FEDDECORR	36.50	39.02	44.38
40	FedAvg	24.10	27.19	32.75
	+ FEDDECORR	34.14	36.81	39.60

Table 3: **Ablation study.** Based on FedAvg and TinyImageNet, we run experiments with and without FEDDECORR with different number of clients and amounts of data heterogeneity.

5.3 Ablation study on number of clients

Next, we study whether the improvements brought by FEDDECORR still pertains as number of clients increases. We partition TinyImageNet into $10/20/30/40$ clients according to different α, and then run FedAvg with and without FEDDECORR. Results are shown in Tab. 3. From this result, we see that as the number of clients increases from 10 to 40, both FedAvg and FedAvg with FEDDECORR show decreasing trend in performance. However, in the case with FEDDECORR, the performance decreases much slower than that of FedAvg. In another word, the performance improvements yield from FEDDECORR increase from around $\mathbf{3}\% \sim \mathbf{5}\%$ to around $\mathbf{7}\% \sim \mathbf{10}\%$ with

the rise of number of clients. Therefore, interestingly, we show through experiments that the improvements brought by FEDDECORR can be even greater under the more challenging settings with more clients. This experiments demonstrate the potential of FEDDECORR being applied to real world federated learning settings with massive number of clients.

5.4 Comparison with Other Decorrelation Methods

Some decorrelation regularizations such as DeCov [7] and Structured-DeCov [31] were proposed to improve generalization in standard classification tasks. Both of these methods directly operate on the covariance matrix of the representations instead of the correlation matrix like our FEDDECORR . To compare our FEDDECORR with their decorrelation methods, we follow the same procedure as in FEDDECORR and apply DeCov and Structured-DeCov during local training. Our experiments are based on TinyImageNet and FedAvg. TinyImageNet is partitioned into 10 clients according to various α. Results are shown in Tab. 4. Surprisingly, we see that unlike our FEDDECORR which steadily improves the baseline, adding DeCov or Structured-DeCov both degrade the performance in federated learning. We conjecture that this is because directly regularizing covariance matrix can be highly unstable, leading to undesired modification on representations. This experiment shows that our design of regularizing *correlation matrix* instead of the *covariance matrix* is of vital importance.

	FedAvg	DeCov [7]	St.-Decov [31]	FEDDECORR
$\alpha = 0.05$	35.02	32.88	32.04	40.29
$\alpha = 0.1$	39.30	37.29	37.74	43.86
$\alpha = 0.5$	46.92	46.29	45.85	50.01

Table 4: **Comparison with other decorrelation methods.** Based on FedAvg and TinyImageNet, we use different decorrelation regularizations in local training.

5.5 Computation Efficiency

We demonstrate FEDDECORR's advantage in terms of computation efficiency. We compare FEDDECORR with some other methods that also apply additional regularization terms during local training such as FedProx and MOON. We partition CIFAR10, CIFAR100 and TinyImageNet into 10 clients with $\alpha = 0.5$ and report the total computation time required for one round of training for FedAvg, FedProx, MOON, and FEDDECORR . Results are shown in Tab. 5. All results are produced with a NVIDIA Tesla V100 GPU. We see that FEDDECORR incurs a negligible computation overhead on top of the naïve FedAvg, while FedProx and MOON introduce about $0.5 \sim 1\times$ additional computation cost. The advantage of FEDDECORR in terms of efficiency is mainly because it is only a painless regularization on output representation vectors of the model, without requiring computing parameter-wise regularization like FedProx nor extra forward passes like MOON.

	CIFAR10	CIFAR100	TinyImageNet
FedAvg	6.7	6.9	25.4
FedProx	12.1	12.3	33.2
MOON	12.2	12.7	38.1
FEDDECORR	6.9	7.1	25.7

Table 5: **Comparing computation time.** We report the total computation time (in minute) for one round of training on the three datasets for FedAvg, FedProx, MOON, and FEDDECORR. Here, FEDDECORR stands for applying FEDDECORR on FedAvg.

6 Conclusion

In this work, we study representations of locally trained models under federated learning with heterogeneous data. Through extensive empirical observations and theoretical analyses, we show that stronger data heterogeneity results in more severe dimensional collapse for representations. Motivated by this, we propose a novel method called FEDDECORR to mitigate dimensional collapse during local training, and thus improving federated learning under heterogeneous data. Extensive experiments show that FEDDECORR yields consistent improvements over previous baseline methods. One limitation of our work is that we only empirically show FEDDECORR is effective in improving the performance of heterogeneous federated learning. Establishing theoretical guarantees is left as future work.

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

[2] Maruan Al-Shedivat, Jennifer Gillenwater, Eric Xing, and Afshin Rostamizadeh. Federated learning via posterior averaging: A new perspective and practical algorithms. *arXiv preprint arXiv:2010.05273*, 2020.

[3] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[4] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*, pages 244–253. PMLR, 2018.

[5] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019.

[6] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[7] Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*, 2015.

[8] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[9] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33:3557–3568, 2020.

[10] Filip Hanzely, Slavomír Hanzely, Samuel Horváth, and Peter Richtárik. Lower bounds and optimal algorithms for personalized federated learning. *Advances in Neural Information Processing Systems*, 33:2304–2315, 2020.

[11] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[12] Tianyu Hua, Wenxiao Wang, Zihui Xue, Sucheng Ren, Yue Wang, and Hang Zhao. On feature decorrelation in self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9598–9608, 2021.

[13] Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 791–800, 2018.

[14] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7865–7873, 2021.

[15] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*, 2021.

[16] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[17] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2021.

[18] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722, 2021.

[19] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021.

[20] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[21] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

[22] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34, 2021.

[23] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[24] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Bjorn Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. In *International Conference on Machine Learning*, pages 8242–8252. PMLR, 2020.

[25] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[26] Yujun Shi, Kuangqi Zhou, Jian Liang, Zihang Jiang, Jiashi Feng, Philip Torr, Song Bai, and Vincent YF Tan. Mimicking the oracle: An initial phase decorrelation approach for class incremental learning. *arXiv preprint arXiv:2112.04731*, 2021.

[27] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

[28] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[29] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[30] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. Addressing class imbalance in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10165–10173, 2021.

[31] Wei Xiong, Bo Du, Lefei Zhang, Ruimin Hu, and Dacheng Tao. Regularizing deep convolutional neural networks with a structured decorrelation constraint. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 519–528. IEEE, 2016.

[32] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261. PMLR, 2019.

[33] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

[34] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M Alvarez. Personalized federated learning with first order model optimization. *arXiv preprint arXiv:2012.08565*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Discussed in Conclusion section.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Federated learning enables the protection of data privacy, therefore, there is no significant negative social impact of this work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]

(b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include our code in the supplemental material
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We only used public available data and code.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We only used public available data and code.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]